Time Sequence Summarization: Theory and Applications

Quang-Khai Pham

UNIVERSITE DE NANTES

ÉCOLE DOCTORALE

"SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE MATHEMATIQUES"

Année: 2010

Thèse de Doctorat de l'Université de Nantes

Spécialité : INFORMATIQUE

Présentée et soutenue publiquement par

Quang-Khai Pham

le 09 juillet 2010

à l'École polytechnique de l'Université de Nantes

Time Sequence Summarization: Theory and Applications

Jury

President	Pr. Michel SCHOLL	CNAM Paris	
Rapporteurs	Pr. Bernd AMANN	LIP6, Université Paris 6	
	Pr. Mohand-Said HACID	LIRIS, Université Claude Bernard Lyon 1	
Examinateurs	Pr. Boualem BENATALLAH	UNSW, Sydney, Australie	
	Pr. Noureddine MOUADDIB	LINA, Université de Nantes	
	Dr. Guillaume RASCHIA	LINA, Université de Nantes	

Directeur de thèse	:	Pr. Noureddine MOUADDIB
Co-directeur de thèse	:	Pr. Boualem BENATALLAH
Encadrant de thèse	:	Dr. Guillaume RASCHIA
Laboratoire	:	LINA UMR 6241

2, rue de la Houssiniere. BP 92209. 44322 Nantes, Cedex 03. N° ED 503-095

Time Sequence Summarization: Theory and Applications

Résumé de séquences d'événements: théorie et applications

Quang-Khai Pham

⋈

favet neptunus eunti

Université de Nantes
&
University of New South Wales

To my parents and family,
who trusted and supported me,
who believed in me and made all this possible,

Acknowledgements

This PhD work was achieved in a Cotutelle program between the University of Nantes, in the Atlas-Grim group, and the School of Computer Science and Engineering (CSE) at the University of New South Wales (UNSW), in the Service-Oriented Computing (SOC) group. Working in both groups has been a great pleasure, a formidable experience and a unique privilege.

First of all, I would like to thank my supervisors Professor Noureddine Mouaddib and Professor Boualem Benatallah who made this Cotutelle program possible. My French supervisor, Professor Noureddine Mouaddib has encouraged me, trusted me and provided me all the means to pursue my work in this Cotutelle program. But, most of all, I would like to express all my gratitude to my Australian supervisor, Professor Boualem Benatallah, for his guidance, encouragements and support during these five years. Boualem has guided me and taught me the way high quality research should be led. I deeply thank him for all the energy and patience it took to teach me. His scientific insight, his outstanding intuition and his exceptional passion have inspired me to reach for the same excellence.

I gratefully thank my co-supervisors and co-authors Dr. Régis Saint-Paul and Dr. Guillaume Raschia for all their invaluable time, energy and efforts. I have really enjoyed the work achieved together, the numerous hours spent in discussions that have bloomed into most interesting ideas. They have allowed me to express, develop and explore my creative skills while enforcing my scientific rigor.

I have special thoughts for all the PhD students, colleagues, administrative staff, visitors, friends from the Atlas-Grim and from the SOC group, reviewers and organizers from conferences I have submitted my work to, and all the people from other horizons whom I have worked with, whom I had great pleasure discussing my work with or who have simply supported or encouraged me. Without any order or preference: Julie Lafoux, Dr. Woralak Kongdenfha, Dr. Hakim Hacid, Dr. Claudia Marinica, Pierrick Bruneau, Dr. Julien Ponge, Dr. Helen Paik, Dr. Fethi A. Rabhi, Dr. Adnene Guabtni, Renato & Elena, Julie Bador, etc..

Last but, by all means not least, I am forever indebted to my family who has encouraged and supported me throughout my life. They have provided me with all the means to access high quality education and given me the thirst and enthusiasm for higher education. In particular, I want to thank my mother who has sacrificed so much for me to achieve what I have today.

Finally, I would like to thank the French National Scientific Research Center (CNRS), the region Pays de la Loire, the Atlas-Grim group, the University of Nantes, the French Ministry of Foreign affairs, the SOC group and the CSE for their financial support of my work through scholarships and travel grants. Without their help, it would have been very difficult to complete this Cotutelle program.

Abstract

Domains such as medicine, the WWW, business or finance generate and store on a daily basis massive amounts of data. This data is represented as a collection of time sequences of events where each event is described as a set of descriptors taken from various descriptive domains and associated to a time of occurrence. These archives represent valuable sources of insight for analysts to browse, analyze and discover golden nuggets of knowledge. For instance, biologists could discover disease risk factors by analyzing patient history, web content producers and marketing people are interested in profiling client behaviors, traders investigate financial data for understanding global trends or anticipating market moves. However, these applications require mining massive sequences, e.g., finance can generate millions of events daily, where the variability of event descriptors could be very high, since descriptors could be extracted from textual contents. For these reasons, discovering golden nuggets of knowledge for such domains with conventional data mining techniques is a challenging task. Recent studies show that data mining methods might need to operate on derived forms of the data, including aggregate values, previous mining results or summaries. Knowledge extracted in such a way is called Higher Order Knowledge. In this thesis work, we propose to address this challenge and we define the concept of "Time sequence summarization" whose purpose is to support chronology-dependent applications to scale on very large data sources. Time sequence summarization uses the content and temporal information of events to generate a more concise, yet informative enough, time sequence that can seamlessly be substituted for the original time sequence in the desired application. We propose a user-oriented approach called TSaR built on a 3-step process: Generalization, grouping and concept formation. TSaR uses background knowledge in the form of taxonomies to represent event descriptors at a higher level of abstraction. A temporal parameter controls the grouping process and only allows events close on the timeline to be gathered. Also, we propose to make the time sequence summarization process parameter-free. For this purpose, we reformulate the summarization problem into a novel clustering problem. The originality of this clustering problem relies on the specificity of the objective function to optimize. Indeed, the objective function takes into account both the content and the proximity of events on the timeline. We present two greedy approaches calledG-BUSS and GRASS to build a solution to this problem. Finally, we explore and analyze how time sequence summaries contribute to discovering Higher Order Knowledge. We analytically characterize the higher order patterns discovered from summaries and devise a methodology that uses the patterns discovered to uncover even more refined patterns. We evaluate and validate our summarization algorithms and our methodology by an extensive set of experiments on real world data extracted from Reuters's financial news archives.

Keywords: Time sequence, Event sequence, Summarization, Categorical data, Data mining, Clustering, Sequential pattern mining

Résumé

Les domaines de la médecine, du web, du commerce ou de la finance génèrent et stockent de grandes masses d'information sous la forme de séquences d'événements. Ces archives représentent des sources d'information très riches pour des analystes avides d'y découvrir des perles de connaissance. Par exemple, les biologistes cherchent à découvrir les facteurs de risque d'une maladie en analysant l'historique des patients, les producteurs de contenu web et les bureaux de marketing examinent les habitudes de consommation des clients et les opérateurs boursiers suivent les évolutions du marché pour mieux l'anticiper. Cependant, ces applications requièrent l'exploration de séquences d'événements très volumineuses, par exemple, la finance génère quotidiennement des millions d'événements, où les événements peuvent être décrits par des termes extraits de riches contenus textuels. La variabilité des descripteurs peut alors être très grande. De ce fait, découvrir des connaissances non triviales à l'aide d'approches classiques de fouille de données dans ces sources d'information prolixes est un problème difficile. Une étude récente montre que les approches classiques de fouille de données peuvent tirer profit de formes condensées de ces données, telles que des résultats d'agrégation ou encore des résumés. La connaissance ainsi extraite est qualifiée de connaissance d'ordre supérieur. À partir de ce constat, nous présentons dans ces travaux le concept de "résumé de séquence d'événements" dont le but est d'amener les applications dépendantes du temps à gagner un facteur d'échelle sur de grandes masses de données. Un résumé s'obtient en transformant une séquence d'événements où les événements sont ordonnés chronologiquement. Chaque événement est précisément décrit par un ensemble fini de descripteurs symboliques. Le résumé produit est alors une séquence d'événements, plus concise que la séquence initiale, et pouvant s'y substituer dans les applications. Nous proposons une première méthode de construction guidée par l'utilisateur, appelée TSaR. Il s'agit d'un processus en trois phases : i) une généralisation, ii) un regroupement et iii) une formation de concepts. TSaR utilise des connaissances de domaine exprimées sous forme de taxonomies pour généraliser les descripteurs d'événements. Une fenêtre temporelle est donnée pour contrôler le processus de regroupement selon la proximité temporelle des événements. Dans un second temps, pour rendre le processus de résumé autonome, c'est-à-dire sans paramétrage, nous proposons une redéfinition du problème de résumé en un nouveau problème de classification. L'originalité de ce problème de classification tient au fait que la fonction objective à optimiser dépend simultanément du contenu des événements et de leur proximité dans le temps. Nous proposons deux algorithmes gloutons appelés G-BUSS et GRASS pour répondre à ce problème. Enfin, nous explorons et analysons l'aptitude des résumés de séquences d'événements à contribuer à l'extraction de motifs séquentiels d'ordre supérieur. Nous analysons les caractéristiques des motifs fréquents extraits des résumés et proposons une méthodologie qui s'appuie sur ces motifs pour en découvrir d'autres, à granularité plus fine. Nous évaluons et validons nos approches de résumé et notre méthodologie par un ensemble d'expériences sur un jeu de données réelles extraites des archives d'actualités financières produites par Reuters.

Mots clés: Séquence d'événements, Résumé, Temps, Données catégorielles, Classification, Fouille de données, Motifs séquentiels

Contents

Résumé étendu

Introduction

Des domaines tels que la médecine, le WWW, les affaires ou la finance génèrent et stockent quotidiennement des quantités massives de données. En effet, la taille des entrepôts de données actuels est de l'ordre du téraoctet et continue de croître. Les entrepôts les plus large sont même susceptibles de passer à l'échelle du pétaoctet d'ici 2012 [Win]. Ces sources de données colossales représentent de précieuses sources d'inspiration pour les analystes qui veulent pouvoir y naviguer, analyser et découvrir des perles de connaissance. Plus précisément, certaines applications sont d'un grand attrait : les prévisions économiques, les prévisions de ventes, l'analyse du recensement, la bourse ou l'analyse de l'utilisation d'Internet sont des applications qui s'appuient sur l'analyse de larges collections de séries de données. Il est coutume de penser qu'il existe une structure interne telle que l'auto corrélation, des tendances ou des variations saisonnières qui pourraient être prises en compte dans le processus de découverte de connaissances. Traditionnellement, ces applications traitent des séries de données numériques monodimensionnelles (voire multidimensionnelles). Par exemple, dans l'étude de l'évolution des marchés financiers, les traders analysent les indices et valeurs boursières à l'échelle de la minute, de l'heure ou de la journée pour identifier de nouvelles opportunités d'investissement ou pour consolider leurs investissements.

Cependant, l'avènement des technologies des bases de données au début des années 80 [McG81] a enrichi ce paysage d'applications et permet l'analyse de formes plus complexes de séquences, c'est-à-dire des séquences de transactions. Dans ces séquences, une transaction est caractérisée par un ensemble d'objets et par une estampille. Par conséquent, l'unité de la donnée n'est plus contrainte d'être monodimensionnelle (voire multidimensionnelle) et numérique, mais peut être multidimensionnelle et catégorielle. Dans ce nouveau contexte, les biologistes peuvent découvrir les facteurs de risque de maladies en analysant l'historique des patients [WRZ05], les producteurs de contenu web et les agences de marketing sont intéressés par le profilage des comportements des clients [SCDT00] et les traders peuvent analyser les informations financières pour comprendre ou pour anticiper les tendances des mouvements de marché [ZZ04].

Un exemple de cette évolution est le paradigme de l'analyse du panier de la ménagère proposé par Agrawal et al. dans [AS95]. Cette nouvelle forme d'analyse, appelée *Extraction de Motifs Séquentiels (EMS)*, consiste à découvrir les séries de collections d'objets qui sont fréquemment achetés ensemble par les clients. Par conséquent, extraction de motifs séquentiels repose sur l'analyse des séquences de transactions: Chaque client est associé à une séquence de transactions ordonnées chronologiquement et chaque transaction est définie par un ensemble d'objets qu'il a acquis et une estampille indiquant quand l'opération a eu lieu. Un exemple concret pour illustrer les connaissances découvertes par le biais de l'extraction de motifs séquentiels dans ces séquences est la suivante: " Dans une enseigne de location de DVD, les clients qui louent " Star Wars IV: un nouvel espoir " loueront "Star Wars V: l'empire contre-attaque" et "Star Wars VI: le retour du Jedi" à

quelques jours d'intervalle."

Nous soulignons ici le contraste qui existe entre les deux types de *séries* évoquées dans la littérature. D'une part, *les séries temporelles* sont des suites d'objets ordonnées par ordre chronologique, où chaque objet est définie sur une (ou plusieurs) dimension(s) discrète(s) à valeur numérique. D'autre part, les *séquences temporelles* sont des suites d'objets/données ordonnées par ordre chronologique où chaque objet est défini sur une (ou plusieurs) dimension(s) à valeur catégorielle et/ou numérique. Plus généralement, les séquences temporelles sont aussi appelés *séquences temporelles d'événements* (*séquences temporelles*, *séquence d'événements* ou *séquence* pour faire court) [PRSP+09].

L'extraction de connaissances à partir de sources de données très volumineuses, par exemple, les entrepôts de données de ventes ou les logs d'usage du Web, est une tâche qui exige l'emploi d'algorithmes rapides et efficaces. Malheureusement, des études récentes sur la fouille du contenu du Web [MTT04, RKS07] ont montré que les performances des techniques de fouille de données les plus pointues rencontrent une limite lorsque les paramètres de l'algorithme de fouille sont trop raffinés. En fait, ces études montrent que lorsque le paramètre de support des algorithmes d'extraction de motifs séquentiels descend en dessous 0,06%, le temps d'exécution de l'algorithme explose exponentiellement. Ce constat a motivé notre travail pour chercher et concevoir des techniques de transformation de données pour créer des représentations plus concises des séquences d'événements et ainsi aider de telles applications à passer à l'échelle. Cette transformation est appelée: *le résumé de séquence d'événements*.

Au cours des trente dernières années, les techniques de synthèse ou de résumé de données ont été élaborées pour diverses sources de données tels que les fichiers textes, les bases de données, les entrepôts de données, les flux de données, etc.. Le but de ces technologies est de représenter la source d'entrée sous une forme plus concise pour des usages qui incluent, mais sans s'y limiter, le stockage, l'analyse, la visualisation ou la navigation exploratoire.

Les techniques développées spécifiquement à des fins de stockage sont plus communément appelées des algorithmes de *compression*. La compression de données considère une donnée comme une suite de bits. L'objectif de la compression consiste donc à transformer une suite de bits A en une suite de bits B plus courte, contenant les mêmes informations, en utilisant un algorithme particulier. Il s'agit d'une opération de codage, c'est-à-dire changer la représentation de l'information dans le but de rendre la représentation compressée plus courte que la représentation originale. Cependant, la compression réalisée transforme structurellement l'information d'origine et ne peut donc pas être traitée sans une transformation inverse au préalable, c'est-à-dire une *décompression*. Cette forme de résumé ne correspond donc pas au contexte de notre travail de thèse et ne sera donc pas considérée dans le reste de ces travaux.

Beaucoup d'efforts ont été investis dans la conception de techniques de résumés qui utilisent la sémantique du contenu des données pour créer des représentations plus concises. La croissance continue et exponentielle des contenus riches en données séquentielles, par exemple, sous forme de fils de nouvelles RSS ou même de fils Twitter, a déclenché la nécessité de faire usage et de gérer la nature temporelle des événements dans les séquences. Á notre connaissance, peu d'intérêt a été accordé à la prise en compte de l'information temporelle dans le but de résumer. À la lumière des contraintes des applications de fouille de données qui reposent largement sur la chronologie des événements des séquences de données, la construction des résumés de séquences d'événements vient avec de nombreux défis à relever:

- Les données qui sont à résumer sont complexes et peuvent contenir aussi bien de

l'information non-structurée, par exemple du texte, que de l'information structurée, par exemple des valeurs d'attributs.

- Le volume des données à traiter croît de manière exponentielle

- Les applications ont tendance à basculer vers le traitement des données en temps réel. Cette contrainte supplémentaire requiert que les algorithmes utilisés aient une complexité algorithmique la plus faible possible, idéalement linéaire ou sous-linéaire.

Dans ce travail de thèse, nous relever les défis inhérents à la construction de résumés de séquences temporelles dans le but d'aider les applications dépendantes du temps à passer à l'échelle. Pour cela, nous présentons les contributions suivantes:

- Nous définissons formellement la notion de résumé de séquence d'événements pour délimiter précisément le périmètre de notre recherche. Nous étudions en détail l'état de l'art des méthodes existantes dans la littérature qui rentrent dans ce périmètre.

- Nous propose une approche orientée utilisateur, appelée TSAR, pour créer un résumé de séquence d'événements. Cette approche orientée utilisateur procède en trois phases: (i) une phase de généralisation, (ii) une phase de regroupement et (iii) une phase de formation de concepts. TSAR produit un résumé, c'est-à-dire une séquence d'événements, où les données sont représentés à un niveau d'abstraction plus élevé et où les événements sont regroupés uniquement s'ils sont *similaires* et *proches* sur la ligne temporelle. La technique proposée a une complexité algorithmique linéaire avec le nombre d'événements à traiter dans la séquence. L'utilisateur contrôle la qualité des résumés d'un point de vue du contenu et du l'ordonnancement des événements dans les résumés grâces à un paramétrage précis de chaque étape du processus de résumé.

- Dans le but de rentre le processus de résumé indépendant des paramètres d'entrée, nous reformulons le problème de résumé de séquence d'événements en un nouveau problème de fouille de données. L'originalité de cette redéfinition du problème réside dans la nature de la fonction objective à optimiser. En effet, ici la fonction objective doit considérer simultanément l'information provenant du contenu des données et l'information temporelle associée aux données. Nous proposons deux approches gloutonnes pour construire des solutions sous-optimales appelées G-BUSS et GRASS. Notamment, GRASS est une technique de classification hiérarchique ascendante basée sur un algorithme parallèle qui exploite les caractéristiques de la fonction objective. Nous montrons à travers nos expériences que GRASS permet d'accélérer le processus de fouille de données de 2 à 3 ordres de grandeurs par rapport à la technique de base G-BUSS.

- Nous proposons d'étudier comment un résumé de séquence d'événement peut être utilisé en pratique sur une application telle que l'extraction de motifs séquentiels sur des données du monde réel. Par conséquent, nous étudions soigneusement les relations qui existent entre les motifs séquentiels découverts dans une collection de séquences originales et les motifs séquentiels *d'ordre supérieur* découverts dans une collection de résumés de séquences d'événements. Nous en proposons une étude analytique ainsi qu'une étude expérimentale. Aussi, nous proposons une méthodologie qui utilise au mieux l'analyse faite précédemment pour découvrir des motifs plus riches et plus précis à partir des motifs d'ordre supérieur extraits des résumés. Cette méthodologie est appelée *Fouille exploratoire*.

- Nous soutenons toutes nos contributions grâce à d'importants jeux d'expériences sur des données réelles extraites des archives de l'actualité financière produite par Reuters en 2003.

Le restant de ce résumé étendu de la thèse est organisé comme suit. Nous définissons dans la Section 1 le concept de *résumé de séquence d'événements* afin de délimiter le champ d'étude de cette thèse. Puis, nous discutons de l'état de l'art des méthodes permettant de construire un résumé à partir de sources de données qui contiennent de l'information temporelle. En outre, nous formalisons le problème de la construction d'un résumé de séquence d'événements. Nous proposons dans la Section 2 la technique orientée utilisateur appelée TSaR. Dans la Section 3, nous proposons d'aborder le problème lié au fait de devoir paramétrer précisément l'algorithme TSaR. Par conséquent, nous reformulons le problème du résumé de séquence d'événements en un nouveau problème de classification. L'originalité de ce problème de classification réside dans la fonction objective à optimiser. En effet, la fonction objective doit prendre en compte simultanément le contenu sémantique et l'information temporelle des données. Nous proposons deux approches gloutonnes, G-BUSS et GRASS. Dans la Section 4, nous proposons un scénario d'application sur des données réelles extraites des archives d'actualités financières de Reuters. Nous étudions dans quelle mesure le résumé de séquences d'événements peut aider une application dépendante du temps telle que l'extraction de motifs séquentiels à découvrir de la connaissance.

1 Résumé de séquences d'événements: état de l'art

1.1 Introduction

Dans cette thèse, nous positionnons notre travail plus proche des idées au coeur de la recherche sur les bases de données. Dans cette perspective, notre compréhension de la tâche de résumé est de construire une nouvelle représentation d'un ensemble de données primitifs. Cette représentation peut ensuite être utilisée pour soutenir d'autres applications complexes qui peuvent éventuellement avoir recours à des traitements intensifs. Plus précisément, nous visons à soutenir les applications dépendantes du temps, à savoir, les applications qui reposent sur l'ordre chronologique des données pour être significatif. Un exemple d'application dépendante du temps est l'Extraction des Motifs Séquentiel [AS95]. Un autre exemple d'application dépendante du temps est Google Finance [Goo]. Il s'agit d'une application de visualisation développée par Google. Sur une ligne temporelle définie par l'analyste, Google Finance fournit aux analystes un outil pour naviguer au travers des indices et des valeurs boursières des entreprises tout en visualisant des informations générales sur les entreprises. Cette information textuelle de base est fourni à l'utilisateur sous la forme d'une séquence de nouvelles qui apparaissent à certains moments jugés *intéressants*, par exemple, lors des sauts de prix. Par conséquent, dans ce travail de thèse, nous considérons le travail de résumé la tâche de transformer, en un objet plus concis et plus abstrait, des objets pris dans une séquence d'objets primitifs.

1.2 Résumé dans...

Nous présentons dans les sections suivantes les principales techniques de résumé rencontrées dans la littérature sur les bases de données (de transactions, relationnelles et temporelles), sur les flux de données et les séquences d'événements.

1.2.1 Les bases de données de transactions

Les principales approches de résumé de données proposées pour les bases de données de transactions, au sens du panier de la ménagère, sont les suivants: Chandola et al. [CK05], SUMMARY [WK06], HYPER [XJFD08] et Wan et al. [WA05]. Les méthodes présentées s'appuient sur l'extraction d'itemsets fréquents (FIM). Le problème de ces approches réside dans la perte d'information lorsque le paramètre de support de l'algorithme d'extraction des itemsets fréquents est fixé trop haut. Il faut noter que ce paramètre doit tout de même être fixé de manière assez fine sinon les temps d'exécution explosent exponentiellement.

Chandola et al. [CK05]

Chandola et al. proposent deux algorithmes de résumé pour les ensembles de données de transactions avec des attributs catégoriels. Dans ce travail, les auteurs formulent le problème de résumé comme un problème d'optimisation qui comporte deux fonctions objectives: (i) gain en compression et (ii) la perte d'information. Les auteurs considèrent que la tâche du résumé est de compresser une série de transactions en un ensemble plus restreint, c'est-à-dire, un ensemble ayant un plus petit nombre de transactions tout en conservant le maximum d'information possible. La première approche proposée consiste à utiliser un algorithme de classification classique pour générer des classes et ensuite de représenter chaque classe par un objet représentatif de la classe. La deuxième approche proposée est une approche Buttom-Up qui utilise la fouille d'itemset fréquents pour représenter des groupes de transactions.

SUMMARY [WK06]

Avec SUMMARY, Wang et Karypis proposent une technique pour générer des résumés de bases de données de transactions basée sur l'extraction d'itemsets fermés fréquents. L'idée est d'identifier les itemsets fermés fréquents, car plus compacts. Ensuite, les itemsets fermés les plus longs sont utilisés pour représenter les transactions de la base de données.

HYPER [XJFD08]

Xiang et al. proposent de résumer les bases de données de transactions grâce à HYPER. C'est une technique qui considère les données d'entrée sous la forme d'une matrice. Le résumer est donc un ensemble d'hyperrectangles calculés de telle sorte à couvrir toutes les cellules de la matrice, c'est-à-dire de la base de données de transactions. Les auteurs démontrent que le problème est NP-difficile et proposent une technique qui s'appuie aussi sur l'identification d'itemsets fréquents.

Wan et al. [WA05]

Wan et al. [WA05] proposent de créer une structure spécifiquement pour supporter les applications d'extraction de motifs séquentiels. Les auteurs construisent une structure appelée "base de données de transaction compacte". A l'aide d'un algorithme basé sur Apriori, les auteurs exploitent cette structure pour extraire des motifs fréquents tout en améliorant le temps de calcul par rapport aux méthodes classiques à base d'Apriori.

1.2.2 Les bases de données relationnelles

Il existe un certain nombre de techniques de résumé spécifiquement développées pour les bases de données relationnelles comme: AOI [Cai91], Fascicles [JMN99], SPARTAN

[BGR01], ItCompress [JNOT04] et SaintEtiQ [RM02, SPRM05].

Induction orientée attribut (AOI) [Cai91]

L'induction orientée attribut (AOI) est une technique initialement proposée par H. Cai, J. Han et N. Cercone [Cai91, HCCC92, HF96]. Le processus a ensuite été amélioré et mis en oeuvre dans DBLearn [HFH+94] puis dans DBMiner [HFW+96].

L'objectif initial de AOI est de traiter les tables de bases de données relationnelles (mais n'est pas limité à) pour la découverte de connaissances. La découverte de connaissances dans une table R($A_1, \ldots, A_n$) est obtenue en réduisant le domaine des attributs A_i, puis en réduisant le nombre de lignes dans R par un procédé de *Généralisation et de Fusion*. Les auteurs supposent des connaissances de base sont disponibles et/ou fournies par des spécialistes du domaine sous la forme de hiérarchies de concepts, autrement dit, des taxonomies. À chaque itération, le processus AOI choisit un attribut A_i et toutes les valeurs d'attribut dans R définie sur A_i sont généralisées une fois en utilisant la taxonomie associée. Le processus est répété jusqu'à ce qu'un critère d'arrêt soit atteint (taux de compression, nombre de tuples restant etc.).

Fascicles [JMN99]

Jagadish et al. [JMN99] ont introduit la notion de *Fascicles* pour réduire la taille des bases de données relationnelles dans le but de réduire au minimum leur stockage. En outre, les auteurs proposent aussi une méthode utilisant les fascicles produits pour extraire des motifs des bases de données compressées. Fascicles s'appuie sur une forme élargie des règles d'association, plus exactement, la fouille d'itemsets fréquents, pour compresser les données. Les auteurs soulignent le fait que, souvent, de nombreux tuples peuvent avoir des valeurs similaires sur plusieurs attributs. Par conséquent, la notion de fascicles capture et forme des groupes à partir de tels tuples. Un fascicule est un sous-ensemble de tuples F dans une relation R pour lesquels il existe un ensemble de k *attributs compacts*. Un ensemble d'attributs A est dit compact lorsque les tuples dans F ont les mêmes valeurs d'attributs pour les attributs dans A.

SPARTAN [BGR01]

SPARTAN [BGR01] est une technique proposée pour compresser les données de grandes tables en utilisant la sémantique des données. Les auteurs font les observations suivantes sur la démarche de Fascicles [JMN99]: Fascicles permet de faire de la compresssion avec perte en regroupant les tuples similaires. Le degré de similarité est spécifié par les paramètres définis par l'utilisateur. Dans cette perspective, les fascicles permettent de garantir des bornes supérieures sur l'erreur de compression. Toutefois, Fascicules procède au niveau des tuples pour détecter les tendances, à savoir, les tuples ayant des valeurs semblables sur plusieurs attributs. Ce procédé orienté tuple pourrait être impossible à réaliser lorsque les données ont de fortes dépendances entre les attributs, c'est-à-dire, entre colonne. Par conséquent, l'idée sous-jacente de SPARTAN est de détecter de telles dépendances de colonne et de représenter les dépendances de façon concise et précise à partir d'un modèle prédictif, à savoir, *les arbres de classification et de régression*.

ItCompress [JNOT04]

La technique de compression itérative, alias ItCompress, est une technique de compression sémantique proposée par Jagadish et al. dans [JNOT04]. L'idée sous-jacente des auteurs est de compresser une table de données notée R en réduisant le nombre de tuples, ou lignes,

dans la table. Cet effet de compression est obtenu en représentant les lignes similaires par une ligne représentative. Dans la pratique, étant donné un paramètre de tolérance d'erreur définie sur chaque attribut, ItCompress trouve un ensemble de lignes représentatives qui correspond le mieux aux tuples dans la table d'origine au sein de l'intervalle de tolérance d'erreur. Toutes les lignes représentatives sont regroupés dans une table notée RR. La table compressée, notée R_C, contient alors trois attributs, à savoir, (i) $ERId$, (ii) Bitmap et (iii) les valeurs aberrantes. $ERId$ attribut aux lignes dans R un identifiant correspondant à l'identifiant d'une ligne représentative dans la table RR. L'attribut Bitmap exprime, pour une ligne donnée dans la table original R, quelles sont les valeurs des attributs de $ERId$ qui ne sont pas à portée de la valeur de tolérance d'erreur. Le dernier attribut contient les valeurs aberrantes, c'est-à-dire celles qui ne sont pas à portée de la valeur de tolérance d'erreur. La question principale dans ItCompress est de choisir une bonne série de lignes représentatives qui maximise la couverture totale des lignes représentées. Les auteurs montrent que ce problème est équivalent au problème des k-centre [GJ79] et est donc un problème NP-difficile.

ItCompress calcule une solution de manière itérative. À la première itération, l'algorithme choisit au hasard un ensemble de k lignes représentatives dans la table d'entrée. Puis, à chaque itération, ce choix aléatoire est améliorée avec l'objectif d'accroître la couverture totale sur la table à compresser. Même si cette technique ne garantit pas l'optimalité, les auteurs montrent qu'ItCompress donne un bon taux de compression sans sacrifier son efficacité.

SaintEtiQ [RM02, SPRM05]

La dernière technique que nous étudions ici est l'approche SaintEtiQ proposé par R. Saint-Paul et al. dans [RM02, SPRM05]. SaintEtiQ n'a pas été spécifiquement conçu pour la compression de données relationnelles à des fins de stockage. L'objectif principal de SaintEtiQ est de produire une vue plus petite et plus concise de bases de données très volumineuses. Cette technique prend en entrée des tuples d'une base de données et produit une forme de connaissance en sortie. Le processus est divisé en deux grandes étapes: la première consiste à réécrire les données brutes en entrée et la seconde est l'étape d'apprentissage. L'étape de réécriture s'appuie sur la théorie des ensembles flous de Zadeh [Zad65] pour transformer des tuples bruts en entrée en tuples candidats. Cette étape de réécriture est réalisée conformément aux connaissances de domaine fournies au système sous la forme de partitions linguistique floues sur les domaines d'attribut. Chaque classe d'une partition est aussi marquée par un descripteur linguistique qui est fourni par l'utilisateur ou un expert du domaine.

Chaque tuple candidat est évalué par un algorithme d'apprentissage. Tout d'abord, le tuple candidat est classée dans la hiérarchie de résumés à partir de la racine de la hiérarchie. L'algorithme de classification conceptuelle trouve le meilleur noeud résumé dans lequel il faut ajouter le tuple candidat suivant une approche top-down. A chaque noeud dans la hiérarchie de résumés, la hiérarchie est modifiée pour intégrer cette nouvelle instance au moyen d'opérations qui peuvent créer ou supprimer des noeuds enfants. Les décisions sont prises sur la base d'un critère d'optimisation locale appelée la qualité de la partition (PQ), qui tente de minimiser la longueur de la description intentionnelle du résumé. Cette description intentionnelle est constituée d'un ensemble flou de descripteurs de domaine sur chaque attribut et est associé à une mesure de possibilité.

1.2.3 Les bases de données temporelles

Les bases de données temporelles sont un domaine de recherche dynamique et fructueux qui a émergé au début des années 60. Le domaine a motivé des centaines de chercheurs à publier plus de deux mille papiers de recherche. Les principaux aspects de ce corpus de travail peuvent être trouvés, et sans s'y limiter, dans une série de bibliographies [BADW82, Soo91, Kli93, SS98, WJW98, Noh04]. Il existe de nombreuses recherches approfondies sur les fondements théoriques des bases de données temporelles. Ces orientations de recherche incluent sans s'y limiter:

- Comment modéliser une sémantique différente de la notion du temps en bases de données relationnelles classiques?
- Créer de nouveaux langages de requête
- Le traitement de requêtes pour bases de données temporelles
- Méthodes d'accès et de stockage pour optimiser les requêtes contrainte par le temps
- Information temporelle incomplète
- Contraintes d'intégrité temporelle, etc.

Á notre connaissance, un intérêt très limité a été accordé aux méthodes pour représenter les bases de données temporelles en une forme plus concise mais informative. Le corpus de travail le plus pertinent pour un résumé des bases de données temporelles sont l'opérateur PACK définie pour des intervalles de temps [DDL02], le travail de Claudio Bettini sur la compression sémantique de données temporelles [Bet01] et Wang et al. pour leur travail sur le temps de transaction dans les systèmes de base de données temporelles [WZZ08].

L'opérateur PACK [DDL02]

L'opérateur PACK considère en entrée une relation unique n-aire notée R, où au moins un attribut est un intervalle de temps. Cet attribut est appelé "au cours". L'opérateur PACK produit une autre relation n-aire qui est connu sous le nom de *forme canonique* de R, notée PACK(R). L'objectif principal de cette forme canonique est de réduire la redondance de la relation originale R. Intuitivement, l'opérateur PACK groupe les tuples dans R qui ont des descriptions identiques, à savoir, les valeurs d'attribut, et qui sont contigus sur la ligne temporelle, c'est-à-dire, les intervalles de temps sur lesquels les tuples se chevauchent.

La compression sémantique de données temporelles de Claudio Bettini [Bet01]

Claudio Bettini propose dans [Bet01] une méthode pour compresser une base de données temporelles. L'auteur s'appuie sur les notions de *granularité temporelle* et sur des *hypothèses sémantiques* pour réaliser la compression. Il s'appuie sur la notion d'hypothèse sémantique qui exprime la façon dont les valeurs de certains attributs évoluent avec le temps et comment ces valeurs changent lorsqu'on les considère en termes de granularité de temps différents. L'auteur considère qu'une base de données temporelles est représentée comme une base de données relationnelle avec un attribut qui indique la période de validité des données. La base de données temporelle compressée est ensuite utilisée pour répondre à des requêtes.

ArchIS [WZZ08]

Archis est un système proposé par Wang et al. dans [WZZ08] pour appuyer des applications temporelles sur des SGBDs. Les auteurs s'attaquent au problème de la conception d'une solution intégrée pour résoudre plusieurs problèmes: (i) l'expression des représentations temporelles et modèle de données, (ii) un puissant langage de requêtes temporelles et des requêtes instantanées, et (iii) l'indexation, le regroupement et l'optimisation de requête pour gérer l'information temporelle efficacement. Afin de soutenir et d'optimiser les requêtes sur les données temporelles de transactions, les auteurs proposent une approche basée sur XML pour représenter l'histoire de base de données dans une forme aggrégée.

1.2.4 Les flux de données

Dans le début des années 90, les chercheurs ont fait le constat qu'un nombre croissant d'applications du monde réel avait besoin de gérer de très grandes quantités de données générées par des sources multiples et de façon automatisée. Des exemples de ces sources comprennent: les indicateurs financiers, les mesures de performance dans le suivi des réseaux informatiques, l'utilisation des logs du Web, les capteurs, les détails des relevés d'appels les télécommunications, etc.. Ces sources sont supposées être de taille infinie, de produire des données de façon continue en temps réel, éventuellement, à des débits très élevés et irréguliers. Pour ces raisons, le meilleur modèle de représentation pour ces données n'est pas celui des relations persistantes mais celui *des flux de données continus transitoires*, appelé aussi *flux* pour faire court. Le traitement des données des flux continus dans le but répondre à requêtes continues, de trouver les articles les plus fréquentes dans les flux ou de découvrir des connaissances pose de nombreux défis. En effet, un tel environnement très dense en données volatiles vient avec des exigences très fortes sur les algorithmes qui traitent les données de flux. Par exemple, puisque les flux sont potentiellement infinis, il est communément admis que les flux ne peuvent pas être stockés et traités en mode hors connexion. Par ailleurs, chaque donnée d'un flux ne peut être considérée qu'une seule fois: soit elle est traitée, soit elle est rejetée.

1.2.5 Travaux précédents

La part du corpus des travaux destinés aux flux de données numériques est hors du périmètre d'étude de cette thèse. Dans ce domaine là, les techniques proposées s'appuient généralement sur la relation d'ordre total qui existe sur les ensembles de valeurs numériques, par exemple, pour calculer les agrégats, les distances, les similarités, etc. Comme il n'existe pas un ordre naturel total sur les valeurs d'attributs catégoriels, ces techniques sont mal adaptées aux données des flux de données catégorielles. Toutefois, en supposant une certaine adaptation des techniques aux attributs catégoriels, les idées sous-jacentes dans certaines approches peuvent encore être utiles, à savoir, l'échantillonnage, le load shedding, les synopsis, etc...

Échantillonnage

L'échantillonnage [Olk93, OR90, Vit85, CMN99, JPA04, HK04, CMR05] ou encore [JMR05, EN06, Agg06b] est une technique simple et bien étudiée. Mais c'est aussi un puissant outil probabiliste pour décider si une donnée doit être traitée ou non. L'échantillonnage permet de capturer dans un espace mémoire réduit et limité les données les plus récentes à partir d'un flux d'entrée. La limite du taux d'erreur d'échantillonnage peut être calculée par l'utilisation d'une fonction de la fréquence d'échantillonnage. La borne supérieure de Hoeffding [Hoe63], c'est-à-dire, la probabilité pour la somme de variables aléatoires de

s'écarter de sa valeur attendue, a souvent été utilisée avec des techniques d'apprentissage machine très rapides [DH01]. Toutefois, les méthodes d'échantillonnage souffrent en général des limitations suivantes: (i) la taille d'un flux en entrée est inconnue et trouver les limites d'erreur exige une analyse particulière, (ii) l'échantillonnage ne permet pas de répondre au problème de la fluctuation des débits des données qui est inhérente aux flux de données et (iii) l'échantillonnage ne permet pas de répondre à applications qui nécessitent de travailler sur des données historisées, par exemple, la classification k-means ou requêtes continues. Néanmoins, l'échantillonnage pourrait encore être utilisé pour maintenir la part des données récentes les plus pertinentes.

Load shedding

Le load shedding [BM03, TCZ+03] est une technique similaire à l'échantillonnage. L'idée est de laisser tomber des portions du flux de données. Le load shedding a été utilisé avec succès pour répondre aux requêtes approximatives. Cependant, cette technique ne convient pas pour tâches de fouille de données car elle pourrait laisser tomber des portions de flux de données qui pourraient représenter ou de contribuer à un motif d'intérêt, par exemple, lors de l'extraction de motifs séquentiels fréquents.

Sketching

L'idée du sketching [BBD+02, Mut05] est de projeter aléatoirement un sous-ensemble des attributs des données. Cette approche peut être comprise comme une forme d'échantillonnage vertical sur le flux de données entrant. Le sketching a été appliqué pour comparer différents flux ou pour répondre à des requêtes d'agrégation.

Synopsis

La création d'une structure de synopsis des données est l'application d'une technique de résumé afin de représenter un flux de données entrant en un autre flux de données (ou en un autre format) qui peut ensuite être utilisé pour une analyse plus approfondie. Il existe un pot-pourri de techniques qui sont conformes à cette définition. Ces techniques comprennent: l'analyse par ondelettes, l'analyse des moments fréquents, les quantiles ou les histogrammes. Nous donnons ici un aperçu des méthodes les plus pertinentes:

- Quantiles [GK01, LXLY04, AM04, GRM05, ZLX+06].
- Moments fréquents [BBD+02].
- Analyse à base d'ondelettes [VW99, CGRS00, GKMS01, GKS04, KM05].
- Histograms mono- ou multi-dimensionnels [PIHS96, JMN99, GK01, GKS01, TGIK03, GGI+02, MM02, WS03, GSW04, Dob05, KNRC05]: Les histogrammes sont des outils efficaces pour capturer rapidement la densité de la fréquence des intervalles (dans le cas d'attributs numériques). Dans le cas de l'attribut catégorique, histogrammes pourrait saisir la fréquence de chaque valeur d'attribut. Toutefois, étant donné qu'il n'existe pas un ordre naturel total sur les attributs catégoriels, les histogrammes de domaines catégoriels perdent le bénéfice de la capture de la densité de fréquence des agrégats tels que sur des intervalles numériques.

Fenêtre glissante

Une technique pertinente pour le résumé de données est le concept de *fenêtre glissante* [BBD+02, BDMO02]. Ce modèle tient compte des éléments les plus récents dans les flux

de données. Comme un flux de données évolue, les éléments à la fin de la fenêtre sont mis au rebut et de nouveaux éléments sont ajoutés. Les fenêtres glissantes ont plusieurs propriétés intéressantes qui leur permettent d'être utilisées pour des applications comme répondre aux requêtes approximatives. La sémantique d'une fenêtre glissante est claire et les utilisateurs comprennent bien sa sémantique. Les fenêtres glissantes sont déterministes et mettent l'accent sur les données fraîches.

Fouille de données – classification

La fouille de données dans les flux de données, notamment en matière de regroupement flux de données, est une autre forme populaire de résumé de flux de données qui a attiré de nombreux travaux de recherche [GMMO00, AHWY03, GMM+03, OWNL04, HXDH03, Agg06a, GLSS06, AHWY07, CCC08]. Guha et al. font de la classification de flux de données en utilisant une approche fondée sur le k-means [GMMO00, GMM+03]. L'approche proposée permet en un seul passage sur le flux de données de générer k classes occupant un faible espace mémoire. Cette approche nécessite $O(nk)$ temps et $O(n\epsilon)$ espace avec k le nombre de classes. Les auteurs ont également prouvé que toute approche k-médian qui permet d'obtenir une approximation constante ne peut pas atteindre un meilleur temps d'exécution que $O(nk)$. Babcock et al. améliorent l'approche de Guha et al. dans [BDMO03], grâce à l'utilisation d'histogrammes exponentiels (EH). Les auteurs améliorent l'approche de Guha et al. en abordant le problème de la fusion des groupes lorsque deux ensembles de centres de classes à fusionner sont éloignés. Pour cela, ils maintiennent une structure EH. Lin et al. [LKLC03] proposent SAX pour transformer des séries chronologiques en une représentation symbolique. La réduction de la dimensionnalité et de la numérosité résulte de l'association (i) d'une technique d'approximation fragmentaire globale suivie (ii) de la représentation de chaque agrégat par un symbole discret, par exemple, {a, b, c,. . . }. Aggarwal et al. [AHWY03, AHWY07] proposent une technique de classification appelé CluStream à base de micro-classes et d'une structure de temps pyramidale pour résumer les données de flux volumineux. Les auteurs construisent un ensemble de micro-classes également appelé *instantanés* à des moments précis dans le temps qui suivent un schéma pyramidal. Une micro-classe est un ensemble d'informations représentées par des *vecteurs fonction de classe* [ZRL96] à laquelle on ajoute des informations sur les estampilles. La structure de temps pyramidale fournit une approche efficace pour répondre aux exigences de stockage et permet de rappeler des statistiques sommaires à partir de différents horizons temporels.

Les techniques énumérées ici montrent une caractéristique commune: elles considèrent toutes la classification de données numériques. Á notre connaissance, peu d'intérêt a été donné à la classification ou à la construction des résumés de flux de données catégorielles. Toutefois, ce besoin est très réel et présent. Les techniques connues pour traiter les flux de données catégorielles semblent avoir principalement émergé ces dernières années avec les contributions suivantes [HXDH03, OWNL04, CL05, PMR06, Agg06a, WFZ+08].

Parmi les techniques les plus prometteuses, Ong et al. [OWNL04] proposent SCLOPE, une approche hybride basé sur CluStream et sur CLOPE [YGY02]. CLOPE est un algorithme de classification initialement développé pour les données de transaction. L'idée derrière CLOPE est de créer des classes tout en optimisant une fonction de critère global qui cherche à accroître la coopération intra-classe en accroissant le rapport hauteur-largeur des histogrammes de classe. Un histogramme de classe est tout simplement un histogramme qui capte la fréquence des articles de transaction des opérations dans une classe. Par conséquent, SCLOPE capitalise sur CluStream en construisant des micro-classes, et en

utilisant la structure de temps pyramidale, et sur CLOPE en abordant la question des attributs catégoriels grâce à l'utilisation des histogrammes de classe. Puisque SCLOPE adresse le problème classification sur les flux de données, les histogrammes de classes sont construits en un seul passage en utilisant la structure de FP-Tree [HY00]. La structure de temps pyramidale permet d'intégrer une logique de temps pour réaliser des constructions d'instantanés. Malheureusement, cette approche n'a pas de mécanisme pour séquentialiser les micro-classes au sein d'un instantané.

Nous avons proposé dans les travaux antérieurs une approche [PMR06] pour résumer les flux de données en utilisant une technique de classification conceptuelle. Comme l'approche est basée sur SaintEtiQ, elle souffre fondamentalement des mêmes limitations. Le résumé produit ne reflète pas la chronologie des données d'entrée et ne peut être directement exploité par les applications dépendantes du temps.

Après de nombreux travaux sur les flux de données numériques, Aggarwal et al. ont changé d'intérêt et se sont tournés dans [Agg06a] vers les flux de données catégorielles. Les auteurs proposent une méthodologie similaire à l'OLAP où des résumés statistiques sont calculés sur les données stockées à intervalle régulier dans le but de répondre à des requêtes répétées. Plus intéressant encore, les auteurs attribuent un poids sensible au facteur temps pour chaque donnée. Grâce à une fonction f de dégradation, ce poids est utilisé pour exprimer le taux de décroissance de chaque donnée et est utilisé pour contrôler la fraîcheur des données historisées. Les pôles sont un ensemble de classes *gouttelettes* qui contiennent un certain nombre d'informations statistiques: (a) nombre de cooccurences des valeurs d'attribut catégorique, (b) nombre de valeurs possibles de chaque attribut catégorique survenant dans un groupe, (c) le nombre de points de données dans la classe, (d) la somme des poids du temps au temps t et (e) l'horodatage du dernier élément rajouté à la classe. Les auteurs ont introduit une méthode élégante pour gérer la décroissance du poids d'une classe, cependant, les gouttelettes de classe sont générées uniquement en fonction de leur similitude avec les données entrantes et non en fonction de leur distance dans le temps.

1.2.6 Les séquences d'événements

Les applications de suivi des utilisateurs ou du système, dans des domaines tels que les télécommunications, la biostatistique ou le WWW, génèrent de grandes quantités de données de séquences appelé *séquences d'événements*. Les séquences d'événements sont composées d'*événements* qui se produisent à un point précis dans le temps. Dans [KT08, KT09], Kiernan et Terzi proposent une définition formelle du résumé de séquence d'événements et donnent les propriétés souhaitables qu'un résumé de séquence d'événements devraient présenter:

1. **La concision et la précision:** Le système de résumé doit construire un résumé concis mais qui décrit précisément les données d'entrée.

2. **Description globale des données:** Les résumés doivent donner une indication de la structure globale de la séquence d'événements et de son évolution dans le temps.

3. **Identification de motifs locaux:** Le résumé doit révéler des informations sur les motifs lacaux: événements ou suspects ou une combinaison d'événements qui se produisent à certains points dans le temps doivent pouvoir être identifiés simplement en regardant le résumé.

4. **Libre de paramètrage:** Aucune paramétrage supplémentaire ne devrait être requis de la part de l'analyste pour que la méthode de résumé puisse donner des résultats instructifs et utiles.

Les auteurs considèrent des séquences particulières où plusieurs événements peuvent se produire à un même point t dans le temps. Ils proposent de s'appuyer sur le principe de Minimum Description Length (MDL) pour produire d'une manière sans paramètres un résumé complet d'une séquence d'événements. L'idée de base est de segmenter la chronologie des événements d'entrée de la séquence en k tronçons contigus, sans chevauchement des intervalles de temps: ces tronçons sont aussi appelé des *segments*. Par la suite, la partie des données correspondant à chaque segment S est décrite grâce à un modèle local M. En un mot, les auteurs formalisent le problème du résumé d'une séquence d'événement en un problème d'identification d'un entier k, de segmenter la séquence d'entrée S en k segments et d'identifier pour chaque segment S le modèle M qui décrit le mieux les données de S tout en minimisant la longueur totale de description. Les auteurs montrent que ce problème peut être résolus de façon optimale en temps polynomial et donnent l'algorithme correspondant. En outre, les auteurs proposent une solution alternative sous-optimale, mais pratique et efficace.

Le résumé établi par Kiernan et Terzi réduit efficacement la séquence d'événements d'entrée en un ensemble de k segments et donne des informations intéressantes sur la densité des données dans chaque segment. Toutefois, puisque le modèle utilisé pour décrire chaque segment S est un modèle probabiliste, les algorithmes de fouille de données ne peuvent s'exécuter directement sur le résumé produit et nécessite une forme de *décompression* avant. En outre, le modèle M utilisé pour décrire chaque segment S perd l'information temporelle des événements au niveau du segment S. Les informations temporelles perdues sont l'ordre dans lequel les événements apparaissent dans la séquence d'événements.

1.3 Résumé de séquences d'événements: définition

Etant donné une séquence d'événements s, un résumé de séquence d'événements est noté $\chi(s) = (s^2_C, s^\star_M)$ et est défini de la manière suivante:

- $s^2_C = \{(Y_i, t'_i)\}$ avec $1 \leq i \leq m \leq n$, est une séquences d'événements où chaque événement (Y_i, t'_i) dans s^2_C est en fait un *groupe* Y_i d'événements $(x_{i,j}, t_{i,j})$, pris dans s et obtenus grâce à une forme de classification C qui tient compte du contenu et de l'information temporelle des événements, auquel est associé une estampille t'_i.

- $s^\star_M = \{(x^\star_i, t'_i)\}$ est une séquence d'événements ordinaire où chaque événement représente un groupe d'événements Y_i et est obtenu for formation de concept. Les concepts $x^\star_i$ sont obtenus en caractérisant les événements dans le groupe Y_i.

s^2_C et $s^\star_M$ peuvent être compris comme *l'extension* et *l'intention*, respectivement, du résumé $\chi(s)$.

Nous avons présenté dans cette partie l'état de l'art des techniques capables de produire de résumés pour des modèles de données relativement similaires aux séquences de données. Cependant, cette étude a mis en évidence l'inadéquation de ces méthodes pour prendre en compte complètement l'aspect temporel des données et de produire un résumé tel que nous le définissons. Nous proposons dans la partie suivante une approche orientée utilisateur pour créer des résumés de séquences d'événements.

2 TSAR: une technique de résumé de séquences d'événements orientée utilisateur

Dans les travaux précédents [KT08], Kiernan et Terzi montrent que le résumé sans paramétrage est une technique souhaitable pour l'utilisateur. Nous pensons que cette propriété est d'importance car les outils de résumé sont conçus pour aider les utilisateurs dans leurs tâches d'analyse de données, en particulier lorsque des ensembles de données volumineux sont considérés. Effet, les utilisateurs non-informaticiens n'ont pas nécessairement les connaissances techniques pour paramétrer et modifier des algorithmes complexes pour répondre aux spécificités et aux exigences de tels ensembles de données. Sinon, le paramétrage requis de l'utilisateur doit être simple et compréhensible [HBMB05] et les structures de données produites informatives au regard de ces paramètres.

L'inconvénient d'une telle méthodologie du point de vue utilisateur est le manque de contrôle sur la sortie produite. Cette limitation pourrait être frustrante si elle va à l'encontre de la manière dont l'utilisateur perçoit la façon dont des groupes d'événements devraient être organisés. Cela surcharge sa capacité à traiter les quantités d'information supplémentaires. Nous nous efforçons de répondre à cette limitation, en termes de flexibilité et des conditions du regroupement des techniques de résumés sans paramétrage. Nous proposons une solution qui implique plus activement l'analyste, mais de manière simple, et qui requiert qu'il décide comment les résumés doivent être construits. Cette approche centrée sur l'utilisateur est appelée TSAR. Le but de TSAR est de fournir aux analystes un outil qui peut produire un résumé informatif et intuitif à partir d'une séquence d'événements.

2.1 Principes

Le principe de base de TSAR est de représenter les descripteurs d'événement sous une forme plus abstraite en utilisant une certaine forme de connaissances de base (BK) pour réduire leur variabilité. Lorsque la variabilité des événements est réduite, certains événements pourraient avoir des descriptions similaires, à savoir, des ensembles de descripteurs identiques à un certain niveau d'abstraction. Si cela se produit et ces événements sont proches sur la ligne temporelle, ils sont rassemblés pour former un groupe ou une classe. Chaque groupe formé est alors représenté par un seul événement, aussi appelé *événement représentatif*, qui est produit (dans notre cas, il est pris dans) à partir du groupe correspondant. Pour cela, TSAR est conçu comme un processus en 3 phases qui s'appuie sur la généralisation des descripteurs d'événements, le regroupement des événements similaires et proches sur la ligne temporelle et sur la formation de concept.

La phase de généralisation est responsable pour représenter les descripteurs des événements d'entrée dans une forme plus abstraite grâce à l'utilisation d'une base de connaissances fournie par l'utilisateur. Cette base de connaissance est entrée sous la forme d'une taxonomie, c'est-à-dire, une hiérarchie de concepts, une pour chaque domaine sur lequel les descripteurs d'événement sont définis. En outre, le niveau d'abstraction de chaque descripteur d'événements est définie par un paramètre appelé *vecteur de généralisation* ϑ. Ce vecteur de généralisation contrôle pour chaque domaine de définition, le niveau auquel les descripteurs d'événements sont à généraliser. Par ailleurs, le vecteur de généralisation contrôle la précision du résumé produit du point de vue du contenu.

Étant donné que l'abstraction des descripteurs d'événements réduit la variabilité des descripteurs, des événements généralisés pourraient éventuellement avoir des descriptions similaires. Dans cette approche, nous considérons des ensembles de descripteurs d'événement comme étant *similaires* si elles sont identiques (à un niveau de généralisation donné). Ainsi, la phase de regroupement est responsable pour le groupement des événements généralisés

qui sont identiques. Ce processus de regroupement est contrôlée par un paramètre temporel w. Intuitivement, le paramètre w indique la *distance maximale* sur la ligne temporelle qui autorise des événements similaires à être regroupés. Vu sous un angle différent, le paramètre w fixe (ou estime) la localité temporelle de chaque événement. Par conséquent, w contrôle la précision du résumé du point de vue temporel (par rapport à la chronologie des événements).

La dernière phase est la phase de formation de concept. Cette phase est responsable de la production ou du choix d'un événement pour représenter un groupe d'événements obtenus dans la phase de regroupement. Puisque nous considérons que deux événements sont semblables dès lors que leurs descripteurs sont identiques, à un niveau d'abstraction donné, la formation de concept est simple. Nous choisissons arbitrairement l'événement le plus ancien du groupe pour représenter le groupe.

2.2 Algorithme

Nous proposons un algorithme incrémental ayant une complexité linéaire et une empreinte mémoire faible. Dans cet algorithme, nous supposons qu'une séquence d'événements peut être contenue en mémoire et nous ne tenons pas compte des environnements où le chargement en mémoire de sous-parties de la séquence doit être fait auparavant. D'un point de vue opérationnel, notre mise en oeuvre de TSAR considère comme entrée une séquence d'événements et retourne en sortie une autre séquence d'événements où: (i) les descripteurs d'événements sont abstraits par la généralisation et (ii) les événements similaires dans une certaine localité temporelle sont regroupés. L'algorithme est paramétré par un ensemble de taxonomies (fournies dans un format XML), un vecteur de généralisation (fourni dans un tableau d'entiers) et un paramètre de localité temporelle (exprimé sous la forme d'une valeur numérique entière qui représente un nombre de groupes). Les événements sont pris en compte dans l'ordre croissant d'estampille, un à la fois. L'algorithme incrémental généralise chaque événement entrant en un événement généralisé. Cet événement généralisé est ensuite comparé aux autres événements qui ont été regroupés au sein de groupes à une distance d_t de w.

TSAR effectue la généralisation, le regroupement et la formation de concept à la volée pour chaque événement entrant. Le processus a une complexité algorithmique linéaire avec le nombre d'événements dans la séquence d'événements en entrée. L'empreinte mémoire de TSAR repose sur deux objets: (i) une table de hachage en mémoire qui regroupe les résultats précalculés de généralisations de descripteurs et (ii) $|W|$ le nombre de groupes au sein de la fenêtre de localité temporelle w. Comme les groupes qui sont à une distance d_t supérieur à w sont sauté par la fenêtre W, ceux-ci peuvent être directement écrites sur le disque ou transmis en flux à d'autres applications. Par conséquent, TSAR n'a besoin de maintenir en mémoire qu'un petit nombre de groupes. Cette utilisation de la mémoire est liée par la largeur de w et la taille moyenne m des descripteurs d'événement.

Il convient de noter que, dans certaines configurations, le nombre d'événements dans un groupe en mémoire pourrait devenir très important. Cela se produit par exemple lorsque le paramètre de localité temporelle est choisi trop grand ou quand il existe de nombreuses répétitions d'événements similaires à travers la séquence. Ce scénario peut être traité par l'exploitation (i) de l'hypothèse que $s^{\star}_{M}$ est la forme la plus utile d'un résumé et (ii) le fait que le processus de formation de concept choisit simplement le premier événement du groupe pour le représenter, c'est-à-dire, le plus ancien événement dans le groupe. Par conséquent, un événement entrant e' qui est semblable à un groupe (Y, t') dans W n'a pas besoin d'être physiquement ajouté au groupe (Y, t'), puisque le groupe (Y, t') sera de toute façon représenté par le plus ancien événement dans (Y, t') par le processus de formation de concept. Enfin, l'empreinte mémoire globale de TSAR est constante et limitée par rapport

à la quantité de RAM disponible sur n'importe quelle machine récente.

2.3 Résultats d'expériences

Grâce à une étude expérimentale approfondie, nous avons montré sur un large jeu de données extraits des archives financières de Reuters que TSAR est une technique de résumé de séquence d'événements qui permet de produire un résumé avec un temps d'exécution linéaire avec le nombre d'événements à traiter. Afin d'évaluer la qualité des résumés produits par TSAR, nous introduisons deux mesures, à savoir la précision sémantique et la précision temporelle d'un résumé. Ainsi, TSAR produit des résumés qui permettent d'atteindre des taux de compression élevés tout en conservant une grande précision temporelle. Le taux de compression peut être amélioré si l'utilisateur autorise le processus de généralisation d'abstraire encore plus les données. Ce choix a une incidence directe sur la précision sémantique du résumé produit. Nous montrons empiriquement qu'à chaque fois le niveau de généralisation des descripteurs d'événement est incrémenté de 1, la précision sémantique du résumé est réduit d'environ 50% et le taux de compression augmenté d'environ 50% aussi. Il appartient ensuite à l'utilisateur de décider, dans le but de parvenir à un taux de compression plus fort, s'il est prêt (i) à accepter plus de perte de précision temporelle et de maintenir la précision sémantique ou (ii) d'accepter la perte de plus de précision sémantique et de maintenir la précision temporelle.

3 Le résumé de séquences d'événements: un nouveau problème de classification

La technique de résumé TSAR présentée précédemment s'articule en trois phases: (i) la généralisation des descripteurs d'événement, (ii) le regroupement des événements ayant des descripteurs généralisés similaires au sein d'une certaine localité temporelle et (iii) la représentation de chaque groupe par un événement représentatif. Si nous regardons l'approche plus en détail, nous pouvons remarquer que les résumés produits par TSAR ont une ressemblance frappante avec des classes qui pourrait être produite par un algorithme traditionnelle de classification de données. En fait, les idées sous-jacentes sont très similaires avec TSAR, à savoir, *diviser ou partitionner* les événements dans une séquence temporelle en *différents* groupes d'événements *similaires*.

En effet, TSAR rassemble seulement ensemble des événements dont les descripteurs généralisés sont identiques. En d'autres termes, TSAR rassemble des événements dont les descripteurs sont *similaires* à un niveau d'abstraction donné; Ce niveau d'abstraction est fixé par le processus de généralisation. Cette opération de regroupement correspond typiquement à la méthodologie des techniques traditionnelles de classification de données. Ces techniques reposent sur les caractéristiques communes des deux objets et une fonction de mesure basée sur une distance pour calculer leur similarité. Les objets sont regroupés lorsque leur similitude correspond à une condition donnée. En outre, dans TSAR, les événements éligibles pour le groupement doivent être situé dans une certaine localité temporelle définie par le paramètre de localité temporelle w. Cette condition peut être considérée comme une forme de segmentation temporelle de la ligne temporelle. Nous présentons donc comment nous pouvons redéfinir le problème de résumé en un nouveau problème de classification.

3.1 Redéfinition du problème du résumé

Sous la lumière de ces observations, le résumé réalisé par TSAR peut facilement être interprété comme une classification qui opère une forme de segmentation temporelle flexible

de la ligne temporelle et qui groupe les événements similaires au sein de chaque segment produit (à noter que les limites de la segmentation ne sont pas bien définies). En contraste avec la classification de données classique, TSaR ajoute une étape de formation de concept pour représenter chaque groupe d'événements par un événement représentatif unique. Par conséquent, une question intéressante à explorer est la suivante: est-il possible de tirer parti des méthodes traditionnelles de classification de données, par exemple, k-means ou la classification hiérarchique ascendante, pour construire un résumé de séquence d'événement tel que nous le définissons dans ces travaux?

Pour répondre positivement à cette question, une méthode de classification de données doit remplir deux conditions au préalable: (i) manipuler des données catégorielles et (ii) gérer la dimension temporelle associée aux événements. En ce qui concerne les données catégorielles, il existe une multitude de travaux de recherche sur la classification des données catégorielles. En fait, nous pensons que le défi à relever, pour exploiter un algorithme classique de classification de données pour réaliser un résumé de séquence d'événements, est la manière dont l'information temporelle est traitée. La manière la plus intuitive et la plus simple de manipuler le temps associé à des événements dans une séquence temporelle est de considérer le temps comme un attribut numérique qui sera traitée comme tout autre attribut. Toutefois, nous montrons les limites de cette hypothèse au travers des deux exemples suivants.

D'une part, supposons que la dimension temporelle est considérée de manière équivalente à tout autre attribut. Mécaniquement, la dimension temporelle a moins de poids lorsque les événements sont décrits sur un grand nombre d'attributs que lorsque les événements sont décrits sur un petit nombre d'attributs. Par conséquent, deux événements très similaires qui se produisent très loin sur la ligne temporelle pourraient être regroupés ensemble.

D'autre part, supposons que la dimension temporelle est considérée comme une dimension discriminante. En d'autres termes, la méthode de classification commencera par segmenter la chronologie des événements, puis une classification sera faite sur les autres attributs dans chaque segment de la ligne temporelle. Cette approche est exactement celle adoptée par Kiernan et Terzi dans [KT08] comme décrit précédemment. L'idée est intéressante, mais la segmentation temporelle peut effectuer une coupe trop nette et pourrait empêcher des classes plus compactes d'être formées.

Intuitivement, une solution à ce problème consiste à attribuer un poids approprié à la composante temporelle. Le défi d'utiliser une méthode de classification classique pour faire du résumé de séquence d'événements peut alors être réduit à la question de la définition d'un poids approprié à la composante temporelle. Nous avons l'intention de relever ce défi et, pour cela, nous reformulons le problème du résumé de séquence d'événements d'une manière qui comprennent totalement la composante temporelle des événements. Ce faisant, nous exprimons en fait un nouveau problème de classification dont la fonction objective à optimiser doit prendre en considération, simultanément, le contenu des événements et le temps associé. Aussi, à travers cette redéfinition du problème, nous adressons la propriété mise en avant par Kiernan et Terzi, c'est-à-dire, le processus de résumé de séquences d'événement doit se faire sans paramétrage.

De ce fait, nous redéfinissons résumé séquence temporelle en utilisant la terminologie consacrée à la classification de données. Pour cela , nous définissons une nouvelle fonction de coût pour évaluer la distance entre les événements sur la ligne temporelle. La nouveauté de ce coût se situe dans la double considération du contenu et le temps associés aux événements. Ainsi, le nouveau problème de classification est présenté comme suit: Étant donné un taux de compression désiré, un résumé de séquence d'événement optimal est un résumé qui a atteint le taux de compression souhaité tout en en minimisant la fonction

de coût qui considère la similarité entre le contenu des événements et la proximité des événements sur la ligne temporelle.

3.2 Solutions

Nous proposons trois approches, à savoir N-TSS, G-BUSS et GRASS pour résoudre le problème présenté. N-TSS est une solution naïve qui construit tous les résumés candidats et choisit ensuite celui qui minimise la fonction de coût. Cette solution est supposée servir de base de comparaison. Cependant, nos expériences préliminaires ont montré que la solution naïve a un temps de calcul prohibitif.

G-BUSS et GRASS sont deux algorithmes gloutons qui construisent des solutions optimales locales. G-BUSS est un algorithme de classification hiérarchique ascendante *classique* qui utilise la fonction de coût introduite. À chaque itération, l'algorithme G-BUSS calcule pour chaque paire d'événements dans la séquence le coût de classification. La paire d'événements qui induit le plus bas coût est sélectionné et les événements sont regroupés ensemble. Le processus est répété jusqu'à ce que le taux de compression désiré soit atteint. Étant donné que G-BUSS a une complexité algorithmique quadratique, nous proposons une technique d'élagage qui permet de réduire en pratique le temps de calcul de l'algorithme. Cette technique d'élagage est basée sur la composante temporelle et permet de définir un voisinage d'événements dans lequel le calcul de coût doit se faire. Cette amélioration permet de réduire en pratique le temps de calcul de G-BUSS de plus d'un ordre de grandeur.

Le deuxième algorithme que nous proposons est appelé GRASS. L'intuition derrière GRASS repose sur deux hypothèses: (i) les événements à proximité sur la ligne temporelle sont plus susceptibles d'être groupés ensemble et (ii) les meilleurs candidats pour être regroupé sont situés dans un voisinage restreint (technique d'élagage introduite pour G-BUSS). De ce fait, GRASS est un algorithme parallèle glouton qui profite de la technique d'élagage introduite pour G-BUSS et du fait que les stations de travail actuelles disposent de processeurs avec au moins deux coeurs d'exécution physique (et quatre pour les processeurs qui implémentent la technologie HyperThreading d'Intel). A chaque itération, GRASS sélectionne au hasard un (ou plusieurs) événements sur la ligne temporelle et décide si cet événement est à regrouper avec un de ses voisins proches. Le processus est répété jusqu'à ce que le taux de compression désiré soit atteint.

3.3 Résultats d'expériences

Nous évaluons et validons nos algorithmes grâce à un ensemble d'expériences sur des données du monde réel: les archives financières de Reuters, soit, une séquence d'environ 10 000 événements. Nous résumons les données d'entrée et évaluons chaque approche sur trois dimensions: (i) le temps de calcul, (ii) le coût de classification total et (iii) la qualité des classes produites, c'est-à-dire, leur homogénéité.

Nos résultats montrent que G-BUSS a bien une complexité algorithmique quadratique mais que l'utilisation de la technique d'élagage introduite, c'est-à-dire, la sélection d'un voisinage restreint lors du calcul du coût de classification, permet de réduire d'un ordre de grandeur le temps d'exécution. De la même manière, sur une échelle logarithmique, GRASS donne l'impression d'être linéaire. Dans les faits en regardant plus précisément les résultats, GRASS a bien un comportement quadratique. Cependant, le fait d'avoir implémenté la technique d'élagage introduite pour G-BUSS et le fait d'utiliser plusieurs coeurs d'exécution pour explorer l'espace de recherche, l'algorithme GRASS permet de produire des classifications de qualité équivalente à celles produites par G-BUSS (il y a

en moyenne une différence de qualité de l'ordre de 1 à 3%). Cependant, GRASS améliore le temps d'exécution de G-BUSS de 2 à 3 ordres de grandeur.

4 Extraction de motifs d'ordre supérieur: application aux données de Reuters

Des domaines tels que la médecine, le WWW, les affaires ou la finance de génèrent et stockent sur une base quotidienne des quantités massives de données. Ces données représentent d'importantes collections de séquences d'événements où les événements sont associés à des ensembles de données *primaires* et à une estampille qui horodate l'événement. Par exemple, Reuters a une équipe de plusieurs milliers de journalistes qui produisent des série d'articles qui relatent les événements survenus à travers le monde. Les informations produites et archivées sont qualifiées de *primaires* ou *brutes* car elles n'ont pas encore été traitées ou analysées. Ces données primaires ont un contenu riche et contiennent souvent un mélange d'informations non structurées, par exemple, du texte libre, et des informations structurées, par exemple, des valeurs définies sur des domaines catégoriels et/ou numériques. Ces archives représentent de précieuses sources d'inspiration pour les analystes avides d'applications de navigation et d'analyse pour y découvrir des perles de connaissance. Par exemple, les biologistes pourraient découvrir les facteurs de risque des maladies en analysant l'historique des patients [WRZ05], les traders peuvent suivre l'évolution des données financières pour mieux comprendre les tendances mondiales et anticiper des mouvements de marché [ZZ04], les producteurs de contenu web et du marketing sont intéressés par le profilage des comportements des clients [SCDT00].

Récemment, Roddick et al. [RSLC08] ont présenté et formalisé le paradigme de *fouille de données d'ordre supérieur* ou *l'exploration de données d'ordre supérieur*. Les auteurs ont observé que dans de nombreux environnements tels que les environnements de traitement en temps réel, la vitesse de calcul pour analyser et extraire de la connaissance atteint certaines limites fixées par les restrictions du matériel et les technologies du firmware. Dans certains cas, les données primaires ne sont pas disponibles ou disponibles pour l'analyse seulement pour une période de temps limitée. En fin de compte, les méthodes d'exploration de données ont besoin, pour fonctionner, de travailler sur une forme dérivée des données. Parmi ces formes dérivées des données on peut citer par exemple: des résultats d'agrégats, de la connaissance résultant de précédentes fouilles de données ou des résumés des données. Par conséquent, le paradigme de fouille de données d'ordre supérieur englobe toutes les méthodes et techniques d'exploration de données qui opèrent sur une forme dérivée des données.

Dans ce contexte, l'extraction de motifs séquentiels est un paradigme d'exploration de données introduite par Agrawal et Srikant dans [AS95] qui est du plus grand intérêt. En effet, le développement de la technologie des codes barres dans les années 90 et la masse croissante des informations provenant des transactions captées par ces technologies ont motivé les auteurs à proposer un nouveau paradigme de fouille de données pour identifier les habitudes récurrentes des clients. Par conséquent, l'objectif initial de extraction de motifs séquentiels consiste à analyser les opérations de vente afin d'identifier les comportements d'achat fréquents: le principe de l'extraction de motifs séquentiels est d'identifier à partir des séquences de transactions ou sous-ensembles de transactions, les séries d'ensembles d'objets aussi appelés *itemsets* qui sont fréquemment achetés par les clients. Ce paradigme d'exploration de données est depuis connu sous le nom *d'analyse du panier de la ménagère*. L'extraction de motifs séquentiels a attiré beaucoup d'attention au cours des 15 dernières années (ces enquêtes [ZB03, TPBM05, HPY05] donnent un aperçu intéressant sur le do-

maine). Le volume croissant des données à traiter et la taille croissante des ensembles de connaissances produites ont motivé les chercheurs à définir des motifs plus compacts, également connu sous le nom *motifs séquentiels fermés* [YHA03, TYH03, WH04, SBI06, BS07].

Agrawal et Srikant ont ensuite étendue et généralisé dans [SA96] le paradigme d'extraction de motifs séquentiels avec le concept d'*extraction de motifs séquentiels généralisés* (EMSG). Les auteurs étendent le paradigme en introduisant des mécanismes d'assouplissement. Par conséquent, l'extraction de motifs séquentiels généralisés peut découvrir à partir des collections de transactions des clients motifs séquentiels où les itemsets sont exprimés à différents niveaux d'abstraction. L'extraction de motifs séquentiels généralisés réalise cet effet par le biais (i) d'un mécanisme d'assouplissement sémantique et (ii) d'un mécanisme d'assouplissement temporel. Le mécanisme d'assouplissement sémantique est chargé de représenter les objets dans les itemsets à différent niveaux de granularité. Ceci est possible grâce à l'utilisation de taxonomies. Dans la pratique, les itemsets sont complétés avec les parents de chaque élément dans l'itemset en se référant aux taxonomies. Cet assouplissement sémantique permet de découvrir des motifs séquentiels où les articles sont exprimés à différents niveaux d'abstraction.

L'assouplissement temporel permet aux algorithmes d'extraction de motifs séquentiels de découvrir des motifs qui n'auraient pas pu être trouvé en utilisant la définition originale de l'extraction de motifs séquentiels. L'assouplissement temporel est exprimé par le biais d'une fenêtre glissante w dans laquelle un ensemble de transactions peut contribuer au support d'un motif candidat p. En d'autres termes, dans la définition initiale de l'extraction de motifs séquentiels, un motif candidat est supporté par une séquence de transactions si chaque itemset dans le motif est supporté par au moins une opération dans la séquence. Dans l'hypothèse de l'assouplissement temporel, un itemset peut être soutenu par l'union d'un ensemble d'au plus w transactions contigües.

Par conséquent, l'assouplissement sémantique permet aux algorithmes d'extraction de motifs séquentiels généralisés de découvrir des connaissances à différents niveaux d'abstraction, par ailleurs dissimulées par la spécificité des données tandis que l'assouplissement temporel permet de découvrir des connaissances qui ne peuvent être trouvées en raison de la définition rigide des événements dans le paradigme traditionnel de l'extraction de motifs séquentiels. Pour ces raisons, l'extraction de motifs séquentiels généralisés peut être facilement comprise comme une forme d'exploration ordre supérieur.

Nous avons présenté précédemment l'approche de résumé TSaR comme une technique dont le but est de construire des résumés pour aider des applications dépendantes du temps à passer à l'échelle. Notamment, une application intéressante est l'extraction de motifs séquentiels. TSaR est conçu comme une étape de prétraitement pour permettre aux applications nécessitant des calculs intenses, tels que l'extraction de motifs séquentiels, à découvrir des connaissances à différents niveaux de représentation. Grâce à l'utilisation de taxonomies, les descripteurs des événements considérés par le système sont exprimés à différents niveaux de granularité. De ce manière, si les techniques classiques d'extraction de motifs séquentiels sont exploitées sur les résumés produits par TSaR, les motifs extraits sont intuitivement comparables à ceux obtenus par les approches d'extraction de motifs séquentiels généralisés. L'exploration de données dans les résumés de séquences d'événements produits par TSaR peut aussi être comprise comme une forme d'exploration de données ordre supérieur. En ce sens, TSaR construit une structure support qui permet l'exploration de données d'ordre supérieur et permet la découverte de *connaissances d'ordre supérieur* d'une manière similaire aux méthodes d'extraction de motifs séquentiels généralisés. A partir de ces observations, nous proposons d'étudier la façon dont un algorithme d'extraction de motifs séquentiels disponible dans le commerce, tel que PrefixSpan [PHMA+01], pourraient exploiter dans la pratique des résumés produits par TSaR

pour extraire des connaissances d'ordre supérieur.

Les contributions abordées dans cette partie applicative sont les suivantes.

Tout d'abord, nous introduisons la notion de *motif d'ordre supérieur*. Nous présentons les motifs d'ordre supérieur comme une forme de connaissance extraite de résumés de séquences d'événements en utilisant une technique classique comme PrefixSpan. Nous analysons avec rigueur les motifs d'ordre supérieur extraits des résumés produits par TSAR par rapport aux motifs découverts à partir des séquences d'événement originales.

Deuxièmement, nous proposons une méthodologie appelée *Fouille exploratoire* qui utilise des motifs d'ordre supérieur découverts sur les résumés produits par TSAR pour découvrir des motifs plus spécifiques, à savoir, des motifs ayant un support encore plus faible. La fouille exploratoire est un processus Drill-Down qui s'appuie sur deux mécanismes: la recombinaison des événements dans un motif et la spécialisation des événements d'un motif. Nous détaillons ces deux mécanismes et discutons de la façon dont ils contribuent à la tâche de découvrir des motifs plus raffinés.

Troisièmement, nous évaluons et validons nos contributions à travers un ensemble important d'expériences sur les archives d'actualités financières écrites par Reuters. L'ensemble des données choisi est un sous-ensemble des données Reuters prétraitées précédemment. L'ensemble des données représente environ 4600 séquences d'événements pour approximativement 100000 événements au total. Nous avons implémenté l'algorithme PrefixSpan et réalisé un large éventail d'expériences d'exploration des données. Les résultats préliminaires viennent appuyer nos revendications qu'une structure support est nécessaire pour les applications de fouille de données dans des domaines tels que la finance. Nous montrons que (i) le temps d'exécution de PrefixSpan explose exponentiellement lorsque l'on réduit le paramètre de support minimum de l'algorithme et (ii) que l'ensemble des résultats produit en sortie pourrait ne pas être humainement exploitable. Puis, nous construisons des résumés TSAR à différents niveaux de précision de contenu et de précision temporelle et explorons les résumés construits de cette manière. Nous montrons que l'exploration de résumés compacts peut permettre la découverte de motifs qui ont un support très faible ($\approx 0,01\%$) tout en améliorant le temps de calcul par un ordre de grandeur. Plus important encore, nous réalisons des opérations de fouille exploratoire à partir de motifs déjà découverts et montrons que l'on peut atteindre l'objectif de trouver des connaissances encore plus spécifiques, à savoir, des motifs ayant un support encore plus faible.

4.1 Définition d'un motif d'ordre supérieur

Roddick et al. ont inventé le terme *fouille de données d'ordre supérieur* dans [RSLC08] pour se référer aux tâches d'extraction de connaissance à partir de données non-primaires. Nous étudions comment les résumés de séquences d'événements peuvent bénéficier une tâche spécifique comme l'extraction de motifs séquentiels. L'idée est de représenter ces sources de données à différents niveaux d'abstraction, puis d'utiliser une méthode classique d'extraction de motifs séquentiels pour découvrir des connaissances. Intuitivement, TSAR est un bon candidat pour cette tâche, car il utilise le contenu des données et des informations temporelles pour créer des résumés qui reflètent la compréhension et les préférences de l'utilisateur. Dans cette perspective, la tâche de découvrir une forme de connaissance sur les résumés TSAR grâce à une méthode d'extraction de motifs séquentiels classique peut être comprise comme une nouvelle forme de fouille de données d'ordre supérieur. Par conséquent, nous étendons la terminologie de Roddick et al. et définissons les motifs extraits de résumés de séquence d'événements comme étant des *motifs d'ordre supérieur*.

4.2 Caractérisation

Nous caractérisons les motifs d'ordre supérieur découverts sur les résumés TSAR et évaluons dans quelle mesure les motifs d'ordre supérieur sont liés à la connaissance découverte sur des séquences temporelles primaires. Nous caractérisons et analysons aussi complètement que possible les relations suivantes: motifs découverts (i) des séquences d'événements originales par rapport aux motifs d'ordre supérieur découverts des séquences généralisées, (ii) le rapport entre les motifs d'ordre supérieur découvert des séquences généralisées par rapport aux motifs d'ordre supérieur découverts des séquences résumées et (iii) le rapport entre les motifs découverts des séquences originales par rapport aux motifs d'ordre supérieur découverts des résumés.

Nous avons établi dans cette section des relations qui lient les motifs séquentiels découverts des séquences temporelles primaires aux motifs d'ordre supérieur découverts des résumés de séquence d'événements. Toutes les relations n'ont pas pu être décrites analytiquement. Cette limitation est dûe au fait que ces relations dépendent profondément des paramètres de résumé, des paramètres de fouille de données et des données elles-mêmes. Tous ces paramètres compliquent grandement la tâche, notamment il est très compliqué d'établir précisément les relations qui lient les motifs d'ordre supérieur découverts des résumés aux motifs découverts des séquences d'origine, à partir du moment où on s'intéresse aux motifs de longueur deux au minimum. Pourtant, nous avons pu identifier les situations et les conditions qui permettent de déterminer si, pour un résumé donné, un motif pourraient être supporté par la séquence d'événements originale sous-jacente.

4.3 Fouille exploratoire: une méthodologie pour découvrir de la connaissance à partir des résumés

Auparavant, nous avons caractérisé les relations qui lient les motifs découverts des séquences originales aux motifs d'ordre supérieur découverts des séquences d'événements généralisés ou résumés. En particulier, nous avons donné une propriété qui stipule que des motifs d'ordre supérieur pouvaient servir à trouver des motifs encore plus précis. Nous comptons sur cette propriété et présentons une méthodologie qui utilise toutes les caractéristiques déjà démontrées pour obtenir des motifs plus spécifiques à partir de motifs d'ordre supérieur découverts des résumés TSAR. Cette méthodologie est appelée *Fouille exploratoire*. La fouille exploratoire est réalisée grâce à deux opérations complémentaires, à savoir: (i) la spécialisation des événements d'un motif et (ii) la recombinaison des événements d'un motif.

Les idées qui guident ces deux opérations complémentaires reposent sur des observations faites à partir de l'utilisation dans TSAR des opérateurs φ_ϑ et ψ_w. D'une part, l'opérateur φ_ϑ réécrit les descripteurs d'événement à un niveau d'abstractio supérieurn. Ce faisant, la variabilité des descripteurs diminue alors que le support de chaque événement généralisé est augmenté. Mécaniquement, les événements qui ont un support très faible dans une séquence d'événement originale, par exemple, en raison de l'utilisation d'un vocabulaire spécialisé, pourrait avoir un support plus important dans son résumé, par exemple, dû à l'association d'un vocabulaire spécialisé à des concepts plus communément utilisés. L'idée de la spécialisation des événements d'un motif est alors d'identifier, dans les résumés de séquence d'événements, un motif p d'intérêt qui aurait un support important et d'essayer de trouver des motifs ayant un support moins élevé en spécialisant certains descripteurs d'événements dans p en utilisant les mêmes taxonomies que celles utilisées lors du processus de résumé.

D'autre part, l'opérateur ψ_w regroupe des événements qui sont similaires et à une cer-

taine distance w les uns des autres sur la ligne temporelle. Comme conséquence directe, les résumés de séquences d'événements sont plus courts que leurs homologues non-résumés et induisent une perte de précision temporelle. Nous avons montré précédemment que cette opération entraîne une perte de capacité de rappel, à savoir, les motifs d'ordre supérieur découverts des résumés de séquences d'événements ne peuvent pas capturer tous les motifs découverts des séquences originales. Par conséquent, l'idée de la recombinaison des événements dans les motifs est d'atténuer les conséquences de l'opérateur de regroupement ψ_w et de produire des motifs candidats à partir de motifs d'ordre supérieur déjà découverts dans les résumés. Ces motifs candidats sont générés par recombinaison des événements qui composent le motif d'ordre supérieur.

4.4 Expériences

Dans cette section, nous expérimentons l'extraction de motifs séquentiels sur les résumés de séquences d'événements produits par TSaR. Nous avons montré dans nos travaux d'analyse qu'il existe un lien fragile entre les motifs découverts des séquences d'événements primaires et les motifs d'ordre supérieur découverts des résumés. Cette connexion dépend grandement de 3 dimensions: (i) les paramètres de résumé, (ii) les paramètres de la fouille de données et (iii) la distribution des données. Il est indéniable qu'extraire des motifs d'ordre supérieur de résumés entraîne une perte de motifs, éventuellement cette perte peut être importante, si l'on se concentre uniquement sur le critère de rappel des motifs qui devraient être découverts des séquences d'événements originales. Toutefois, l'objectif des motifs d'ordre supérieur n'est pas nécessairement de produire exactement les mêmes connaissances que celles découvertes des séquences d'événements originales. Pour cette raison, dans l'application de l'extraction de motifs séquentiels sur les résumés de séquence d'événements, nous pensons qu'il n'y a pas de sens à utiliser les mesures classique et largement acceptées en recherche d'information, à savoir la *précision* et le *rappel*, pour évaluer la qualité des motifs découverts. Nous préférons montrer dans ce travail expérimental que les motifs extraits des résumés permettent de découvrir différents types de connaissances qui seraient autrement restées dissimulées.

Dans cette perspective, il nous semble beaucoup plus intéressant d'évaluer l'utilité des résumés TSaR face à la tâche de découvrir des connaissances très précises et spécifiques. Nous proposons d'expérimenter PrefixSpan sur des résumés TSaR en deux phases: (i) évaluer si les résumés TSaR forment bien une structure support pour les algorithmes comme PrefixSpan, c'est-à-dire, évaluer leur capacité à aider à réduire les temps de calcul, et (ii) effectuer la fouille exploratoire sur les résumés construits par TSaR afin de voir quel type de connaissance peut être découvert. Pour cela, nous avons réutilisé l'ensemble des données d'actualités financières extraites des archives de Reuters. Cependant, comme les algorithmes d'extraction de motifs séquentiels sont des algorithmes très intensifs, nous limitons l'ensemble des données à un mois d'actualités seulement. L'ensemble des données utilisées représente encore approximativement 100000 événements répartis sur 4600 séquences d'événements.

4.4.1 Scénarios d'usage des résumés TSaR

Nos résultats préliminaires montrent que les techniques d'extraction de motifs séquentiels peuvent tirer partie de structures supports comme les résumés, au moins pour réduire les temps de calcul. Toutefois, ces expériences n'informent pas explicitement l'analyste sur la nature, la qualité ou le contenu des connaissances extraites. Pour cette raison, nous présentons dans cette section deux scénarios pour illustrer la façon d'utiliser les résumés TSaR pour extraire des motifs séquentiels.

Scénario 1 (Sc.1): trouver de la connaissance plus abstraite

Le but du premier scénario est de découvrir des motifs d'ordre supérieur des séquences d'événements de l'actualité financière. Ces motifs sont des *tendances* qui informent l'analyste sur le thème général des motifs les plus fréquents. En d'autres termes, les motifs d'ordre supérieur découverts donnent le profil des séries d'événements les plus récurrents parmi les événements survenus. Par conséquent, l'objectif de la découverte des tendances est d'informer l'analyste sur les principaux thèmes des événements fréquents. C'est pourquoi les motifs séquentiels découverts des résumés très compacts, c'est-à-dire, construits avec un niveau de généralisation important et un paramètre de localité temporelle important, donnent de précieux renseignements sur les données sous-jacentes. Si l'analyste a besoin de connaître des tendances à un degré de granularité plus fin, il peut raffiner les résumés TSAR utilisés.

Scénario 2 (Sc.2): découvrir des motifs spécifiques à partir de motifs d'ordre supérieur

Le deuxième scénario est complémentaire au scénario 1. Dans le cas où l'analyste a trouvé un (ensemble de) motif(s) d'ordre supérieur d'intérêt, il faut lui donner l'occasion d'affiner cette connaissance sans nécessiter d'opérer à nouveau une opération d'extraction de motifs séquentiels. Par conséquent, ce scénario permet à l'analyste d'utiliser les connaissances déjà découvertes pour découvrir des motifs plus spécifiques dans un mode Drill-Down utilisant la méthodologie de fouille exploratoire. Nous choisissons un motif d'ordre supérieur d'intérêt et spécialisons les descripteurs de l'événement et/ou recombinons les événements pour générer un ensemble de motifs candidats. Nous calculons le support de ces motifs candidats générés et montrons qu'il est possible de trouver des motifs encore plus spécifiques qui bénéficient d'un support très faible, autrement difficiles à découvrir.

4.4.2 Les scénarios en pratique

Nous construisons des résumés TSAR à partir les données précédentes avec les paramètres de résumé $\vartheta = \langle 1 \rangle$ et $w = 50$. Le résumé $\chi_{\vartheta=\langle 1 \rangle, w=50}(E)$ est notée $E^\star$. Nous explorons les motifs séquentiels ayant pour support $\gamma = 14$ et limitons la longueur maximale des motifs séquentiels à 7. Cette contrainte est ajoutée pour réduire le temps de calcul et le nombre total des motifs extraits. L'ensemble des motifs séquentiels découverts est noté $P_\gamma(E^\star)$. Nous donnons dans le Tableau 1 certains motifs séquentiels d'intérêt découverts dans $E^\star$ et leur support respectif. L'ensemble des descripteurs utilisés pour décrire chaque événement dans le Tableau 1 est donné dans le Tableau 2.

Sc.1: trouver de la connaissance plus abstraite

Dans le premier scénario d'utilisation, nous mettons en évidence la possibilité de découvrir des connaissances générales sous la forme de tendances à partir des résumés compact produits par TSAR. Par exemple, considérons le motif P_3 dans le Tableau 1. De la description des événements donnée dans le tableau 2, des nouvelles qui impliquent *"une sorte d'opération relative aux pays de l'Ouest"* semblent être très fréquentes ($supp_{E^\star}(P_3) = 273$). Plus précisément, le motif $p_{1,3,4}$ a un support de 24 dans $E^\star$ et indique que des événements qui impliquent *"certaines institutions financières"* ou *"des affaires"* sont également fréquents. Ces deux motifs par eux-mêmes fournissent suffisamment d'information sur la tendance de certaines séries d'actualités fréquentes. Ici, la tendance des nouvelles est *"des affaires/opérations financière dans les pays occidentaux"*.

Motif	$supp_{E^\star}$
p_1=$\langle e_1 \rangle$	132
p_2=$\langle e_2 \rangle$	121
p_3=$\langle e_3 \rangle$	273
p_4=$\langle e_4 \rangle$	83
$p_{1,2}$=$\langle e_1, e_2 \rangle$	33
$p_{1,3,4}$=$\langle e_1, e_3, e_4 \rangle$	24
$p_{3,4,1}$=$\langle e_3, e_4, e_1 \rangle$	21

Table 1: Motifs découverts et support

Event	Descripteurs
e_1	{any_operation, financial institution, west}
e_2	{any_operation, sales, west}
e_3	{any_operation, west}
e_4	{any_operation, business, financial institution, west}
$e_{\downarrow 1,1}$	{any_operation, islf, west}
$e_{\downarrow 1,2}$	{any_operation, ins, west}
$e_{\downarrow 1,3}$	{any_operation, fin, west}
$e_{\downarrow 1,4}$	{any_operation, bnk, west}
$e_{\downarrow 2,1}$	{any_operation, bus, west}
$e_{\downarrow 2,2}$	{any_operation, ret, west}
$e_{\downarrow 2,3}$	{any_operation, who, west}

Table 2: Description des événements

Si l'analyste a besoin plus de détails sur les motifs et sur les nouvelles en cause, ce motif d'ordre supérieur découvert dans les résumés pourrait être raffinée grâce à la méthodologie de fouille exploratoire. Tel est l'objet du scénario 2. Nous proposons d'utiliser et d'explorer le motif $p_{1,3,4}$ en plus de détails.

Sc.2: fouille exploratoire

Sc.2.1: recombinaison des événements

Dans le scénario Sc.2, nous essayons de découvrir des tendances plus détaillées à partir des motifs extraits dans le scénario Sc.1 en suivant la méthodologie de fouille exploratoire. Nous commençons par recombiner les événements dans le motif d'ordre supérieur extrait précédemment. Considérons le motif d'ordre supérieur $p_{1,3,4}$ dans le Tableau 1. Le support de $p_{1,3,4}$ dans $E^\star$ est de 24. En fait, quand on extrait des motifs séquentiels de $E^\star$ avec un support minimum de $\gamma = 24$, $p_{1,3,4}$ est l'unique motif qui implique les trois événements e_1, e_3 et e_4. En d'autres termes, les motifs $\langle e_1, e_4, e_3 \rangle$, $\langle e_3, e_1, e_4 \rangle$, $\langle e_3, e_4, e_1 \rangle$, $\langle e_4, e_1, e_3 \rangle$ et $\langle e_4, e_3, e_1 \rangle$ ont un support inférieur à 24 dans $E^\star$.

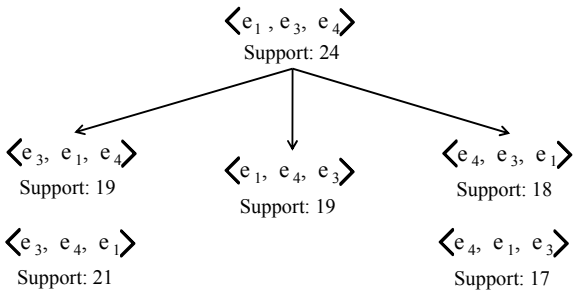


Figure 1: Séquences d'événements recombinés

Par conséquent, nous commençons par recombiner les événements dans le motif $p_{1,3,4}$ et nous produisons toutes les combinaisons possibles des événements de $p_{1,3,4}$. L'ensemble des motifs candidats résultant de la recombinaison et leur support respectif dans $E^\star$ est donné dans la Figure 1. Notez que le support de tous les candidats recombinés est inférieure au support de $p_{1,3,4}$. En moyenne, chaque séquence combinée a un support inférieur à celui du motif $p_{1,3,4}$ d'environ 21%, tandis que le motif candidat qui a le support le plus important

est $p_{1,3,4}$ avec un support égal à 21, soit 12,5% inférieur à $p_{1,3,4}$. Cette observation montre que nous avons été capables de découvrir d'autres motifs qui: (i) contiennent tous les trois événements e_1, e_2 et e_3, et qui (ii) ont un support plus faible que $p_{1,3,4}$, sans avoir à complètement refaire une opération complète d'extraction de motifs.

Sc.2.2: spécialisation des événements d'un motif

Le second mécanisme impliqué dans la fouille exploratoire est la spécialisation des événements d'un motif. Nous illustrons ce deuxième mécanisme en utilisant un motif choisi parmi les connaissances extraites précédemment. Supposons que nous nous intéressons à des motifs qui contiennent les descripteurs *"sales"* et *"financial sector"* du domaine *"Industrial sector"* (Tableau 4.5(b) et Tableau 4.5(a) du Chapitre 4 donnent des détails de la spécialisation des descripteurs *"sales"* et *"financial sector"*, respectivement).

Dans la pratique, sans aucune connaissance préalable, le motif d'ordre supérieur $p_{1,2} = \langle e_1, e_2 \rangle$ est une série de deux événements qui peut être comprise comme suit: *"une sorte d'opération dans le secteur financier dans un pays occidental"* suivi d'*"une sorte d'opération de vente d'une industrie dans un pays occidental"*. Ces deux événements donnent des renseignements généraux sur l'emplacement et la nature de l'industrie concernée. Ces connaissances peuvent être raffinées grâce à la spécialisation des descripteurs *"sales"* et *"financial sector"*. Par conséquent, le motif $p_{1,2}$ est spécialisée en 12 motifs spécialisés candidats. Une remarque intéressante est que parmi ces motifs candidats spécialisés, il y en a un qui a un support de seulement 4.

Nous avons fourni dans cette section une étude expérimentale approfondie sur l'utilisation des résumés TSAR pour l'extraction de motifs séquentiels fréquents avec un algorithme classique appelé PrefixSpan. Nous avons montré que pour certaines applications, telles que les applications financières, l'extraction de motifs séquentiels des données brutes au moyen d'un algorithme classique n'est pas suffisante. Les motifs intéressants, à savoir, les motifs qui impliquent au moins 2 à 4 événements, ne sont extraits qu'à des niveaux de support très faible, c'est-à-dire, $\gamma \leq 0,06\%$. Malheureusement, à ces faibles niveaux de support, le temps de calcul explose et l'utilisateur est submergé par la masse des motifs extraits. D'autre part, les résumés TSAR sont apparus comme des candidats intéressants pour appuyer les algorithmes d'extraction de motifs séquentiels. Les expériences montrent qu'il est possible d'extraire des motifs séquentiels à partir des résumés à des niveaux de support très faibles, c'est-à-dire, $\gamma \approx 0.01\%$, mais cette opération nécessite des résumés compacts. Toutefois, nous avons fourni deux scénarios d'utilisation et démontré que l'application de la méthodologie de fouille exploratoire sur les motifs d'ordre supérieur a permis de découvrir à partir des résumés des motifs séquentiels très précis.

Conclusion

Dans ce travail de thèse, nous avons abordé un certain nombre de défis pour aider des applications dépendantes du temps à passer à l'échelle. Nous avons mis en place et formalisé la notion de "Résumé de séquence d'événements". Puis, nous avons proposé plusieurs techniques qui permettent de produire un résumé de séquence d'événement. La première approche est une technique de résumé orientée utilisateur appelée TSAR. TSAR est conçu pour tenir compte des préférences de l'utilisateur et de sa compréhension des domaines sur lesquels les données sont définies. TSAR produit avec ces préférences une séquence d'événements plus compacte où les événements sont représentés à différents niveaux d'abstraction. L'effet de compression est obtenu par le regroupement de données qui sont semblables à un certain niveau d'abstraction et qui sont proches sur la ligne

temporelle.

Puis, nous avons proposé une approche de résumé sans paramétrage. Pour cela, nous avons reformulé le problème de résumé en un nouveau problème de classification de données. L'originalité de cette reformulation du problème en un problème de classification est que la fonction objective à optimiser doit prendre en compte simultanément le contenu et l'information temporelle des données. Nous proposons deux algorithmes gloutons qui permettent de construire des solutions localement optimales à ce problème, à savoir G-BUSS et GRASS.

Étant donné que les résumés de séquences d'événements ont été conçus pour soutenir les applications dépendantes du temps, nous avons fourni une vaste étude sur l'utilisation des résumés dans le contexte d'extraction de motifs séquentiels. Par conséquent, nous avons complètement caractérisé la connaissance qui pourrait être extraite à partir des résumés TSaR au regard des connaissances qui peuvent être extraites à partir de séquences originales. Aussi, nous avons exploré les connaissances extraites des résumés grâce à la méthodologie introduite: la fouille exploratoire. Nous avons montré sur quelques exemples concrets tirés des archives des actualités financières de Reuters que notre méthodologie nous a permis de découvrir grâce à l'utilisation des résumés, des motifs et des connaissances très précises et concrètes.

List of Figures

List of Tables

Introduction

"Now the reason the enlightened prince and the wise general conquer the enemy whenever they move and their achievements surpass those of ordinary men is foreknowledge"

— Sun Zi, The Art of War, 6^{th} century BC.

January 18th 2006. TechCrunch relayed the following rumor on its website:

> "**YouTube Acquisition Rumors**
> Rumors are flying that Silicon Valley based YouTube (profiled here) has signed an agreement to be acquired. Whoever the buyer may be, it's not News Corp. They have confirmed directly to me it has not acquired YouTube. YouTube raised $3.5 million in venture capital just three months ago from Sequoia. It was founded in February 2005."

During the period January 13th to January 20th 2006, Google's share price lost -14.32%, News Corp. lost -0.52%, Yahoo! lost -15.43% and Microsoft lost -2.86%.

October 6th 2006. TechCrunch France relayed the following rumor on its website:

> "**Des rumeurs d'acquisition de YouTube par Google**
> Nous venons de recevoir un email faisant part d'une acquisition probable de YouTube par Google qui serait en court de finalisation. La rumeur fait état d'un prix d'acquisition de $1,65 milliard. Nous avons appelé un investisseur de la place qui confirme que cette rumeur circule bien...
> *[We just received an email informing us that Google is in talks with Youtube for a takeover plan. The rumor states the takeover could cost $1.65 billion. We contacted a trader who confirmed the rumor was spreading out on the trade floor...]*"

October 7th 2006. The Wall Street Journal confirms Google is in talk with Youtube.

October 10th 2006. The Guardian announces:

> "**Google nets YouTube in $1.65bn takeover**
> The founders of the video website YouTube last night accepted a $1.65bn (£880m) takeover offer from Google for their 20-month-old venture, which has a big online following but has yet to make money..."

During the period September 29th to October 13th 2006, Google's share price gained +6.31%, News Corp. gained +4.60%, Yahoo! lost -3.40% and Microsoft gained +3.72%.

April 14th 2007. Bloomberg announces:

> "**Google's DoubleClick Strategic Move**
> Three billion dollars? On Apr. 13, Google announced that it would pay $3.1 billion for the advertising outfit DoubleClick.[...] DoubleClick has something that Google, for all its money and smarts, doesn't: a vibrant advertising business for banners, **videos**, and other so-called display ads..."

Six months right after spending $1.65 billion, Google announces it would reinvest $3.1 billion to buy a company specialized in display advertising, notably in video ads. "Is this a move to monetize Youtube and make it profitable? Will there be an impact of any shares of IT companies I hold? Will Yahoo!'s share price drop again? Will News Corp.'s share price increase like in October?" These are legitimate and realistic questions an investor would ask himself when first receiving Bloomberg's news flash. Without any knowledge of the past, answering these questions is a challenging task. However, hints to wisely answer these questions could eventually be derived by analyzing historical data and by identifying similar situations that could have occurred to related companies. This would *"just"* require to analyze gigantic financial news archives and correlate the news information to thousands of companies' share price evolution: It is simply not humanly possible. This is typically the goal of a *Knowledge Discovery in Databases* (KDD) process: Help us cope with this data overload to identify valid, novel, potentially useful and ultimately understandable patterns (or knowledge in general) [FPSS96].

Domains such as medicine, the WWW, business or finance generate and store on a daily basis massive amounts of time-varying data. For instance, data warehouses keep swelling within the terabyte scale while the largest ones are expected to grow by an order of magnitude and reach the petabyte scale by 2012 [Win]. These colossal data sources represent valuable sources of insight for specialists to browse, analyze and discover golden nuggets of knowledge. More specifically, some applications of interest such as economics forecasting, sales forecasting, census analysis, web usage analysis or stock market analysis rely on discovering and extracting patterns from large collections of series of data points. It is commonly accepted that there exists some internal structure such as autocorrelation, trend or seasonal variation that could be accounted for in the knowledge discovery process. Traditionally, these analysis applications operate on series of mono- or multi-dimensional numerical data. For example, in stock market analysis, investors and trader focus on the daily, hourly or minutely variations of a companies' share prices to predict future trends.

However, the advent of database technologies in the early 70's and 80's [McG81] has enriched this landscape of applications and allowed the analysis of more complex forms of sequences, i.e., sequences of transactions where transactions are sets of items. Hence, the data unit is no longer constrained to be a numerical data object. In this new context, biologists could discover disease risk factors by analyzing patient histories [WRZ05], web content producers and marketing people are interested in profiling client behaviors [SCDT00] and traders investigate financial data for understanding global trends or anticipating market moves [ZZ04]. An example of such evolution is the *customer market basket analysis* paradigm proposed by Agrawal et al. in 1995 [AS95]. This novel form of analysis, known as *Sequential Pattern Mining (SPM)*, consists in discovering collections of items that are frequently purchased together by customers. Hence, sequential pattern mining relies on the analysis of sequences of customer transactions: Each customer is associated with a sequence of chronologically ordered transactions and each transaction is a set of items he purchased. A tangible example used by the authors to illustrate the knowledge discovered

by means of sequential pattern mining in such sequences is the following: "In a book store, 5% of customers bought "Foundation", then "Foundation and Empire", and then "Second Foundation"" within a certain period of time.

We highlight here the contrast between the two types of *series* of data that exist in the literature. On the one hand, "Time series" are sequences of chronologically ordered data objects, where each data object is a mono- or multi-dimensional numerical value. On the other hand, "Time sequences" are sequences of chronologically ordered data objects where each data object is a multi-dimensional categorical and/or numerical entity. More generically, "Time sequences" are also called "Time sequences of events" [PRSP+09]. Discovering knowledge in very large sources of time sequences of events, e.g., sales data warehouses or web usage logs, is a process-intensive task that requires fast and efficient algorithms. Unfortunately, recent studies on Web Usage Mining (WUM) [MTT04,RKS07] have shown that even state-of-the-art pattern mining algorithms hit a performance limit when the data mining parameters are too refined, i.e., when the support parameter drops approximatively below 0.06%. This context has motivated our work in designing a data transformation technique to create a concise, yet comprehensive and informative, representation of time-varying data that can support such categories of applications. We name this transformation: *Time sequence summarization*.

In order to support applications such as sequential pattern mining, time sequence summarization comes with a strong constraint inherent to time sequences: Events in time sequences are chronologically ordered and some data mining applications rely on this chronology to produce meaningful and useful knowledge. For instance, the sequence $\langle$*Lehman Brothers's Bankruptcy*, *Lehman Brothers's Rescue*$\rangle$ only makes sense because the events related to the *bankruptcy* needs to occur before the *rescue* could happen. Hence, time sequence summarization needs to account for the temporal ordering of the data.

During the past thirty years, data summarization techniques have been developed for various data sources such as text files, databases, data warehouses, data streams, etc.. The purpose of these technologies is to represent the input source in a more concise form for usages that include, but not limited to, storage, analysis, visualization or exploratory browsing. The approaches developed for storage purposes are more commonly called data *compression* algorithms and are mainly aimed towards file compaction. Data compression relies on the syntactic, e.g., byte, character or word redundancies, or statistic properties of the input file to reduce its representation. However, the compressed file is structurally different from its original form and thus can not be directly operated on without prior *decompression*. This form of summarization does not fit the context of time sequence summarization and will not be considered in our work.

The alternative idea is to take advantage of the semantic content of the data to perform summarization. During the past decade, semantic data summarization has been addressed in various areas such as databases, data warehouses or datastreams [HCCC92, HF96, JMN99, BGR01, JNOT04, SPRM05, PMR06]. These approaches output summaries that have been used for various purposes such as flexible query or exploratory browsing. However, accounting for the *time* dimension in the summarization process is an additional constraint that requires the chronology of events in the sequence to somehow be preserved. Much effort has been put into designing summarization techniques that use the semantic content of the data to create more concise representations. The continuous and exponential growth of content-rich sequential data feeds, e.g., in the form of RSS news feeds or even Twitter feeds, has triggered the need to somehow manage and make use of the temporal nature of events in sequences. To the best of our knowledge, small interest has been given

to temporal information for the purpose of summarizing. Under the light of applications that heavily rely on the chronology of events in a sequence, facing this new constraint comes with numerous challenges:

- The data generated is complex, i.e., contains both unstructured information (free text) and structured information (attribute-value pairs).
- The volume of the data keeps increasing.
- Applications require low complexity solutions.

In this thesis work, we address the challenges inherent to supporting chronology dependent and process intensive applications. For this purpose we make the following contributions:

- We formally define the concept of "Time sequence summarization" and thoroughly study related work.
- We propose a used-oriented technique to produce a time sequence summary.
- We reformulate the time sequence summarization as a novel conceptual clustering problem where a criterion function that considers both the content and temporal information of data needs to be optimized. We propose a naive solution and two greedy methods to solve this problem.
- We propose to study how well a time sequence summary can support in practice, on real world data, a specific chronology dependent applications: Sequential Pattern Mining. For this purpose, we thoroughly study the relationships that link sequential patterns discovered in original sequences to patterns discovered on summaries. We detail the analysis of these relationships.
- We propose an novel methodology to use patterns discovered in summaries called *Exploratory mining* to discover even more specific knowledge.
- We support all our contributions thanks to extensive sets of experiments on real world data extracted from Reuters 2003 financial news archives.

This dissertation is organized as follows.

We define in Chapter 1 the concept of "Time sequence summarization" to delimit the scope of our study. Then, we discuss the state-of-the-art methods that have been proposed to summarize time-varying data according to our definition. Also, we formalize the problem of building time sequence summaries. We present in Chapter 2 a user-oriented technique to build a time sequence summary, called TSaR. TSaR builds on top of the ideas of the *Generalize and Merge* and uses background knowledge in the form of domain specific taxonomies to represent events at higher levels of abstraction. Summarization is achieved by gathering *similar* events that occur *close* on the timeline and each step is parameterized by a user-defined parameter. In Chapter 3, we propose to address the issue of having to parameterize the summarization algorithm. Hence, we reformulate the time sequence summarization activity into a novel conceptual clustering problem where events are clustered based on the *similarity* of their content and their *proximity* on the timeline. We introduce the parameter-free property and switch the summarization activity to an optimization problem. We propose a naive solution, called N-TSS, and two greedy solutions, called G-BUSS and GRASS, to solve this new problem formulation. We present and thoroughly study in Chapter 4 an application framework to show how time sequence summarization can benefit in practice, on real world data, a specific chronology dependent application, namely Sequential Pattern Mining.

Chapter 1

Time sequence summarization: State of the art

1 Introduction

The analysis of series of data, or time-varying data in general, has long been the interest of specialists for applications as various as signal processing, economics forecasting, sales forecasting, census analysis, census analysis or stock market analysis. The belief is the existence of some internal structure such as autocorrelation, trend or seasonal variation that could be accounted for while forecasting or monitoring. Traditionally these applications rely on the analysis of a series of one dimensional numerical data. For example, in stock market analysis, trader focus on the daily, hourly or minutely variations of a company's stock value to predict future trends and consolidate or create new business opportunities.

However, the advent of database technologies in the early 80's [McG81] has enriched this landscape of applications and allowed the analysis of more complex forms of data such as *sequences of transactions*. For example, Agrawal et al. [AS95] proposed to mine customer transaction databases for discovering frequent consumer patterns. This novel form of analysis consists in finding sequential patterns, i.e., series of sets of items frequently purchased together by customers at different points in time. A tangible example used by the authors to illustrate the knowledge discovered by means of sequential pattern mining in such sequences is the following: "In a book store, 5% of customers bought "Foundation", then "Foundation and Empire", and then "Second Foundation"" within a certain period of time.

Through these two categories of applications, we highlight the contrast between the two types of *series* that are considered: (i) "Time series" and (ii) "Time sequences". These two types of time-varying data are commonly misdefined and used one for the other. On the one hand, a "Time series" is a sequence of mono- or multi-dimensional numerical data points. On the other hand, a "Time sequence" is sequence of mono- or multi-dimensional categorical and/or numerical data objects. In this thesis, we focus our work on the second type of series, i.e., on time sequences.

In a similar way, the concept of *summarization* has often been misunderstood with the concept of *compression*. The effect of both concepts is the same, i.e., reduce a piece of information into a more concise piece of information. However, the purpose of each method differs. *Compression*, also known as *syntactic compression*, uses the structural information of data to reduce its representation. Syntactic compression techniques use statistical or dictionary-based methods, e.g., Lempel-Ziv [ZL77]. These methods are *syntactic* since they consider data as a large string of bytes and operate at the byte level.

In contrast, other methods [JMN99, BGR01] consider the semantics of the data to achieve compression. These methods are more commonly called *semantic compression* techniques. The general idea in these approaches is to derive a descriptive model from the data. This model is built in a way that considers the semantic content of the data. The compressed version of the data is more concise but remains comprehensive without the need for *desummarization*. From this point of view, we believe that the concept of *summarization* is closer to semantic compression.

"Time sequence summarization" is an activity that is defined by the two ambiguous notions "time sequence" and "summarization". For this reason, we start this chapter by giving a definitive definition to "Time sequence summarization". By doing so, we are able to delimit the scope of this research work and present with precision and in details the state of the art that lies within this scope. Then, we will formally define the problem of building time sequence summaries.

Organization of the chapter

The remaining of this chapter is organized as follows. In Section 2, we attempt to disambiguate the concepts of "Time sequence" and of "Summarization" to precisely delimit related work and position the contributions presented in this thesis. We discuss in Section 3 work related to summarization in domains that consider time-varying data, i.e., (i) customer transaction databases, (ii) relational databases, (iii) temporal databases, (iv) data streams and (v) event sequences. Section 4 presents our definition of "Time sequence summarization" and we conclude this chapter in Section 5.

2 "Time sequence" and "Summarization"

"Time sequence" and "summarization" are two terms that are ambiguous as well in English common knowledge as in Computer Science research literature. This observation requires us to find or give a definitive definition to both terms to precisely delimit the scope of our study. For this purpose, we explore in this section the definitions of the two terms that have been considered in the literature.

2.1 Definitions of a "Time sequence"

The expression "Time sequence" has endorsed various meanings in Computer Science research during the past two decades. In the quest of finding a definition for the expression "Time sequence", one is tempted to refer to well established sources of common knowledge such as Merriam-Webster's Dictionary online[1], Dictionary.com[2] or the universal encyclopedia Wikipedia[3]. Here are some of the definitions given:

Webster's: First attempt, first pitfall. The expression "time sequence" is not referenced, however, the term "sequence" is defined as follows:

> " **Main Entry:** sequence
> **Pronunciation:** 'sē-kwən(t)s, -ˌkwen(t)s
> **Function:** noun
> **Etymology:** Middle English, from Anglo-French, from Medieval Latin sequentia, from Late Latin, sequel, literally, act of following, from Latin *sequent-*,

[1] http://www.merriam-webster.com/dictionary
[2] http://dictionary.reference.com
[3] www.wikipedia.org

sequens, present participle of *sequi*
Date: 14th century
[...] **2:** a continuous or connected series: as [...] **d**: a set of elements ordered so that they can be labeled with positive integers"

Dictionary.com: Second attempt, second pitfall. Similar to Webster's dictionary, the expression "Time sequence" is not referenced and the term to follow is "Sequence".

" **se·quence** [**see**-kw*uh* ns]
-noun
[...] *10.* Mathematics. a set whose elements have an order similar to that of the positive integers; a map from the positive integers to a given set. [...]
Origin: 1350–1400; ME < LL *sequentia*, equiv. to *sequ-* (s. of *sequi* to follow) + *-entia* -ENCE
Synonyms: 1. See SERIES. 2. arrangement. 4. outcome, sequel."

The synonym "series" leads to the following definition:

" **se·ries** [**seer**-eez]
-noun
1. a group or a number of related or similar things, events, etc., arranged or occurring in temporal, spatial, or other order or succession; sequence."

Wikipedia: The user is automatically redirected to the definition of a "Time series":

" In statistics, signal processing and financial mathematics, a time series is a sequence of data points, measured typically at successive times spaced at uniform time intervals. Examples of time series are the daily closing value of the Dow Jones index or the annual flow volume of the Nile River at Aswan. [...] Time series data have a natural temporal ordering."

The first observation one can make from these definitions is that the expression "Time sequence" is known but not necessarily well defined. For instance, Wikipedia needs to redirect the user to the definition of a "Time series". Still, there exists a consensus between Merriam-Websters dictionary and Dictionary.com, i.e., a time sequence is described as a set of *objects* ordered or mapped by/to positive integer values. However, these definitions remain unclear on the exact nature of the objects and their ordering criterion. We believe that the nature of the objects considered needs to be well defined since categorical data can not be processed the same way as numerical data. For instance, data clustering techniques rely on the continuity and the total order that exist in numerical domains. Thanks to this property, traditional data clustering methods can define distances and aggregation functions, e.g., min, max, average, etc., to compare objects. However, these metrics are not or are poorly applicable to categorical domains since there does not exist a total order on categorical data.

Literature in Computer Science has also given conflicting definitions. For instance, Lin et al. [LRSB96, LR98] or Korn et al. [KJF97] define a time sequence as a series of data objects associated to a timestamp and ordered by ascending timestamp. These objects can be 1-dimensional or multi-dimensional numerical values.

Meanwhile, Agrawal and Srikant introduced in 1995 the concept of Sequential Pattern Mining [AS95] for analyzing customer transactions which is later known as the *customer market basket analysis* paradigm. The authors' purpose is to discover frequent sequences of itemsets purchased together by customers. In this scenario, the data mined is a collection of sequences of customer transactions, where each transaction consists of: (i) a customer id,

(ii) a transaction time, or timestamp, and (iii) an itemset, i.e., the set of items purchased by the customer. The authors simply refer to the input data as a *sequence* or a *data sequence*.

In 1997, Mannila et al. [MTV97] introduce the concept of *Event Sequence* for the task of *frequent episode mining*. Frequent episode mining is an analysis task that aims at finding partially ordered collections of events that occur frequently together. The authors define an *event sequence* as a sequence or series of events where each event has an associated time of occurrence. An event is a pair (A, t) where A is an event taken from E a set of predefined event types and where t is an integer, the occurrence time of the event. Eventually, event types can be defined on several attributes, but the authors limit their study and consider and event type as a single value. Events in the sequence are organized by increasing time of occurrence t.

In a nutshell, Mannila et al. consider series of *objects* taken from a predefined set of event types and these objects are ordered by ascending time value. On the other hand, Agrawal and Srikant consider sequences of transactions. In other words, Agrawal and Srikant describe these *objects* as a collection of items. The important feature in this definition is the fact items are taken from categorical domains.

2.2 Taxonomy of "Time sequence" considered

In this thesis work, we look at the concept of "Time sequence" under the angle presented by Agrawal et al. and by Mannila et al.. We consider time sequences in their most general form, i.e., a time sequence is a series of objects, also called *events*, to which is associated a time of occurrence that is materialized as a timestamp. The timestamp is used to organize events in the sequence by ascending order on the timeline. Each event is described by a set of values, more generally called *descriptors*, that can be of numerical and/or categorical nature. Descriptors can be taken from one or more attributes, also called *descriptive domains*. Eventually, several descriptors can be taken from one same descriptive domain.

This general definition of a time sequence allows us to abstract various representations of time sequences or of time-varying data models in the literature. Indeed, time sequences are not new concepts and, as mentioned previously, have been used in different applications under different denominations. Table 1.1 gives three examples of conventional data sources that can be understood as time sequences in the sense of our work, namely, (i) customer transaction databases, (ii) relational databases and (iii) event sequences. For instance, the weekly record of a stock price is commonly known as a "Time series" which can also be understood as a particular instance of time sequence, i.e., a time sequence where each event is a mono-dimensional or multi-dimensional numerical value. Time series analysis is a well studied area [Ham94, Hei99, SZ04, Bri08] and lies out of the scope of our study. In this thesis work, we do not consider time series but restrict our focus to time sequences where events are characterized by categorical descriptors.

2.3 Definitions of "Summarization"

The term "Summarization" has multiple senses in English common language and also refers to different activities in Computer Science research. The previous sources of knowledge used to define the expression "Time sequence" give the following definitions for "Summarization":

Webster's:

" **Main Entry:** sum·ma·ri·za·tion
Pronunciation: ˌsə-mə-rə-ˈzā-shən, ˌsəm-rə-
Function: noun

<table>
<tr><th>Dataset type</th><th>Example</th></tr>
<tr><td>Transaction database</td><td>
<table>
<tr><th>TID</th><th>Itemset</th></tr>
<tr><td>T_1</td><td>{Bread, Milk}</td></tr>
<tr><td>T_2</td><td>{Beer, Bread, Diapers, Eggs}</td></tr>
<tr><td>T_3</td><td>{Beer, Cola, Diapers, Milk}</td></tr>
<tr><td>T_4</td><td>{Beer, Bread, Diapers, Milk}</td></tr>
<tr><td>T_5</td><td>{Bread, Cola, Diapers, Milk}</td></tr>
</table>
</td></tr>
<tr><td>Relational database</td><td>
<table>
<tr><th>CID</th><th>Timestamp</th><th>Att. 1</th><th>Att. 2</th><th>Att. 3</th><th>Att. 4</th></tr>
<tr><td>c_1</td><td>t_1</td><td>Bread</td><td>Milk</td><td></td><td></td></tr>
<tr><td>c_1</td><td>t_2</td><td>Beer</td><td>Bread</td><td>Diapers</td><td>Eggs</td></tr>
<tr><td>c_1</td><td>t_3</td><td>Beer</td><td>Cola</td><td>Diapers</td><td>Milk</td></tr>
<tr><td>c_1</td><td>t_4</td><td>Beer</td><td>Bread</td><td>Diapers</td><td>Milk</td></tr>
<tr><td>c_1</td><td>t_5</td><td>Bread</td><td>Cola</td><td>Diapers</td><td>Milk</td></tr>
</table>
</td></tr>
<tr><td>Event sequence</td><td>
<table>
<tr><td>Time</td><td>t_1</td><td>t_2</td><td>t_3</td><td>t_4</td><td>t_5</td><td>t_6</td></tr>
<tr><td></td><td>A</td><td></td><td>B</td><td>A</td><td>D</td><td>B</td></tr>
<tr><td>Events</td><td>C</td><td></td><td>D</td><td>C</td><td>E</td><td>D</td></tr>
<tr><td></td><td>F</td><td></td><td>F</td><td></td><td></td><td></td></tr>
</table>
</td></tr>
</table>

Table 1.1: Examples of time sequences

Date: 1865
[...] **2 :** SUMMARY"

Following up the term "Summary" gives us the following definition:

" **Main Entry:** sum·ma·ry
Pronunciation: ˈsə-mə-rē *also* ˈsəm-rē
Function: noun
Date: 1509
1: an abstract, abridgment, or compendium especially of a preceding discourse"

Dictionary.com:

" **sum·ma·rize** [***suhm****-uh*-rahyz]
-verb (used with object), -rized, -rizůing
1. to make a summary of; state or express in a concise form[...]"

Similarly, we follow up the definition of a "Summary":

" **sum·ma·ry** [***suhm****-uh*-ree]
-noun
1. a comprehensive and usually brief abstract, recapitulation, or compendium of previously stated facts or statements.
[...] **Synonyms:**
1. outline, précis. SUMMARY, BRIEF, DIGEST, SYNOPSIS are terms for a short version of a longer work. A SUMMARY is a brief *statement* or *restatement* of

main points, esp. as a conclusion to a work: a summary of a chapter. A BRIEF is a detailed outline, by heads and subheads, of a discourse (usually legal) to be completed: a brief for an argument. A DIGEST is an abridgment of an article, book, etc., or an organized arrangement of material under heads and titles: a digest of a popular novel; a digest of Roman law. A SYNOPSIS is usually a compressed statement of the plot of a novel, play, etc."

The most remarkable feature in these two definitions is the fact a summary is supposed to be a concise *statement* or *restatement* of an original piece of information. This characteristic can be understood in computer science terminology as a concise *representation* of the primitive piece of information.

Interestingly, the term "Summarization" itself is not referenced by Wikipedia. In return, a list of articles assumingly related to the term "Summarization" is proposed. Among the articles proposed, "Automatic summarization" and "Multi-document summarization" are the main computer science-related propositions. In this case, summarization is defined as the creation of a shortened but comprehensive version of one or multiple text documents. In fact, this definition positions the summarization activity as a subproblem of Natural Language Processing (NLP) research. Note that the definitions given by Merriam-Webster and Dictionary.com also seem to point the summarization task into the direction of NLP research.

In the Core Database Technology research area, data summarization can be understood in at least two different manners:

1. It is a mechanism to represent very large datasets of primitive data by only the relevant and meaningful information. The purpose of this mechanism is for instance to facilitate visualization, data exploration or decision making. Hence, the summarization mechanism is responsible for generating a more concise and general but comprehensive representation of messy primitive data.

2. Summarization appears as a *compression* tool to support other processing intensive applications by reducing the volume of the data. Examples of such applications include querying or data mining tasks such as clustering [BKKS01]. This reduction effect is achieved by grouping redundant primitive data and representing primitive data by suitable representative data objects.

In the later understanding of summarization, however, the *compression* task should not be mistaken with data compression as defined in Information Theory. Indeed, the semantics of compression in Information Theory is the process of encoding a piece of information using less bits than the unencoded version of the same piece of information. Compression is usually achieved by leveraging the structural properties of the data, e.g., using byte-wise redundancies. This compression scenario is mainly aimed at reducing the size of the data for storage or network transmission purposes. This form of compression is not the objective of our work and out of the scope of this thesis.

The few definitions presented here clearly highlight the inherent ambiguity that surrounds the summarization activity. This ambiguity takes it roots from the research area from which summarization is considered: The summarization activity in the area of Natural Language Processing, of Information Theory or of Core Database Technologies do not have the same needs nor objectives. However, even though the nature of the data and the purposes attributed to summarization differ in these examples, there still exist common trends that should be noted. The summarization task is meant to: (i) reduce the volume of the primitive data and (ii) represent the primitive data in a more abstract but comprehensive form.

2.4 Taxonomy of "Summarization" considered

In this thesis, we position our work closer to ideas of the Core Database Technology research area. In this perspective, our understanding of the summarization task is to build a new representation of a primitive dataset. This representation can then be used to support other possibly complex and process-intensive applications. More specifically, we aim at supporting *chronology dependent* applications, i.e., applications that rely on the chronological order of the data to be meaningful. An example of chronology dependent applications in the data mining is Sequential Pattern Mining [AS95]. Another example of chronology dependent application is Google Finance [Goo]. Google Finance is a visualization application developed by Google. In a user-defined timeline, Google Finance provides analysts with a tool to browse through companies' stock prices while visualizing background information about the companies. This background information is provided in the form of a series of chronologically ordered news events that occured at some interesting moments, e.g., during price jumps.

Therefore, in this thesis work, we consider summarization as the activity of transforming a collection of primitive data objects into a more concise and abstract, yet comprehensive and informative, representative object. This representative object can either (i) exist in the collections of primitive data objects or (ii) be a fictive object generated from the underlying data. Note that this task can also be understood as a form of *concept formation* in *Conceptual Clustering* [MS80] research.

3 Summarization in ...

We have identified the summarization activity as the task of forming more concise and abstract, yet comprehensive and informative, data objects from collections of primitive data objects. In fact, summarization has been widely used in various domains for different data models: customer transaction databases, relational databases, data warehouses, temporal databases, data streams, etc.. Due to the absence of a unifying definition and due to the diversity of tasks operated on summaries, each domain has proposed specific summarization solutions. In this section, we explore and discuss the state of the art of research works in these domains that relate to the activity of summarizing time-varying data.

3.1 Customer transaction database

Agrawal et al. introduced in the early 90's [AIS93] the idea of *Frequent Itemset Mining* (FIM) in customer transaction databases [AIS93, MTV94, AS94]. Progress in bar-code technology made it possible to store the *basket* data, i.e., all items purchased by a customer, on a per-transaction basis. Thus, analysts were eager to discover association rules to create new business opportunities. For instance, a piece of knowledge that is typically extracted by frequent itemset mining approaches is: "Men that buy baby nappies (diapers) on Thursdays or Saturdays also by beer". This piece of knowledge can then be exploited in various ways to increase revenues, e.g., position beer closer to diapers or make special package deals on beer on Thursdays.

In this context, a customer market basket database, or more generally, *a customer transaction database (TDB)*, is a collection of customer transactions. In this thesis, we will equivalently use the term *(customer) transaction database* to refer to a customer market basket database. Each customer is associated to a table (or a collection of rows) where each row corresponds to a purchase transaction performed by the customer. Each transaction contains a unique identifier TID and a set of items, called *itemset*, bought by the customer.

Transactions are chronologically ordered by the transaction ID or eventually thanks to a timestamp.

Formally, as defined by Agrawal et al. in [AIS93], let $\mathcal{I} = I_1, \ldots, I_m$ be a set of binary attributes called *items*. *TDB* is a database of transactions. Each transaction t in *TDB* is represented as a binary vector with $t[k] = 1$ if transaction t bought item I_k, and $t[k] = 0$ otherwise. There is one tuple in the database *TDB* for each transaction t. An example of customer transaction database is given in Table 1.2.

TID	**Beer**	**Bread**	**Cola**	**Diapers**	**Eggs**	**Milk**
T_1	1	0	0	0	0	1
T_2	1	1	0	1	1	0
T_3	1	0	1	1	0	1
T_4	1	1	0	1	0	1
T_5	0	1	1	1	0	1

Table 1.2: Example of customer transaction database

On should note that this representation of transaction databases can also be understood as a time sequences of events. Indeed, in a *TDB*, transactions are ordered by ascending transaction TID and the set of attribute-value pairs of each tuple, e.g., {beer=1, bread=0, cola=0, diapers=0, eggs=0, milk=1}, can be considered as the descriptors of the event. It suffices to map a transaction itemset to a set of descriptors where descriptors are items for which the binary vector $t[k]$ equals 1. Hence, the example of Agrawal et al.'s transaction database given in Table 1.2 is equivalent to a time sequence of events as given in Table 1.1.

Thanks to the advent of bar-code technologies, the size of customer transaction databases has dramatically increased and risen new challenges for handling these data sources. These challenges have attracted many researchers' interest in creating more compact representations of these data sources. Hence, a bulk of work [CK05, WA05, WK06, XJFD08] has focused on creating *summaries* for representing customer transaction databases in a more concise but informative form. We discuss in the following paragraphs the state of the art of summarization methods designed for customer transaction databases.

Chandola et al. [CK05]

Chandola et al. propose in [CK05] two summarization algorithms for sets of transactions defined on categorical attributes. In this work, the authors formulate summarization as an optimization problem that involves two objective functions: (i) compaction gain and (ii) information loss. The authors view summarization as the activity of *compacting* a set of transactions into a smaller set of transactions, i.e., a set having a smaller number of transactions, while retaining the maximum possible information.

For this purpose, Chandola et al. consider as input a set of transactions T, with $|T| = m$ transactions, where each transaction T_i is defined on a set of n categorical features $F = \{F_1, F_2, \ldots, F_n\}$. The set F is associated to a set of weights W such that the weight $W_i \in W$ corresponds to the weight of feature $F_i \in F$. A summary S of the set of transactions T is: A set of individual summaries $\{S_1, S_2, \ldots, S_l\}$ where (i) each S_j represents a subset of transactions in T and (ii) each transaction $T_i \in T$ is represented by at least one $S_j \in S$. Hence, each summary $S_j \in S$ is a representation of the subset of the transactions covered and S_j is defined as the feature-wise intersection of all transactions covered, i.e., $S_j = \bigcap_{i=1}^{k} T_i$.

The measures introduced by the authors for evaluating the quality of a summary are based on the (i) compaction gain of the summarization algorithm and (ii) the information

loss. The compaction gain is measure by the ratio $\frac{|T|}{|S|}$ and the information loss of a transaction T_i represented by an individual summary S_k is the weighted sum of the number of features in T_i that are not covered by S_k, i.e., $loss_{ik} = \sum_{q=1}^{n} W_q \times b_q$ where $b_q = 1$ if $T_{iq} \notin S_k$ and 0 otherwise.

The first summarization method proposed uses any conventional clustering algorithm to generate clusters from T then represents each cluster by a representative summary using the feature-wise intersection operator. This technique works well in general but has poor performances, in terms of information loss, when dealing with outliers.

The second summarization method proposed is a Bottom-Up Approach to Summarization (BUS) that uses Frequent Itemset Mining (FIM) in a first step to determine all closed frequent itemsets of T, denoted C_c. Then, at each iteration a *best* candidate is chosen from C_c, i.e., the candidate that reduces the size of the data and incurs the less information loss, and all transactions covered by this candidate are summarized together. The process is repeated until a desired compaction gain is achieved.

Let us give an example of summary produced by Chandola et al.. We adapt the transaction database time sequence given in Table 1.1 to the authors' input format and take the categorical features for characterizing each transaction in the set {Alcohol, Baby, Diary, Food, Soft drink, Pastry}. Each feature has equal weight. Hence, Table 1.3 gives the input set of transactions T. One possible summary is given in Table 1.4. In this example, summary S_1 groups transaction T_1, S_2 groups transactions T_3 and T_4 and S_3 groups transactions T_2 and T_5. The quality of the summary is as follows:

- Compaction gain: $\frac{3}{5}$
- Information loss: $0 + (\frac{2}{6} + \frac{2}{6}) + (\frac{4}{6} + \frac{4}{6}) = 2$

TID	**Alcohol**	**Baby**	**Cooked food**	**Diary**	**Raw food**	**Soft drink**
T_1	N/A	N/A	Bread	Milk	N/A	N/A
T_2	Beer	Diapers	Bread	N/A	Eggs	N/A
T_3	Beer	Diapers	N/A	Milk	N/A	Cola
T_4	Beer	Diapers	Bread	Milk	N/A	N/A
T_5	N/A	Diapers	Bread	Milk	N/A	Cola

Table 1.3: Example T in Chandola et al.'s format

S	**Alcohol**	**Baby**	**Cooked food**	**Diary**	**Raw food**	**Soft drink**
S_1	N/A	N/A	Bread	Milk	N/A	N/A
S_2	Beer	Diapers	***	Milk	N/A	***
S_3	***	Diapers	Bread	***	***	***

Table 1.4: Example of Chandola et al.'s summary

This technique generates compact and comprehensive summaries. However, the authors do not take into account nor use the inherent sequentiality or ordering of transactions in T for summarizing transactions. For instance, it is not explicit with S, if transactions in summary S_2 occur in T before or after transactions in summary S_3. This information is most important for chronology dependent applications. Hence, this technique still lacks a methodology to explicitly reflect the sequentiality of original transactions.

SUMMARY [WK06]

Wang and Karypis [WK06] propose the SUMMARY method to generate summaries from transaction databases represented as in Table 1.1. The authors present a technique based

on *Frequent Closed Itemset Mining* [PBTL99]. The idea is to mine, given a desired minimum support min_sup, the set of all frequent closed itemsets in the input transaction database TDB. Once these itemsets are identified, for each transaction T_i in TDB, any one longest frequent itemset is chosen to represent, or summarize, T_i. This itemset is called the *summary itemset* of T_i and the set of all summary itemsets of transactions is TDB is called a *summary set*.

In practice, the list of frequent items, i.e., items that have minimum support min_sup, in ascending support order is generated and is denoted $f_$list. The list of frequent items in each transaction is sorted according to $f_$list. Let us give an example of summary obtained thanks to SUMMARY. Suppose $min\_sup = 2$, the $f_$list obtained from transaction database in Table 1.1 is $f_$list={Cola:2, Beer:3, Bread:4, Diapers:4, Milk:4}. Table 1.5 gives the ordered frequent itemset list in column 3. From Table 1.5, one summary set w.r.t. the input TDB is {{Beer, Bread, Diapers}:2, {Cola, Diapers, Milk}:2}.

TID	**Items**	**Ordered frequent itemset list**
T_1	{Bread, Milk}	{Bread, Milk}
T_2	{Beer, Bread, Diapers, Eggs}	{Beer, Bread, Diapers}
T_3	{Beer, Cola, Diapers, Milk}	{Cola, Diapers, Milk}
T_4	{Beer, Bread, Diapers, Milk}	{Beer, Bread, Diapers}
T_5	{Bread, Cola, Diapers, Milk}	{Cola, Diapers, Milk}

Table 1.5: Example of summary set, $min\_sup = 2$

The different SUMMARY algorithms proposed by the authors have very interesting performances. The algorithms run very fast even when the selected minimum support is extremely low, i.e., $0,2\% \leq min\_sup \leq 1\%$. However, the summary produced still suffer from at least two shortcomings. First, the summary set produced does not cover all transactions in the TDB. Since the process is guided by a minimum support parameter, outliers will not be selected by the frequent itemset mining algorithm and will not have a representative summary itemset. This is highly undesirable in applications where outliers could be of great significance to analysts. Second, the approach does not explicitly express the trade-off compaction gain for the loss of items in the summary. Again, since items present in the summary are controlled by a minimum support parameter, gaining in data compaction requires to increase the minimum support. By doing so, more transactions will become infrequent (in the sense of the support), and will be completely lost.

HYPER [XJFD08]

Xiang et al. propose to summarize transaction databases thanks to the HYPER [XJFD08] method. The authors assume the input transaction database TDB is represented as a binary matrix such that cell (i, j) has value 1 if transaction i contains item j, 0 otherwise. This representation is exactly the one used in Table 1.2. In [XJFD08], a summary is expressed using the *hyperrectangle* notation where a hyperrectangle H is a Cartesian product of (i) a set of transactions T and (ii) a set of items I, i.e., $H = T \times I = \{(i, j), i \in T \text{ and } j \in I\}$. The cost associated to representing hyperrectangle H is the sum $|T| + |I|$. If a set of hyperrectangles, denoted CDB, covers all the cells of the transaction database TDB, CDB is said to be the *covering database* or *summarization* of TDB. Let us give an example of CDB that covers the transaction database in Table 1.2. Table 1.2 can be covered by six hyperrectangles as defined in Table 1.6.

The summarization problem is formalized as the problem of finding CDB while minimizing the total representation cost, i.e., the sum of representation cost of all hyperrectangles in CDB. The authors prove the problem formulation is a $\mathcal{NP}$-difficult and much related

Hyperrectangle	Transactions	Items
H_1	$\{T_1, T_4\}$	{Beer, Milk}
H_2	$\{T_3, T_5\}$	{Cola, Diapers, Milk}
H_3	$\{T_2, T_4\}$	{Bread, Diapers}
H_4	$\{T_2, T_3\}$	{Beer}
H_5	$\{T_2\}$	{Eggs}
H_5	$\{T_5\}$	{Bread}

Table 1.6: *CDB* of transaction database in Table 1.2

to the Weighted Set Covering problem. The authors propose a solution called HYPER. HYPER relies on the use of frequent itemset mining to handle hyperrectangles having same itemsets as a single group. HYPER finds a solution identical to the solution of the greedy approach for the weighted set covering problem and has an approximation ratio of $\ln(k) + 1$, with k the number of hyperrectangles produced. HYPER has polynomial computational time that depends on:

- $|F_\alpha|$: the number of frequent itemsets having minimum support α.
- $|\mathcal{I}|$: the number of all items in the transaction database TDB.
- $|\mathcal{T}|$: the number of transactions in TDB.

In total, the computation complexity of HYPER is $O(k|\mathcal{T}|(|\mathcal{I}|+log|\mathcal{T}|)(|F_\alpha|+|\mathcal{I}|))$. Note that the authors do not consider the computation cost of the Apriori-based algorithm used to compute all frequent itemsets. In practice, depending on the dataset used, this computation cost can be prohibitive. For instance, on the UCI Machine Learning *chess* dataset ($\mathcal{I} = 75$ and $\mathcal{T} = 3196$), the summarization run time is approximatively 10^4s when $\alpha = 25\%$, 3.5×10^4s when $\alpha = 20\%$ and 8×10^4s when $\alpha = 15\%$.

Since the hyperrectangles in *CDB* are formed from frequent itemsets, the summary produced gives an interesting high level understanding of the transaction database. However, this representation can not be directly reused and piped to other chronology dependent application and requires some form of *desummarization* beforehand. Also, the hyperrectangles produced are not organized to reflect in any way the chronology of the underlying transactions.

Wan et al. [WA05]

Wan et al. [WA05] propose to support sequential pattern mining by representing the input transaction database TDB as a *Compact Transaction Database CTDB*. The author propose an Apriori-based algorithm, called *CT-Apriori*, that uses the specificities of this representation to improve the computational time of conventional Apriori-based algorithms. Compact transaction databases are build using a novel data structure called a *Compact Transaction Tree (CT-Tree)*. The purpose of the *CT-Tree* is to enumerate and represent transactions in a sequence of transactions in a compact manner and associate to each transaction its number of occurrences in the sequence. Therefore, items in a transaction are supposed to be organized in lexicographical order and the generation of a *CT-Tree* starts from a "ROOT" node. Every other node in this tree is compound of two parts: the name of the item and an occurrence count of the path that goes from "ROOT" to this node. Transactions are examined one at a time and the tree is grown from each transaction. For instance, Table 1.7(a) gives an example of transaction database and Figure 1.1 gives the CT-Tree generated from Table 1.7(a).

(a) Original TDB

TID	List of items
T_1	Bread,Milk
T_2	Beer,Bread,Diapers,Eggs
T_3	Beer,Cola,Diapers,Milk
T_4	Beer,Bread,Diapers,Eggs
T_5	Bread,Cola,Diapers,Milk

(b) CTDB

head						
Item	**Bread**	**Diapers**	**Beer**	**Milk**	**Cola**	**Eggs**
Count	4	4	3	3	2	2
body						
Count	**List of items**					
1	Beer,Milk					
2	Bread,Diapers,Beer,Eggs					
1	Diapers,Beers,Milk,Cola					
1	Brad,Diapers,Milk,Cola					

Table 1.7: Example of *Compact Transaction Database*

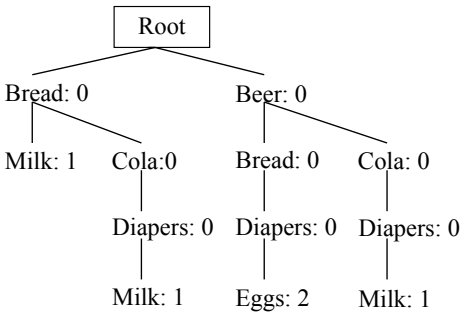


Figure 1.1: *CT-Tree* for transaction database in Table 1.7(a)

The compact transaction database of the input TDB is then generated from its corresponding *CT-Tree*. A $CTDB$ is compound of two parts: a *head* and a *body*. The head of the $CTDB$ is a list of couples (I_n, I_c) where I_n is an item and I_c is the frequency count of I_n in the transaction database TDB. All couples in the head are organized in frequency-descending order. The body of the $CTDB$ is a list of couples (T_c, T_n) where T_c is a unique transaction in TDB and T_n is the occurrence count of T_c in TDB. As an example, Table 1.7(b) gives the $CTDB$ that is built from *CT-Tree* in Figure 1.1.

This compact transaction database structure is specifically designed for supporting sequential pattern mining using an Apriori-based algorithm. Indeed, the head section of the $CTDB$ is designed so that Apriori-based algorithm do not need to scan the entire TDB to generate the set of frequent sequences of length 1. Note that this representation does not necessarily reduce the number of transactions in the $CTDB$. Indeed, if transaction T_4 in Table 1.7(b) was replaced by itemset {Beer,Bread,Diapers,Milk}, all transaction counts in the $CTDB$ would equal 1. No compaction gain would be achieved by building the $CTDB$. Also, the technique does not integrate any mechanism to manage the chronology of transaction in the $CTDB$, i.e., the transactions listed are not organization in a way that would allow to reproduce the chronology of transactions. Hence, this structure is only useful in the context of sequential pattern mining and more specifically to custom-designed Apriori-based algorithms.

3.2 Relational databases

Edgard F. Codd introduced in 1969 the *Relational model* [Cod69] for database management. Codd's idea was to propose a relational view of data and hence represent all data as a mathematical n-ary relation, i.e., as a subset of the Cartesian product of n domains or attributes. These attributes can as well be defined on numerical domains or categorical domains. The relational model relies on first-order logic and describes the data by means of its natural structure only. Queries are operated on the data by means of a *relational*

calculus or *relational algebra* performed on the model. The relational model was then extended with criteria, i.e., *normalization* or *normal forms*, for determining the model's degree of vulnerability to logical inconsistencies and anomalies, e.g., redundancies or functional depedencies on part of a candidate key in a relation. We refer the reader to Chris Date's reference book [Dat04] for an insight into more than thirty years of research on the relational model.

Since, the relational model has been the most popular data model for data representation and organization. Relational Database Management Systems (RDBMS) have become a multi-billion dollar business [Gar07] and are the most used systems worldwide to manage data. Note that the relational model does not directly support the time dimension of data and this time dimension is handled in a very naive way, i.e., as a numerical or categorical attribute as any other. This limitation is addressed in RDBMS by implementing extensions from temporal database research to handle in a more advanced way temporal information [TLHY03]. However, the size of data generated worldwide, captured, analyzed and stored by these systems keeps increasing at a crazy rate [Aki06,GCM+08,Cuk10]. This phenomenon has been foreseen by many researchers from the Core Database research community and various methods we designed to represent this data in a more concise form, yet informative and comprehensive, to allow its analysis. We discuss in this section the most important contributions that build summaries for relational databases.

Attribute Oriented Induction (AOI) [Cai91]

Attribute Oriented Induction (AOI) is an approach initially proposed by H. Cai, J. Han and N. Cercone [Cai91,HCCC92,HF96]. The process was then improved and implemented in DBLearn [HFH+94] then in DBMiner [HFW+96].

The original purpose of AOI is to process relational database tables for knowledge discovery. Knowledge discovery in a relation table $R(A_1, \ldots, A_n)$ is achieved by reducing the domain of attributes A_i, then reducing the number of tuples in R in a *Generalize and Merge* process. The authors assume background knowledge is available and/or provided by knowledge engineers and domains experts in the form of concept hierarchies organized as taxonomies. A taxonomy of a knowledge domain is a particular classification into a hierarchical structure of concepts defined in that domain. Concepts in a taxonomy, also called *descriptors* in this work, are linked by a generalization-specialization relationship, also known as a *IS-A* relationship, and are partially-ordered thanks to this generalization-specialization relationship. The root of the taxonomy is the most general concept, usually a reserved word denoted "any_Domain_name", and the leaves in the taxonomy are the most specific concepts. Figure 1.2 gives an example of taxonomy for the *location* domain. In this example, the root of the taxonomy is "any_Location".

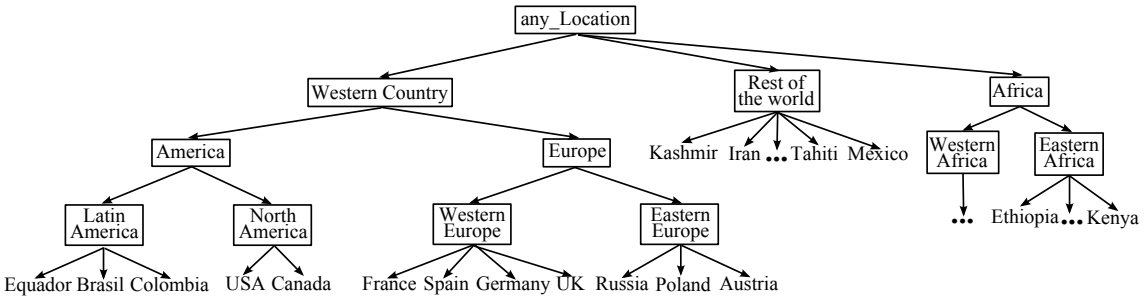


Figure 1.2: Example of taxonomy for the *location* domain

At each iteration, the AOI process chooses an attribute, denoted A_i, and all attribute values in R defined on A_i are generalized once using the taxonomy associated to A_i, denoted

H_{A_i}. A direct consequence of the generalization phase is the reduction of the variability of attribute values for the domain generalized. The output table is called a *generalized* table. In practice, an attribute value d is generalized by replacing d by its parent, denoted d', in taxonomy H_{A_i}. For instance, in our *location* domain taxonomy, the descriptor "France" is generalized into the descriptor "Western Europe".

Since generalization is responsible for reducing the variability of attribute domains, tuples in the generalized table could eventually become identical. Merging is then responsible for grouping tuples that have become identical into one single tuple. The number of tuples that has been grouped together this way is maintained in an extra attribute called *vote*. This generalization and merging process is repeated until one of the following conditions are met: (i) a generalization threshold for one/each attribute is met, (ii) a minimum number of generalization is achieved, (iii) a tuple reduction threshold is met, etc..

Table 1.8(a) gives an example of relational table on which we operate AOI. Relation *Import(Location, Good, Quantity)* gives the amount of goods imported from a given country. Table 1.8(b) gives the generalized version of Table 1.8(a) when the *location* attribute is chosen for generalization. Note that after generalization no tuples have become identical, thus, we reiterate and choose to generalize using the *quantity* attribute. Note that we assume the existence of a taxonomy that maps numerical values to concepts taken in the set {Low, Average, Above average, Important}. The output table is given in Table 1.8(c). We can now note that identical tuples have been generated thanks to this generalization step. These tuples are then merged together and their count maintained, the result is given in Table 1.8(d).

(a) Original table R

Location	**Good**	**Quantity**
Ecuador	Coffee	400
Brazil	Coffee	450
Colombia	Cocoa	300
Ethiopia	Coffee	420
Kashmir	Saffron	80
Kenya	Coffee	410
Iran	Saffron	90
Tahiti	Vanilla	210
Mexico	Vanilla	200

(b) Genealization on *location*

Location	**Good**	**Quantity**	**Vote**
Latin America	Coffee	400	1
Latin America	Coffee	450	1
Latin America	Cocoa	300	1
Eastern Africa	Coffee	420	1
World	Saffron	80	1
Eastern Africa	Coffee	410	1
World	Saffron	90	1
World	Vanilla	210	1
Latin America	Vanilla	200	1

(c) Genealization on *quantity*

Location	**Good**	**Quantity**	**Vote**
Latin America	Coffee	Important	1
Latin America	Coffee	Important	1
Latin America	Cocoa	Above average	1
Eastern Africa	Coffee	Important	1
World	Saffron	Low	1
Eastern Africa	Coffee	Important	1
World	Saffron	Low	1
World	Vanilla	Average	1
Latin America	Vanilla	Average	1

(d) Merged table

Location	**Good**	**Quantity**	**Vote**
Latin America	Coffee	Important	2
Latin America	Cocoa	Above average	1
World	Saffron	Low	2
Eastern Africa	Coffee	Important	2
World	Vanilla	Average	1
Latin America	Vanilla	Average	1

Table 1.8: Example of Attribute Oriented Induction

The authors propose seven basic strategies for operating attribute-oriented induction on relational databases. Among these strategies, the first one states that attributes to be chosen first are the ones with the smallest decomposable components. The purpose of this strategy is to avoid over-generalization. Despite this strategy, attributes can still be over-generalized by the induction process. In [HF96], Han et al. address this issue and improve the induction process by dynamically adapting the level of generalization of each

attribute.

This technique displays interesting domain reduction and tuple numerosity reduction capabilities: In our example, the variability of attribute values decreases from 22 values to 10 values and the number of tuples is reduced from 9 to 6. However, as the process is designed, when generalized tuples become identical, the authors do not propose any mechanism to group identical tuples in a manner that reflects their ordering or sequentiality. For instance, between Table 1.8(b) and Table 1.8(d), the merging step induces the following information lost: "An important amount of coffee was imported from Eastern Africa after an above average amount of cocoa was imported from Latin America (tuples 3 and 4 in Table 1.8(c))". This shortcoming can be troublesome for chronology dependent applications.

Fascicles [JMN99]

Jagadish et al. [JMN99] introduce the notion of *fascicles* to reduce the size of relational databases for the purpose of minimizing storage. Also, the authors propose a method for extracting patterns from the compressed databases produced. Fascicles rely on an extended form of association rules, more exactly, frequent itemset mining, to achieve data compression. The authors highlight the fact that, often, many tuples in a relation share similar values for several attributes. Therefore, the notion of fascicles introduced captures and groups such tuples. A fascicle is a subset of tuples F in a relation R for which there exists a set of k *compact* attributes A where the tuples in F have similar attributes values for attributes in A. An attribute A_i is said to be *compact* for all records in F if the range of A_i's values (for numerical attributes) or the number of distinct values (for categorical attributes) does not exceed a compactness tolerance parameter denoted t_{A_i}. For instance, in the *Import* relation in Table 1.7(a), suppose the compactness tolerance parameters on attributes are:

- $t_{location} = n$: Any attribute value is acceptable; We assume that $|location| = n$.
- $t_{good} = 1$: An exact match is required.
- $t_{quantity} = 50$: The range tolerance is 50, i.e., the maximum difference between attribute values must not exceed 100.

Under these tolerance parameters, an example of fascicle extracted from Table 1.8(a) is the set of tuples F containing tuples: (Ecuador, Coffee, 400), (Brazil, Coffee, 450), (Ethiopia, Coffee, 420) and (Kenya, Coffee, 410).

Storage minimization is achieved by projecting all compact attributes and storing their *representative* values only once separately. The representative value for a given attribute A_i is computed differently whether the attribute is categorical or numerical. In the case of categorical attributes, suppose tuples in F are described by attribute value d on attribute A_i. Since the tolerance parameter is often strict, i.e., $t_{A_i} = 1$, the representative value is trivial and chosen as the value d itself; In our example, for attribute *good*, the representative value chosen would be "coffee". In the case the tolerance parameter is relaxed, i.e., $t_{A_i} = n$, any attribute value can be chosen. In the case of numerical attributes, an aggregative function can be used to compute the representative value, e.g., the average value between the min and max attribute values. Thus, in our example, the representative value for the *quantity* attribute would be $average(410, 450) = 430$.

Jagadish et al. formulate the semantic compression problem as a storage minimization with fascicles problem. Since the authors allow fascicles to contain an arbitrary number of k compact attributes, the search space is considerably increased. The authors propose several algorithms with different parameters, i.e., fixed k compact attribute fascicles

(Single-k algorithm) or $\geq k$ compact attribute fascicles (Multiple-k algorithm), that allow the extraction of such fascicles. The ideas developed for Fascicles are mostly interesting for handling numerical values since defining an aggregate function for numerical values is intuitive. For instance, they can be chosen as the *min*, *max*, *average*, etc.. However, in the case of categorical attributes, even if the tolerance parameter can be relaxed, the authors enforce in practive a strict tolerance parameter, i.e., $t_{A_i} = 1$. This choice is due to the lack of a mechanism in AOI for representing a set of categorical attribute values by a representative value.

For application view point, fascicles are used to extract *frequent* patterns, i.e., in this context patterns are frequent itemsets. The notion of *frequency* is defined by the number of tuples grouped by a fascicle. The *quality* of patterns extracted is directly linked to the compactness factor k of attributes and the tolerance parameters for each attribute of the fascicles generated. Incidentally, the quality of patterns extracted is reflected in the organization of fascicles, thanks to a partial order defined on fascicles and the Smyth [Smy78] ordering for powers sets, by decreasing tolerance and increasing k value.

SPARTAN [BGR01]

SPARTAN [BGR01] is a method proposed to compress massive data tables using its semantics. The authors make the following observations from the Fascicles approach [JMN99]: Fascicles allows to operate lossy compression by grouping *approximatively* matching tuples, where the degree of approximation is specified by user-defined compactness parameters. From this perspective, fascicles allow to guarantee upper bounds on the compression error. However, fascicles proceed on a *tuple-wise* (or *row-wise*) manner to detect patterns, i.e., tuples having similar values on several attributes. This *row-wise* approach could be impossible to achieve when the data has very strong *attribute-wise* (or *column-wise*) dependencies. Therefore, the undercurrent idea in SPARTAN is to detect such column-wise dependencies and represent those dependencies in a concise and accurate predictive model, namely, in *Classification and Regression Trees (CaRT)*.

For this purpose, SPARTAN makes the same assumptions as Fascicles, i.e., a error tolerance parameter is defined for each attribute that the input data table is defined on. Hence, SPARTAN detects predictive attribute dependencies thanks to the use of a Bayesian network [Pea78] on the underlying set of attributes. The sets of predicted attributes selected are then used to generate *CaRT*-based predictors such that the overall storage cost is minimized, within the tolerance error range. The authors demonstrate that finding the *CaRT*s that minimize the overall storage cost maps to the *Weighted Maximum Independent Set* (WMIS) problem and is a $\mathcal{NP}$-hard problem. Hence, the generation of these *CaRT*s go through the costly generation, in terms of computational time and of memory usage, of an important number of *CaRT* predictors. The authors overturn this shortcoming thanks to the use several optimization techniques: (i) random sampling, (ii) leaves are only expanded if accurate predictions are not possible and (iii) pruning the growing tree by exploiting prescribed error tolerance for the predicted attribute.

For instance, in our *Import* relation given in Table 1.8(a), we can devise a simple classification tree for predicting the *good* attribute with the *quantity* attribute as predictor. The CaRT generated is given in Figure 1.3.

Even though SPARTAN outperforms Fascicles from compression view point, SPARTAN suffers from the same shortcomings as Fascicles for supporting chronology dependent applications. The approach was designed to maximize the reduction of data tables size for storage purpose only. Even though *CaRT*s give a high level understanding of attribute dependencies, applications such as mining can not use the data tables compressed with SPARTAN without a form of *decompression* beforehand. Also, in order to maximize the

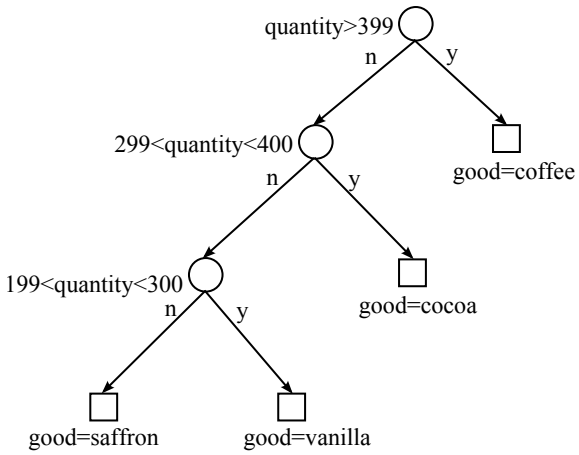


Figure 1.3: *CaRT* for predicting the *good* attribute with *quantity* as predictor

compression ratio of the algorithm, SPARTAN only focuses on attribute dependencies and gather tuples from anywhere in the table of generate the classification trees. As a consequence, applications that rely on the sequentiality of the tuples can not use the compressed table.

ItCompress [JNOT04]

The Iterative Compression, a.k.a. ItCompress, semantic compression approach was proposed by Jagadish et al. in [JNOT04]. The authors' basic underlying idea is to compress a data table R by reducing the number of tuples, or *rows*, in the table. This compression effect is obtained by representing *similar* rows by a representative row (RR). In practice, given an error tolerance parameter defined on each attribute, ItCompress finds a set of representative rows (RR) that matches at best the tuples in the original table within the error tolerance range. All representative rows are grouped in a RR table. The compressed table, denoted R_c, then contains three attributes, i.e., (i) *ERId*, (ii) *Bitmap* and (iii) *Outliers*. The *ERId* attribute matches rows in R with representative rows' ID in the RR table. Suppose row t_1 in R is represented by RR $ERId_1$. The *Bitmap* attribute expresses, for row t_1 in the original table R, which are the attributes values of $ERId_1$ that are not within error tolerance range.

Let us give an example of summary produced by ItCompress with our *Import* relation defined earlier in Table 1.9. Similar to our example on fascicles, we assume the error tolerance parameter for attributes *location*, *good* and *quantity* are 1, 1 and 50 respectively. Table 1.9(a) is the original table R, Table 1.9(b) is the compressed table R_c and Table 1.9(c) is the table of representative rows.

In a nutshell, the output summary is compound of: (i) the RR table which is characterized by a representative row ID and the attributes of the summarized table, (ii) the compressed table where each row corresponds to a row in the original table and where the attributes are (1) the RRid to which the tuple matches, (2) a bitmap expressing which are the outlying values, i.e., values for attributes that do not match within error tolerance, and (3) the set of outlying values. The main issue in ItCompress is to choose a good set of representative rows that maximizes the total coverage of representative rows. The authors show this problem is equivalent to the k-center problem [GJ79] and is therefore a $\mathcal{NP}$-hard problem.

ItCompress computes a solution in an iterative way. At the first iteration, the algorithm randomly chooses a set of k representative rows from the input table. Then, at

(a) Original *Import* relation R

Location	Good	Quantity
Ecuador	Coffee	400
Brazil	Coffee	450
Colombia	Cocoa	300
Ethiopia	Coffee	420
Kashmir	Saffron	80
Kenya	Coffee	410
Iran	Saffron	90
Tahiti	Vanilla	210
Mexico	Vanilla	200

(b) Compressed relation R_c

ERId	Bitmap	Outliers
1	111	
1	011	Brazil
2	111	
1	011	Ethiopia
3	111	
1	011	
3	011	Iran
4	111	
4	011	Tahiti

(c) Representative rows table RR

ERId	Location	Good	Quantity
1	Ecuador	Coffee	400
2	Colombia	Cocoa	300
3	Kashmir	Saffron	80
4	Tahiti	Vanilla	210

Table 1.9: Example of ItCompress summarization

each iteration, this random choice is improved with the objective of increasing the total coverage over the table to be compressed. Even though this approach does not guarantee optimality, the authors show that ItCompress gives good compression ratio without sacrificing efficiency. Note that the authors do not clearly express how the compression ratio is defined, however we assume compression ratio is defined as the ratio $1 - \frac{|RR|-1}{|R|-1}$, i.e., the ratio between the final number of representative rows and the number of tuples in the input table.

ItCompress has computational complexity in $O(kmnl + kdl)$ where n is the number of rows in the original table, k the number of rows in the RR table, m the number of columns, d the total number of domain values/intervals and l the number of iterations. This means that ItCompress is mostly linear in processing time with the number of rows to summarize. The experimental results presented by the authors show that the approach is still influenced by the number of iterations and sampling ratio under certain conditions. Thus, when dealing with high dimension datasets such as in a financial stock market environment, the number of RR, the sampling ratio and the number of iterations are parameters that might highly degrade computational time.

Considering these characteristics, ItCompress appears as a serious candidate for summarizing sequences of events, stored in a relation database. Indeed, the summary produced by ItCompress has the following desirable properties: (i) handling of numerical and categorical data, (ii) the ordering of tuples in the original table is preserved in the compressed table and (iii) the (theoretical) computational complexity is linear.

However the ItCompress approach still suffers from several shortcomings to support chronology dependent applications. The summary produced can not be directly piped to and used by such applications. Some form of *decompression* needs to be operated beforehand so that applications that usually perform on relation tables can be executed. Also, even though the compressed table produced is concise, it is not informative enough for a user. The bitmap representation used is a compact way to represent commonalities between a representative row and a row in the original table. However, much time and effort need to be spent to find the representative row and understand the representation.

This effort is even truer when very large datasets are compressed. In short, the approach lacks a mechanism to represent at a higher level of abstraction rows in the compressed table to make the table more informative.

SaintEtiQ [RM02, SPRM05]

The last technique we discuss here is the SaintEtiQ summarization approach proposed by R. Saint-Paul et al. in [RM02, SPRM05]. SaintEtiQ was not specifically designed to compress relational data for storage purposes, i.e., to reduce large amounts of data into a small package. The main purpose of SaintEtiQ is to produce a smaller and more concise view of very large databases. This approach takes database records as input and gives some kind of knowledge as output. The process is divided into two main steps: the first one consists in rewriting the input *raw* data and the second one is the learning step. The rewriting step relies on Zadeh's fuzzy set theory [Zad65] to transform the input raw tuples into *candidate* tuples. This rewriting step is achieved in accordance with Background Knowledge (BK) provided to the system in the form of fuzzy linguistic partitions over attribute domains. Each class of a partition is also labeled with a linguistic descriptor which is provided by the user or a domain expert. For example, the fuzzy labels {Low, Average, Above average, Important} could belong to a partition built over the domain of the attribute *quantity*. Figure 1.4 gives an example of the partitioning. In our *Import* relation, when the quantity of goods imported equals "400", "400" can be rewritten into two linguistic descriptors: (i) "Above average" and (ii) "Important". Both descriptors have a *satisfaction degree*, i.e., the accuracy of the descriptor regarding the attribute value, of 0.50. The general framework of the approach is illustrated on our *Import* relation in Figure 1.5.

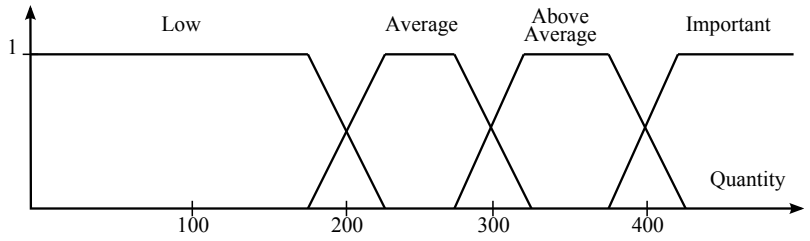


Figure 1.4: Example of linguistic partition for the *quantity* attribute

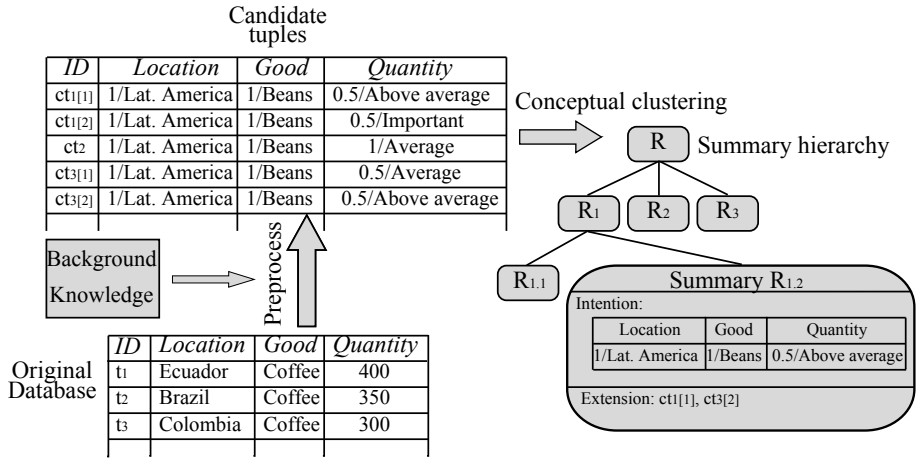


Figure 1.5: SaintEtiQ process applied to the *Import* relation

Once a *raw* tuple is rewritten into one or many *cooked* tuple(s), they are considered by a machine learning algorithm, i.e., a conceptual clustering algorithm. In our example, the first tuple is rewritten into two cooked tuples since quantity "400" is rewritten into two descriptors. First, the candidate tuples are classified into the output summary hierarchy from the root of the hierarchy. The conceptual clustering algorithm finds the best matching summary node to add each input candidate tuple following a top-down approach. At each summary node in the summary hierarchy, the hierarchy is modified to incorporate this new instance through operations that can create or delete child nodes. Decisions are taken based on a local optimization criterion called the Partition Quality (PQ) that tries to minimize the length of the intentional description of the summary. This intentional description is made of a fuzzy set of domain descriptors on each attribute associated with a possibility measure. Designed as a pluggable service-oriented system, SaintEtiQ has linear computational complexity, i.e., in $O(n)$ with n the number of tuples to summarize, for the construction and maintenance of the summaries.

The SaintEtiQ summarization process uses the semantics of input databases to produce summaries that are informative, i.e., no *desummarization* is necessary to comprehend the intention of a summary node. Since the intention of a summary node preserves the tabular structure of the original database, it could be used for other purposes, such as querying [VRUM04, Bec09]. Also, SaintEtiQ is an incremental process and has linear computational complexity. These are desirable properties for a summary structure to support more process-intensive applications such as data mining. However, the hierarchical summary produced by SaintEtiQ does not reflect the sequentiality of candidate tuples fed to the conceptual clustering process. This is the main shortcoming that does not qualify summaries generated by SaintEtiQ as support structures for chronology dependent applications.

3.3 Temporal databases

The temporal database research area is a vibrant and fruitful research domain that emerged in the early 1960's. The domain has driven hundreds of researchers to publish more than two thousand papers. The main aspects of this corpus of work can be found, and not limited to, in a series of surveys [BADW82, Soo91, Kli93, SS98, WJW98, Noh04].

Research in temporal databases was triggered by the observation that real world phenomena contain *time-varying* information, i.e., events occur at specific points in time, and objects and relationships exist over time. Conventional databases capture the state, or a *snapshot*, of an application, company or institution at a given point in time. However, this snapshot does not consider the history of the data and does not model the evolution of the events, objects or relationships in time.

This issue is specifically addressed in temporal databases. The time dimension is given different levels of representation, i.e., (i) on a tuple level or (ii) on an attribute level, and carry different semantics. Most common semantics are (a) *valid time*, (b) *transaction time* and (c) bi-temporal relation that combines both valid and transaction time. The *valid time* of a database fact is the collected time –spanning the past, present and future– where the fact is true in the real world. Intuitively, all facts in real world have a valid time, e.g., its creation time, however, this valid time may not be recorded for a number of reasons and could be defined by the user. For instance, the valid time may be unknown or it may be irrelevant from application point of view. The *transaction time* of a database fact is the time when the fact is current in the database.

Let us give the example from the Wikipedia article on "Temporal database" [4]. John Doe is a fictional man who was born on April 1975 in Smallville and who died in Bingtown

[4] http://en.wikipedia.org/wiki/Temporal_database

in April 2001. We suppose the life on John Doe is recorded in a conventional database table *Person(Name, Address)*, where *Name* is the primary key. Therefore, in this conventional database, an entry *(John Doe,Smallville)* is added when John Doe is born. This entry is then deleted and replaced by *(John Doe, Bingtown)* when John Doe moves to Bingtown. Finally, this entry is deleted when John Doe dies. This example shows the limitations of how conventional databases handle the temporal information related to John Doe. For instance, the information of how long John Doe lived in Smallville before moving to Bingtown is lost. Table 1.10 gives a sum up of the series of facts relating to John Doe. A temporal database corresponding to the facts relating to John Doe using the notions of valid time and transaction time is given in Table 1.11.

Table 1.11 clearly shows that it is possible to retrace John Doe's history on the timeline. For instance, fact entry e_1 states that John Doe's first address is in Smallville. This fact is recorded in the database at date 4th April 1975 (transaction time) but the fact itself is valid starting from 3rd April 1975 (valid time). Then, fact entry e_3 states that John Doe moved to Bingtown on 26th August 1994 and was recorded on 27th December 1994. Given this information, e_2 is inserted and e_1 preserved, i.e., e_1 is not deleted, to record the end of John's address in Smallville. e_1 is updated to reflect the end time of the transaction.

Date	**Event in real world**	**Database action**	**Database shows**
APR 3,1975	John is born	Nothing	No John Doe
APR 4,1975	John's father officially reports birth	Inserted: *Person(John Doe,Smallville)*	John Doe lives in Smallville
AUG 26,1994	John moves to Bingtown but does not register new address	Nothing	John Doe lives in Smallville
DEC 26,1994	Nothing	Nothing	John Doe lives in Smallville
DEC 27,1994	John registers new address	Updated: *Person(John Doe,Bingtown)*	John Doe lives in Bingtown
APR 1,2001	John dies	Deleted: *Person(John Doe)*	No John Doe

Table 1.10: Series of facts concerning John Doe

Entry	**Name**	**Address**	**Valid-from**	**Valid-to**	**Transaction-from**	**Transaction-to**
e_1	John Doe	Smallville	03Apr1975	∞	04Apr1975	27Dec1994
e_2	John Doe	Smallville	03Apr1975	26Aug1994	27Dec1994	∞
e_3	John Doe	Bigtown	26Aug1994	∞	27Dec1994	02Feb2001
e_4	John Doe	Bigtown	26Aug1994	01Jun1995	02Feb2001	∞
e_5	John Doe	Beachy	01Jun1995	03Sep2000	02Feb2001	∞
e_6	John Doe	Bigtown	03Sep2000	∞	02Feb2001	01Apr2001
e_7	John Doe	Bigtown	03Sep2000	01Apr2001	01Apr2001	∞

Table 1.11: Temporal database for John Doe facts

There has been extensive research on the theoretical foundations of temporal database. These research directions include (but are not limited to):

- How to model different semantics of time in conventional relational databases
- Design new query languages
- Query processing for temporal databases
- Storage and access methods to optimize time-constrained queries
- Incomplete temporal information
- Temporal integrity constraints, etc.

To the best of our knowledge, very small interest has been given to represent temporal databases into a more compact and concise form. The corpus of work most relevant to summarization in temporal databases includes the PACK operator defined for time intervals [DDL02], Bettini's work on semantic compression of temporal data [Bet01] and Wang et al.'s work on transaction time in temporal database systems [WZZ08].

The PACK operator [DDL02]

The PACK operator considers as input a unique n-ary relation R where at least one attribute is a time interval. This attribute is called "During". In our example in Table 1.11, the "During" time interval attribute could be defined from the "Transaction-from" and "Transaction-to" attributes. This operator produces another n-ary relation that is known as the *canonical form* of R, denoted PACK(R). The main purpose of this canonical form is to reduce redundancy of the original relation R. Intuitively, the PACK operator groups tuples in R that have identical descriptions, i.e., attribute values, and that are contiguous on the timeline, i.e., the tuples' time interval overlaps.

Let us give an example of how the PACK operator proceeds in Table 1.12. Table 1.12(a) is a temporal database and Table 1.12(b) is a representation of Table 1.12(a) where the attributes "From" and "To" are replaced by a "During" attribute. Finally, Table 1.12(c) gives the packed temporal database.

(a) Original temporal database

Item	From	To
I_1	d02	d04
I_1	d03	d05
I_2	d02	d05
I_2	d04	d06
I_2	d09	d10

(b) Preprocessed temporal database R

Item	During
I_1	[d02:d04]
I_1	[d03:d05]
I_2	[d02:d05]
I_2	[d04:d06]
I_2	[d09:d10]

(c) PACK(R)

Item	During
I_1	[d02:d05]
I_2	[d02:d06]
I_2	[d09:d10]

Table 1.12: Example of PACK operation

The intuitions behind the PACK operator is most interesting since it can handle attributes of any type, i.e., numerical and categorical. Also, tuples in R are packed based on their *contiguity* on the timeline, but note that here, the notion of contiguity is defined by the overlapping of the events' "During" attribute value. However, this PACK operator suffers from at least two shortcomings: (i) large numbers of attributes might result in few packings and (ii) further compaction could be achieved.

In the first case, our simple example shows that relation R is solely defined on the "Item" attribute and the PACK operation is straightforward. When there exists are larger number of attributes defined over large domains, it is less likely for contiguous tuples to have identical attribute values. In this situation, the PACK operator will have smaller impact and few tuples would be packed together. This shortcoming could be leveraged by introducing a mechanism to abstract attribute values such that they become identical at a certain level of representation.

In the second case, Table 1.12(c) shows that tuples (I_2,[d02:d06]) and (I_2,[d09:d10]) have identical "Item" attribute value, i.e., I_2, but these tuples are not packed together since their "During" attribute values do not overlap, i.e., [d02:d06] $\cap$ [d09:d10]=$\emptyset$. More compaction could be achieved and the number of tuples in Table 1.12(c) could be further reduced if tuples (I_2,[d02:d06]) and (I_2,[d09:d10]) were packed together. We argue that this packing operation makes sense since the tuples' "Item" attribute value is identical

and "During" time interval are *close* on the timeline. Hence, this shortcoming could be leveraged by introducing a mechanism to evaluate the *proximity* of the tuples' time interval on the timeline and representing, or *characterizing*, intervals [d02:d06] and [d09:d10] by a more abstract time interval representation.

Claudio Bettini's semantic compression of temporal data [Bet01]

Claudio Bettini propose in [Bet01] a method to compress a temporal database using the semantics regarding how the values of certain attributes evolve over time and how these values change when considered in terms of different time granularities. The author considers a temporal database is represented as a relational database with a valid time attribute. The compressed temporal database is then used for query answering.

The author relies on the notions of time *granularity* and *semantic assumptions* (more specifically, (i) *instant assumption* and (ii) *granularity assumption*) to achieve compression. On one hand, a time granularity is defined as a mapping G from integers to subsets of the time domain such that: (i) $\forall(i,j) \in \mathbb{N}, i < j$, $G(i)$ and $G(j)$ are not empty and each element in $G(i)$ is smaller than all elements in $G(j)$, and (ii) $\forall k \in \mathbb{N}, i < k < j$, if $G(i)$ and $G(j)$ are non empty, than $G(k)$ is non empty. This definition of a time granularity covers standard granularities like day, week, month, quarter, etc..

On the other hand, a semantic assumption provides a formal specification of how unknown values, i.e., implicit values, can be deduced from data explicitly present in the database. In other words, a temporal semantic assumption relies on the use of methods, called *interpolation methods*, to derive an implicit value from explicit values. For instance, one can decide to infer a missing value as the average value between the next and previous values. Semantic assumptions can be of two different nature: (i) *persistence* assumption, denoted $P_X(Y^{meth_1} \dots Y^{meth_n})$ or (ii) *granularity* assumption, denoted $I_X(Y^{conv_1} \dots Y^{conv_n})$. Under the persistence assumption, $P_X(Y^{meth_1} \dots Y^{meth_n})$ is understood as follows: implicit values of attribute Y_i are derived from attribute X thanks to interpolation method $meth_i$.

Under the granularity assumption, information for a certain granule of one time granularity can be derived from information at granules of a different time granularity. Let us explicit these definitions thanks to the example given in Table 1.13. This example presents a relation R=(Item, Price, Time) that represents the evolution of the price of items I_1 and I_2. Here, the time granularity G chosen is *quarter-since-2010*; Hence, G(1) is "1st quarter 2010", ..., and G(5) is "1st quarter 2011". Under the persistence assumption, we can observe that the price of an item increases by 5$ for each passing quarter. Thus, the interpolation method can be defined as $meth_1 = C_j + ((i-1) \times 5)$, where C_j is the price of item I_j at G(1) and $i \geq 1$. Under the granularity assumption, relation $R4$ could say: "If a given item price is stored in the relation for a given quarter, the same price can be considered a good estimate for any month (or week, day, hour, etc.) of that quarter." Hence, since item I_1 costs \$35 during 2nd quarter 2010, item I_1 also costs \$35 in May.

The authors also define a function ϕ called a mapping or time windowing function. Intuitively, this function gives the tuples that hold at a time granularity $G(i)$ of granularity G. In our example: ϕ(1st quarter 2010)=$\{(I_1, \$30), (I_2, \$80)\}$, ϕ(2nd quarter 2010)=$\{(I_1, \$35), (I_2, \$85)\}$ and ϕ(1st quarter 2011)=$\{(I_2, \$100)\}$. Using these concepts, a *temporal module* is defined as the triplet (R,G,ϕ). Putting all the pieces together, given a database and a set of assumptions, semantic compression is achieved by finding the minimal representation of the database such that no tuple implied by the assumptions appear in any of the temporal modules.

In our example, semantic assumption $P_{Item}(Price^{C_1+(i-1)\times 5})$ allows to imply tuple $(I_1, \$35)$, also, semantic assumption $P_{Item}(Price^{C_2+(i-1)\times 5})$ allows to imply tuples $(I_2, \$85)$

and $(I_2, \$100)$. In this case, these three tuples can be dropped from relation R as shown in Table 1.13(b). Consequently, Table 1.13(b) is the minimal representation of the database given in Table 1.13(a) w.r.t. these two semantic assumptions.

(a) Price relation R

Item	Price	Time
I_1	\$30	1st quarter 2010
I_1	\$35	2nd quarter 2010
I_2	\$80	1st quarter 2010
I_2	\$85	2nd quarter 2010
I_2	\$100	1st quarter 2011

(b) Compressed relation R'

Item	Price	Time
I_1	\$30	1st quarter 2010
I_2	\$80	1st quarter 2010

Table 1.13: Example of temporal database compression

This compression scheme allows to compress a temporal database in a way that somehow preserves the chronology of the tuples. Here, *compressed* tuples are simply dropped from the relation and are implied thanks to the semantic assumptions defined. However, the approach proposed comes with the cost of several strong constraints: (i) the interpolation methods *meth* (persistence assumption) or conversion methods *conv* (granularity assumption) only process numerical attributes, and (ii) the time granularity needs to be defined by the user. The latter shortcoming could be leveraged by using a taxonomy of the time dimension, however, the former shortcoming is more troublesome. Usual interpolation functions rely on the total ordering of numerical values to compute aggregates. In the case of categorical attributes where concepts are not necessarily ordered, these aggregate functions are not applicable, e.g., it is not clear how the *average* between concepts "cheap" and "expensive" should be computed.

ArchIS [WZZ08]

ArchIS is a system proposed by Wang et al. in [WZZ08] to support temporal applications on top of Relational Database Management Systems (RDBMS). The authors tackle the problem of designing an integrated solution for several problems: (i) expressive temporal representations and data model, (ii) powerful language for temporal queries and snapshot queries, and (iii) indexing, clustering and query optimization for managing temporal information efficiently. For this purpose of supporting and optimizing queries on transaction time data, the authors propose an XML-based approach to represent database history in a *grouped* form.

In temporal databases, when an attribute value is changed for a given tuple, e.g., a price or salary change, a new tuple is created in the database (remember that in temporal databases, tuples are not deleted so that the history of an entity can be tracked). Temporal queries frequently need to coalesce tuples, e.g., to track all the price changes of a given item. However, temporal coalescing is complex and hard to scale in RDBMSs.

The authors overcome this issue by representing temporal data in a form where attribute values are grouped according to their time of validity. Time intervals that are contiguous or overlap for a same attribute value are coalesced. In other words, a coalesced timestamp history, i.e., start time and end time, is associated to each attribute value. This representation is difficult to implement in flat files but is naturally represented in an XML format.

Let us illustrate this approach thanks to an upgraded version of our John Doe example in Table 1.14. Table 1.14(a) gives the original relation *Person(ID, Name, Address, Job, Salary, (Transaction-)From, (Transaction-)To)*. Table 1.14(b) is the grouped (or *com-*

pressed version) of the *Person* relation. Note that John Doe's name and ID does not change in this history, thus, these attribute values are legitimately grouped throughout the whole history.

The XML-based compressed structure produced by the authors is essentially used for query answering. Powerful temporal queries are expressed thanks to XQuery without the need to define a new query language. This approach is most interesting from summarization view point since it allows to reduce the representation of numerical and categorical attributes. Attribute values are gathered together if they are identical and if they are *close* on the timeline. Here, the notion of *closeness* is defined as contiguity or overlapping of the time intervals where the attribute value is true.

(a) Original *Person* relation

ID	Name	Address	Job	Salary	From	To
1001	John Doe	Smallville	Paperboy	$10,000	04Apr1975	27Dec1977
1001	John Doe	Smallville	Delivery man	$10,000	28Dec1977	01Feb1980
1001	John Doe	Smallville	Delivery man	$12,500	02Feb1980	27Dec1982
1001	John Doe	Bigtown	Engineer	$60,000	15Jun1984	31Dec1986
1001	John Doe	Bigtown	Engineer	$65,000	01Jan1987	31Jun1990
1001	John Doe	Beachy	Engineer	$70,000	01Jul1990	31Dec1994
1001	John Doe	Bigtown	Senior Engineer	$85,000	01Jan1995	31Dec2000
1001	John Doe	Bigtown	Senior Engineer	$85,000	01Jan2001	01May2010

(b) Compressed form of the *Person* relation

ID	Name	Address	Job	Salary
04Apr1975	04Apr1975	04Apr1975	04Apr1975	04Apr1975
			Paperboy	**$10,000**
		Smallville	27Dec1977	01Feb1980
			28Dec1977	02Feb1980
			Delivery man	**$12,500**
		27Dec1982	27Dec1982	27Dec1982
		15Jun1984	15Jun1984	15Jun1984
				$60,000
		Bingtown		31Dec1986
				01Jan1987
1001	**John Doe**		**Engineer**	**$65,000**
		31Jun1990		31Jun1990
		01Jul1990		01Jul1990
		Beachy		**$70,000**
		31Dec1994	31Dec1994	31Dec1994
		01Jan1995	01Jan1995	01Jan1995
		Bingtown	**Senior Engineer**	**$85,000**
01May2010	01May2010	01May2010	01May2010	01May2010

Table 1.14: Example of transaction database - *Person* relation

We believe the underlying ideas could be developed further to achieve a more compact representation: (i) Allow attribute values to be subsumed into more abstract concepts and (ii) perform compression tuple-wise. Indeed, in our example, during the period 04Apr1975 to 27Dec1982 John Doe has worked as a "paperboy" and a "delivery man", and earned a salary of "$10,000" and "$12,500". Intuitively, these primitive values could be rewritten into

a more abstract concept. For instance, the terms "paperboy" and "delivery man" could be subsumed by the concept "manual work" and the values "$10,000" and "$12,500" could be subsumed by the concept "low". These attribute values can be rewritten using Background Knowledge in the form of concept hierarchies as done in [SPRM05]). In this case, the compressed form of relation *Person* could be even more concise.

3.4 Data streams

In the early 90', researchers made the observation that a growing number of real world applications needed handle very large amounts of data generated by multiple sources in an automated way. Examples of such sources include: Financial tickers, performance measurements in networking monitoring, web usage logs, sensors, call detail records in telecommunications, etc.. These sources are assumed to be unbounded in size, continuously produce data in real time and eventually at very high and uneven rates. For these reasons, this data is not best modeled as *persistent* relations but as transient *continuous data streams*, also called *streams* for short. Processing continuous data streams for the purpose of knowledge discovery, e.g., find most frequent items in the stream, raises many challenges. Indeed, such a data-intensive environment comes with very strong requirements on the algorithms that process data streams. For instance, since streams are potentially infinite it is commonly accepted that streams can not be stored and processed offline. Incidentally, each input data object can only be considered once, either it is processed or discarded.

Here, we introduce the necessary notions to show how research on data stream processing is linked to our work on summarization of time sequences. For this purpose, we discuss in Section 3.4.1 the different representations of data streams used in the fruitful literature throughout the last two decades. We clearly state the nature and representation of the data streams we consider as related work. This distinction is of utmost importance since part of the research corpus on how to represent data streams within constrained memory will be ruled out of related work. Then, we discuss the limitations of traditional database management systems for handling streaming data and its consecrated applications, e.g., *continuous queries*. Indeed, continuous queries is a class of queries that makes sense thanks to the dynamic and constantly evolving nature of data streams. Traditional database management systems are very ill prepared for handling such queries. Hence, this is how we show the extent to which the ideas put in place to support continuous queries intersect with our work on summarizing time sequences of events.

3.4.1 Data stream model

Research on data streams, in its early stages, has given various definitions of a data stream. Here we present the most consensual definitions so that time sequence summarization can be positioned w.r.t. all the research efforts on data streams. Two main tendencies clearly appear in the literature on data streams: Data streams are either considered as (i) an ordered series of numerical objects or as (ii) an ordered series of relational tuples. In the former case, data streams are defined as a series of data points, which can be simple numerical values [BS05, Mut05] or more complex relation tuples [BBD$^+$02] defined on multiple numerical attributes. Definition 1.1 gives a formal definition of a data stream of numerical values and Definition 1.2 is its extension to streams of relational tuples defined on numerical attributes.

Definition 1.1 (Data stream of numerical values)
A data stream consists of an ordered sequence of data points $x[1], \ldots, x[i], \ldots, x[n], \ldots$ *such that the value of each data point* $x[i]$ *lies in a bounded range, i.e.* $x[i] \in [R_{min}, R_{max}]$.

Definition 1.2 (Multi-dimensional numerical data stream)
A multi-dimensional data stream consists of an ordered sequence of data points $x[1], \ldots, x[i], \ldots, x[n], \ldots$. *Let* R *be a relation defined on a set of* m *numerical attributes* A. *Each data item* $x[i]$ *in the data stream is a tuple in* R *and defined on attributes in* A. *Let* $x[i,k]$ *be the value of* $x[i]$ *on attribute* A_k *with* $1 \leq k \leq m$ *and* A_k *is numerical. Each data point value* $x[i,k]$ *lies in a bounded range, i.e.* $x[i,k] \in [R_{min}, R_{max}]$.

In the latter case, a data stream can be understood as an extension of traditional relational databases into a streaming environment. Therefore, each data point in a data stream is a relational tuple [ABB+04, DGGR02, Agg06a] defined on multiple attributes that can be numerical or categorical. Definition 1.3 formally defines such data stream as a generic multi-dimensional data stream.

Definition 1.3 (Generic multi-dimensional data stream)
A generic multi-dimensional data stream consists of an ordered sequence of data points $x[1], \ldots, x[i], \ldots, x[n], \ldots$. *Let* R *be a relation defined on a set of* m *numerical attributes* A. *Each data item* $x[i]$ *in the data stream is a tuple in* R *and defined on attributes in* A. *Let* $x[i,k]$ *be the value of* $x[i]$ *on attribute* A_k *with* $1 \leq k \leq m$ *and* A_k *is either numerical or categorical. If* A_k *is numerical, each data point value* $x[i,k]$ *lies in a bounded range, i.e.* $x[i,k] \in [R_{min}, R_{max}]$. *If* A_k *is categorical, each data point value* $x[i,k]$ *is a descriptor taken from the definition domain of* A_k.

These definitions implicitly embed a notion of *time* and ordering of data points in data streams. One can clearly notice the lack of a time dimension in the definitions given here. This notion of time can be either explicitly or implicitly expressed in each data point. In the case the notion of time is explicitly expressed, each data point in a stream is timestamped thanks to a designated attribute. Explicit timestamps are used when tuples correspond to a real world event that occurs at a particular time and that information is important for the understanding of the event. In contrast, when the notion of time is implicitly expressed, either the system that generates the stream does not include a timestamp or the time associated to a tuple is not important. In this case, for the purpose of ordering incoming data points of a stream, the data stream management system can add a special field to capture the arrival time of each item. This distinction between implicit and explicit timestamps is similar to that between transaction and valid time in the temporal database literature.

Interestingly, Muthukrishnan gives in his survey on data streams algorithms and their applications [Mut05] three formal models for data streams: (i) Time series model, (ii) Cash register model and (iii) Turnstile model. Suppose a *signal* A, also understood as a one-dimensional function $A : [1 \ldots n] \rightarrow \mathcal{R}$, is described by an input stream $a_1, a_2, \ldots$ arriving sequentially, item by item. Even though the author's definition of a data stream matches our definition of a mono-dimensional numerical data stream as given in Definition 1.1, the formalism of the three models introduced apply to generic multi-dimensional data streams. The model essentially differs on how data points a_i describe A:

• Time series model: Each a_i equals $A[i]$ and they appear in increasing order of i. This time series model is most suitable for updating the financial trade volumes each minute, or for observing the evolution of temperatures every hour, etc..

• Cash register model: Each a_i is an increment to $A[j]$. The state of the signal after seeing the ith stream item is denoted $A_i[j]$. Hence, since a_i is an increment I_i to the current state of signal A, $A_i[j]$ is expressed as follows: $A_i[j] = A_{i-1}[j] + I_i$, i.e., as the sum of the previous state $A_{i-1}[j]$ and the increment I_i. This model is similar to a

cash register where multiple incoming a_is could increment cash register $A[j]$'s state over time. This model is one of the most popular data stream model. It fits applications such as IP address monitoring, e.g., monitoring multiple accesses to a web server from a same IP.

• Turnstile model: Each a_i is an update to $A[j]$. The state of the signal after seeing the ith stream item is denoted $A_i[j]$. Hence, since a_i is an update U_i to the current state of the signal A, $A_i[j]$ is expressed as follows: $A_i[j] = A_{i-1}[j] + U_i$, i.e., as the sum of the previous state $A_{i-1}[j]$ and the update U_i where U_i can be positive or negative. This model is the most generic model and allows to capture fully dynamic situations, i.e., situations where there are inserts and deletes to the system.

Considering these formal models that describe data streams, we can easily state that research on time sequences summarization should be related to research on data streams modeled in the *cash register* model. Indeed, once produced, events in time sequences have a proper existence and can only be altered (or *incremented*) be an increment to a previous event. New events do not annihilate previous events.

3.4.2 Limitations of traditional DBMS

Processing data streams needs to take into account the specificities of streaming environments. Babcock et al. [BBD+02] list these constraints as follows:

- Data items in data streams arrive into the system in real time.
- The system might need to address multiple streams simultaneously.
- The system has no control over the order of arrival of stream items within a data stream or across multiple streams.
- The arrival rate can be very high and uneven.
- Streams are potentially unbounded in size.
- Each data item in a stream can only be processed exactly once. Once processed, a data item is either discarded or archived, eventually in a transformed version. Since streams are potentially unbound in size, the storage memory remains limited in comparison to the size of the stream.

Hence, systems intended to manage data streams need to be able to cope with the continuous arrival of multiple rapid, time-varying, possibly unpredictable and unbounded streams. The naive approach would be to simply load the arriving data stream into a traditional Relational Database Management System (RDBMS) and operate on it. However, traditional RDBMS are not designed for continuously loading individual data items. These systems are ill prepared to handle heavily streamed-oriented applications.

Specific data stream systems have been developed to address monitoring, analysis or query answering needs. The application domains include finance, web applications, security, networking and sensor monitoring. iPolicyNetworks [iPo] and Traderbot [Tra] are examples of such dedicated systems. iPolicyNetworks provides an integrated security platform with services such as firewall and intrusion detection over multi-gigabyte network packet streams. Traderbot is a web-based financial search engine that evaluates queries over real-time streaming financial stock data. The website allows users to register one-time queries as well as continuous queries.

3.4.3 Handling contiguous queries

Continuous queries, in contrast with traditional *ad-hoc* queries, are queries that are issued once and that henceforth run *continuously* over a dataset. Relational databases are not originally designed to support contiguous queries since the history of a tuple's updates is not necessarily maintained. Note that this history could eventually be maintained if the relational database is the representation of a temporal database, in this case, related work is presented in Section 3.3. However, in data streaming environments, continuous queries gain full meaning and are of utmost importance.

Answering continuous queries on data streams is one of the main challenges that have led to the development of specific systems. As a stream is updated with new items, new results can match a registered continuous query. These results are then returned to the user or application. Supporting continuous queries on data streams is a challenging task. The issue was first implicitly tackled by Gupta and Mumick with *materialized views* [GM99], which is a form of continuous query since materialized views are continuously updated to reflect changes on the underlying relations. Materialized views have then been extended to the chronicles data model [JMS95].

Answering continuous queries on data streams was first explicitly defined in *Tapestry* [TGNO92], then a large corpus of work has followed, including, *TelegraphCQ* [CC03], *ATLaS* [WZL03], *Aurora* [CC02], the *Tribeca* [Sul96] stream-processing system for network traffic analysis, *GSQL* [CJS03] is a SQL-like language developed for *Gigascope* or *CQL* [ABW06], a SQL-based declarative language to register continuous queries against streams and updatable relations, etc..

This corpus of research work on query answering in data streams has quickly raised the necessity to provide approximate answers to ad-hoc and continuous queries. Arasu et al. [ABB$^+$04] started to distinguish between (i) queries that can be answered exactly given a bounded amount of memory and (ii) queries that must be approximated unless steam items were accessible on disk. The authors show that, without prior knowledge of the size of input streams, it is impossible to bound the memory requirements for most common queries such as join queries (unless attribute domains are restricted) or aggregate queries.

These results has thus motivated a large corpus of work on developing general techniques for data reduction and synopsis construction. The general techniques developed include *random sampling*, *histograms*, *wavelets*, *synopsis* (or *sketches*) or *sliding windows*. The basic intuition behind these approaches is to produce a concise yet informative representation of the stream that can be maintained in memory. This is the reason why our research work on time sequence summarization intersects with the Algorithms community efforts in the data streaming environment.

3.4.4 Prior work

Most of the corpus of work intended for data streams of numerical objects as defined in Definition 1.2 is out of the scope of our work. The techniques proposed usually rely on the total order defined on numerical values, e.g., to compute aggregates, distances, similarities, etc.. Since there does not exist a natural total order on categorical attribute values, these techniques poorly adapt to multi-dimensional categorical data streams. However, provided some adaptation to categorical attributes, the underlying ideas in some approaches can still be useful.

Sampling

Sampling [Olk93, OR90, Vit85, CMN99, JPA04, HK04, CMR05, JMR05, EN06, Agg06b] is a old and simple, yet powerful, probabilistic tool to decide if a data item should be processed

or not. Sampling allows to capture in reduced and constrained memory the most recent data from an input stream. The boundary of the error rate induced by sampling is used as a function for the sampling rate. Hoeffding's [Hoe63] error upper bound, i.e., the probability for the sum of random variables to deviate from its expected value, has been used to measure the sample size, e.g., with Very Fast Machine Learning techniques [DH01]. However, sampling approaches usually suffer from the following shortcomings: (i) the input size of a stream is unknown so finding the error bounds requires special analysis, (ii) sampling does not address the problem of fluctuating data rates that is inherent to data streams and (iii) sampling does not fit applications that require more historical data, e.g., top-k queries or continuous queries. Nevertheless, sampling could still be used as a complementary approach for keeping most relevant recent data.

Load shedding

Load shedding [BM03, TCZ+03] is a technique similar to sampling. The idea is to drop portions of the data stream. Load shedding has been successfully used for approximate query answering. However, this approach does not suit all data mining tasks, e.g., sequential pattern mining, since it could drop trunks of the data stream that might represent or contribute to a pattern of interest.

Sketching

The idea of sketching [BBD+02, Mut05] is to randomly project a subset of features. This approach can be understood as vertical sampling of the input data stream. Sketching has been applied in comparing data streams and aggregate queries.

Synopsis

Creating a synopsis data structure is the application of a *summarization* technique to represent an incoming data stream into another stream or format that can be used for further analysis. There exists a potpourri of techniques that comply to this definition of a synopsis that include wavelet analysis, frequency moments, quantiles or histograms. We list here the most relevant methods:

- Quantiles [GK01, LXLY04, AM04, GRM05, ZLX+06]
- Frequency moments [BBD+02]
- Wavelet-based analysis [VW99, CGRS00, GKMS01, GKS04, KM05]
- Single- or multi-dimensional histograms [PIHS96, JMN99, GK01, GKS01, TGIK03, GGI+02, MM02, WS03, GSW04, Dob05, KNRC05]: Histograms are efficient tools to quickly capture the frequency density of intervals (in the case of numerical attributes). In the case of categorical attribute, histograms could capture the frequency of each attribute value. However, since there does not naturally exist a total order on categorical attributes, histograms of categorical domains loose the benefit of capturing the frequency density of aggregates such as on numerical intervals.

Sliding window

Also relevant to summarization, the *sliding window* [BBD+02, BDMO02] model captures the most recent data items in a stream. As the data stream progresses, items from the end of the window are discarded and new items are added. Sliding windows have several attractive properties that allow it to be used for applications such as approximate query

answering. The semantics of a sliding window are clear and users understand what the approximation is. Sliding windows are deterministic and emphasizes recent data.

Data mining – clustering

Data stream, in particular data stream clustering, is another popular form of data stream summarization that has attracted extensive research in [GMMO00, AHWY03, GMM+03] and in [OWNL04, HXDH03, Agg06a, GLSS06, AHWY07, CCC08]. Guha et al. perform data stream clustering using a k-median based approach [GMMO00, GMM+03]. The proposed approach makes a single pass over the data stream and generates k clusters that use small space. The approach requires $o(nk)$ time and $O(n\epsilon)$ space. The authors also prove that any k-median approach that achieves a constant factor approximation can not achieve better run time than $O(nk)$. Babcock et al. improve Guha et al.'s approach in [BDMO03] thanks to the use of Exponential Histograms (EH). To do so, the authors address the problem of merging clusters when the two sets of cluster centers to be merged are far apart. This is achieved by maintaining an EH structure. Lin et al. [LKLC03] propose SAX to transform time series into a symbolic representation. Dimensionality and numerosity reduction is achieved by computing piecewise aggregate approximation and in a second step, each aggregate is represented by a discrete symbol, e.g., {a,b,c,...}. Aggarwal et al. [AHWY03, AHWY07] propose a clustering framework called CluStream based on micro-clusters and a pyramidal time frame for summarizing massive data streams. The authors build a set of micro-clusters also called *snapshots* at particular moments in time which follow a pyramidal pattern. A micro-cluster is a set of information represented as *cluster feature vectors* [ZRL96] to which is added information about timestamps. The pyramidal time frame provides an effective trade off between the storage requirements and the ability to recall summary statistics from different time horizons.

The techniques enumerated here show a common feature: They all consider clustering on numerical data streams. To the best of our knowledge, small interest has been given to clustering or building summaries for categorical data streams. However, this need is very true and actual. Methods for categorical data streams seem to have mainly emerged these last few years with the following contributions [HXDH03, OWNL04, CL05, PMR06, Agg06a, WFZ+08].

Among the most promising techniques, Ong et al. [OWNL04] propose SCLOPE, a clever hybrid approach based on CluStream and CLOPE [YGY02]. CLOPE is a clustering algorithm originally developed for transaction data. The idea behind CLOPE is to build clusters while optimizing a global criterion function that tries to increase the intra-cluster overlapping of transaction items by increasing the height-to-width ratio of *cluster histograms*. A cluster histogram is simply a histogram that captures the frequency of transaction items of transactions in a cluster. Therefore, SCLOPE capitalizes on CluStream by building micro-clusters and using the pyramidal time frame, and on CLOPE by addressing the issue of categorical attributes thanks to the use of cluster histograms. Since SCLOPE addresses clustering on data streams, cluster histograms are built in one pass using the *FP-Tree* structure [HY00]. The pyramidal time frame allows to integrate a time logics for building snapshots. Unfortunately, this approach lacks a mechanism to sequentialize micro-clusters within a snapshot.

We proposed in previous work an approach [PMR06] to summarize data streams using a conceptual clustering method. Since the approach is based on SaintEtiQ, the approach fundamentally suffers from the same shortcomings. The summary produced does not reflect the chronology of the input steram items and can not be directly exploited by chronology dependent applications.

After much work on numerical data streams, Aggarwal et al. have switched interest in [Agg06a] towards categorical data streams. The authors propose a methodology similar to Online Analytical Processing algorithms (OLAP) where statistical summary data is stored at regular interval for the purpose of answering repeated queries. Most interestingly, the authors assign a time-sensitive weight to each data point. Thanks to a fading function f, this weight is used to express the decay rate of each data point which is used to control the freshness of historical data. The clusters are a set of *cluster droplets* that contain a number of statistical information: (a) number of co-occurring categorical attribute values, (b) number of possible values of each categorical attribute occurring in a cluster, (c) number of data points in the cluster, (d) sum of time weights at time t and (e) the timestamp of the last data item added to the cluster. The authors have introduced an elegant method to handle the time decay of a cluster, however, cluster droplets are grown only based on their similarity with an incoming data point and not their *distance* or *proximity* on the timeline.

3.5 Event sequences

System or user monitoring applications, in domains such as telecommunications, biostatistics or the WWW, generate large amounts of sequence data called *event sequences*. Event sequences consist of *events* that occur at specific points in time. A formal definition of an event sequence is given in Definition 1.4. Alarms in telecommunication networks and web access logs are examples of such event sequences. In the case of telecommunication networks, the set E of event types could be the set of possible error messages. In the case of web access analysis, the set E of event types could be the set of possible pages. Note that as event sequences are defined, event sequences can easily be understood as a particular instance of multi-dimensional data streams, temporal database or even relational database (where a tuple is an event). This explains why a number of data mining techniques from these areas have been adapted to event sequences. We give in Figure 1.6 an example of event sequence as defined in Definition 1.4.

Definition 1.4 (Event sequence)
Let $\mathcal{E}$ be a set of event types. An event sequence is a collection of pairs (E, t) where $E \in \mathcal{E}$ is an event type and t is the time of occurrence of event E.

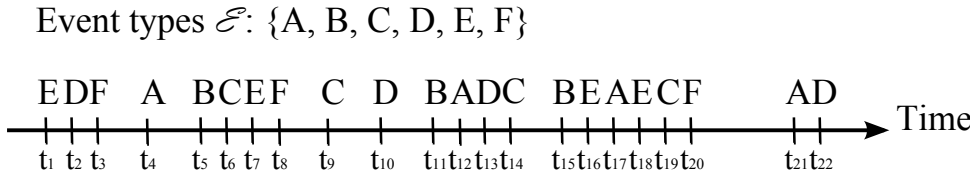


Figure 1.6: Example of event sequence

The practical applications of the event sequence data model has attracted many research efforts in mining frequent patterns [MTAI97, Spi99a, SOL03, MR04, Abr06, SBC06, ENK06, HC08, OKUA09], also known as frequent *episodes*. These data mining algorithms succeed in extracting recurring local patterns, but prove inadequate in the task of finding a global model of the data. Also, the knowledge discovered by these technique usually overwhelm the analyst. In such case, analysts require additional tools to visualize or navigate through the output results. The alternative idea is to create *summaries* of the event sequences to provide a global model of the data [KT08, KT09].

In [KT08, KT09], Kiernan and Terzi propose a formal definition of event sequence summarization and give the desirable properties an event sequence summary should have:

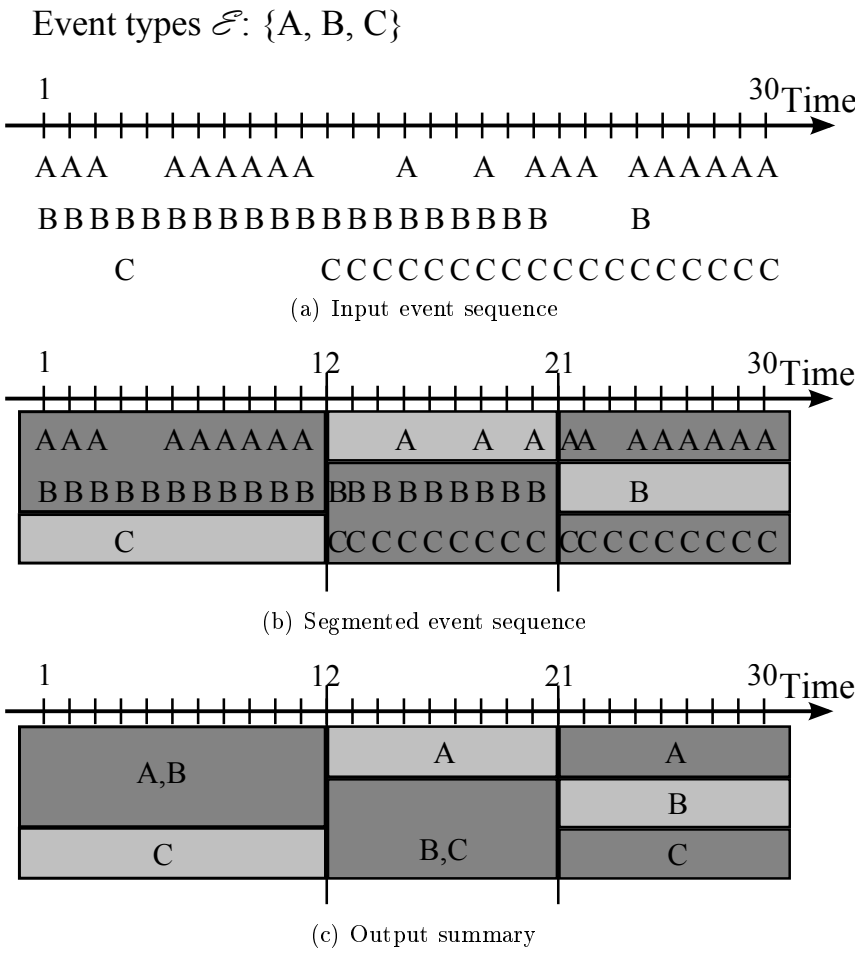
1. **Brevity and accuracy:** The summarization system should construct *short* summaries that *accurately* describe the input data.

2. **Global data description:** The summaries should give an indication of the global structure of the event sequence and its evolution in time.

3. **Local pattern identification:** The summary should reveal information about the local patterns: Normal or suspicious events or combination of events that occur at certain points in time should be identified by just looking at the summary.

4. **Parameter free:** No extra tuning should be required by the analyst in order for the summarization method to give informative and useful results.

The authors consider particular event sequences where multiple events could occur at a same time point t. They propose to rely on the Minimum Description Length (MDL) principle to produce in a parameter-free way a comprehensive summary of an event sequence. The core idea is to segment the input event sequence timeline into k contiguous, non overlapping, intervals also called *segments*. Thereafter, the portion of data corresponding to each segment S is described thanks to a local model M. The local model M can be understood as a partition of event types $\mathcal{E}$ into groups $\{X_1, \ldots, X_m\}$ such that $X_i \in \mathcal{E}$ and $X_j \cap X_{j'} = \emptyset$ for every $j \neq j'$. Each group X_j is described by a single parameter $p(X_j)$ that corresponds to the probability of seeing an event of any type in X_j within segment S.



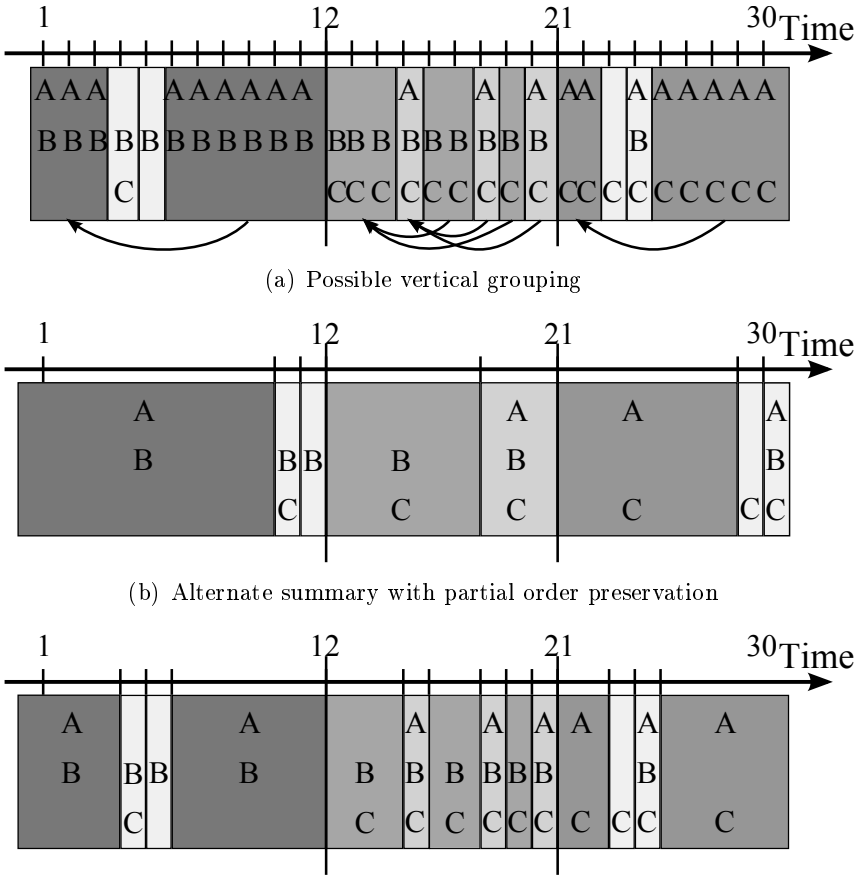
(a) Input event sequence

(b) Segmented event sequence

(c) Output summary

Figure 1.7: Example of summarization

In a nutshell, the authors formalize the event sequence summarization problem as the problem of identifying an integer k, segmenting input sequence S into a partition of k segments $\{S_1, \ldots, S_n\}$ and identifying for each segment S_i the model M_i that best describes the data in S_i while minimizing a total description length metric. The authors show that this problem can be optimally solved in polynomial time and give the corresponding algorithm. Also, the authors propose an alternative sub-optimal but practical and efficient approach. Figure 1.7 gives an example of summaries produced by Kiernan and Terzi. The shaded area for each group in the summary in Table 1.7(c) represents the probability of seeing an event from that group. For more readability, the proportions of groups $X_{1,2} = \{C\}$, $X_{2,1} = \{A\}$ and $X_{3,2} = \{B\}$ are not respected in Table 1.7(c).



(a) Possible vertical grouping

(b) Alternate summary with partial order preservation

(c) Alternate summary with total order preservation

Figure 1.8: Example of alternate summarizations

The summary produced by Kiernan and Terzi effectively reduces the input event sequence into a set of $k = 3$ segments and gives interesting information on the data density in each segment. However, since the model used to describe each segment S is a probabilistic model, traditional mining algorithms can not directly operate on the summary output. Also, the model M used to describe each segment S looses the temporal information of events within segment S. The temporal information lost is the order in which events appear in the system. For instance, in segment S_1 in Table 1.7(b), events A and B arrive at $t = 1$, then, at $t = 4$, event A stops arriving and event C arrives. This information is lost in Table 1.7(c).

In order to address this issue, a preprocessing step to *desummarize* the summary could be adopted beforehand. For instance, the preprocessing step can simply consist in associating an approximate time of occurrence, or timestamp, to each group X_j in S, so that

groups X_j can be chronologically ordered. This timestamp could be computed from the timestamp of events in each group, e.g., thanks to an aggregate function *min*, *max*, *avg*, etc.: This operation is also known as a *characterization* of the timestamp. Thereafter, traditional mining methods could apply to the preprocessed summary.

In its current state, the summarization operated here can be understood as a *horizontal* grouping of data in each segment S. The alternative approach would be a *vertical* grouping of events in each segment S. Vertical grouping would allow to address the issue of loss of temporal information in a controlled manner. We give an example of an alternate way to operate summarization in Figure 1.8. Figure 1.8(a) gives one possible vertical grouping of events in each segment. The corresponding summary obtained is presented in Figure 1.8(b). We can note in this figure that the sequentiality of events is partially preserved, e.g., in segment S_1, we preserved the sequence $S' = \langle\{A, B\}, \{B, C\}, \{B\}\rangle$ but lost the information that $\langle A, B\rangle$ is the last group of events in the original segment. Figure 1.8(c) presents a second alternative summary. In this case, the overall sequentiality of events in completely presented. These examples show that the *compression ratio* achieved, here expressed as the ratio $\rho = 1 - \frac{\#\text{groups in summary-1}}{\#\text{events in original sequence-1}}$, by the different alternative summaries is different, i.e., in Figure 1.8(b) $\rho = 0.76$ and in Figure 1.8(c) $\rho = 0.55$. This difference in compaction gain is traded off in the form of temporal accuracy of the summary produced.

3.6 Discussion

Summarization techniques developed in each research area came with specific limitations. We propose to sum up in Table 1.15 the main limitations identified for each approach. As a reminder, time sequence summarization was introduced as an activity to support chronology dependent and process-intensive applications on very large data sets. Hence, a number of desirable properties naturally arise: For instance, (i) the summarization algorithm should have low algorithmic complexity, (ii) all the data should be represented in the summary, (iii) the chronology of events should somehow be reflected in the summary, (iv) applications should operate seamlessly on summaries, etc.. For this reason, we consider that the related works, studied throughout this chapter, present a limitation when they do not satisfy these intuitive and natural properties. From Table 1.15, in each research area, we can identify some main shortcomings w.r.t. these properties. These shortcomings are listed and analyzed in the following paragraphs:

- Customer transaction databases: The purpose of summarization in this area is to (but not limited to) facilitate knowledge extraction tasks. This objective explains why summaries are built using data mining techniques, e.g., Frequent Itemset Mining (FIM). However, such pattern mining techniques rely on algorithms that have high computational complexity. For instance, the candidate generation phase in Apriori-based algorithms explodes exponentially with the decrease of the minimum support parameter. Also, once patterns are generated, the notion of chronology of events in the summary does not hold anymore. Hence, these two observations explain the limitations identified in Table 1.15.

- Relational databases: Some relational database summarization techniques are designed for storage purposes. Therefore, research works have focused on optimizing the compaction of databases in a fast and efficient manner. Hence, higher compaction gain is usually achieved by considering the time dimension as any other dimension and results in disregarding the temporal ordering of the data.

- Temporal databases: In contrast with relational databases, here, the time dimensions (valid time and transaction time) are use to discriminate the tuples to be grouped

together. Hence, most research work has focused on these time dimensions and have given low interest in designing mechanisms for grouping the data based on its content.

Data model	**Ref.**	**Shortcomings**				
		Complexity	Coverage	Chronology	Substitution	Other observations
Customer TDB	[CK05]	High	Yes	Yes		Based on FIM
	[WK06]		Yes			Based on FIM; No explicit compaction gain
	[XJFD08]	Polynomial		Yes	Yes	Based on FIM
	[WA05]			Yes		Generates a structure for Apriori-based SPM
Relational DB	[Cai91]			Yes		
	[JMN99]					Low support for categorical data
	[BGR01]	$O(\log n)$ to $O(\rho\frac{n^2}{2})$		Yes	Yes	
	[JNOT04]				Yes	Compressed table R_c not informative
	[SPRM05]			Yes	Yes	
Temporal DB	[DDL02]					No mechanism to abstract the data; Low compaction gain on high dimension data
	[Bet01]					Numerical data only
	[WZZ08]					No mechanism for tuple wise compression
Data streams	[OWNL04]			Yes		Lacks a mechanism to order micro-clusters within a snapshot
	[PMR06]			Yes	Yes	Based on SaintEtiQ
	[Agg06a]			Yes		Cluster droplets grown only based on their similarity; Lacks a mechanism that considers the temporal proximity of droplets
Event sequences	[KT08]	Polynomial		Yes	Yes	Lacks a mechanism to order groups within each segment

Table 1.15: Shortcomings of main summarization approaches proposed in the literature

- Data streams: Data streams processing is a young research area. The main applications that drive research efforts in this area include financial analysis, monitoring sensor networks, etc.. By essence, these application deal with numerical data and few efforts have been given to summarize categorical data streams. Here, the closest activity to time sequence summarization is the clustering of categorical data streams. However, these methods mainly suffer from the lack of a mechanism to handle the temporal information associated to events, i.e., events are solely clustered based on their content.

- Event sequences: Finally, Kiernan and Terzi's efforts on summarizing event sequences is closest to our research work. The main difference between the authors approach to summarization and our understanding of time sequence summarization lies in the final objective of the summary. Indeed, Kiernan and Terzi designed event sequence

summaries to help the end-user understand his data. On the other hand, we understand time sequence summarization as a support structure for other applications. This difference in our objectives explain (i) why the author's technique only partially takes into account the temporal information of events, i.e., through segmentation, and (ii) why the summary produced can not be directly exploited by other applications.

In this section, we have thoroughly discussed work related to time sequence summarization developed for customer transaction databases, relational databases, temporal databases, data streams and event sequences. This corpus of related work has shown that building summaries capable of (i) handling categorical domains and (ii) the temporal dimension of data are by themselves challenging tasks. However, to the best of our knowledge, designing summaries for time sequence data that simultaneously take into account (i) the content and (ii) the temporal dimension of the data is a new and very challenging research problem. In the following section, we formally define the concept of *Time sequence summarization.*

4 Time Sequence Summarization: Problem Definition

In this section, we present the problem of building time sequence summaries. We give an illustrative example then introduce the terminology necessary to formally define *Time sequence summarization.*

4.1 Illustrative example

To illustrate the ideas and concepts exposed in this section, we generate in Example 1.1 a simple toy example with a time sequence of events extracted from conference proceedings. Here, we associate to author N. Koudas a time sequence of events where each event is one conference publication timestamped by its date of presentation. For simplicity, the set of descriptors describing an event is taken from one single descriptive domain, namely, the paper's *topic.* Without loss of generality, this discussion is valid for any number of descriptive domains.

Example 1.1
We present here an example of time sequence of events extracted from conference proceedings. Descriptors in the itemsets are taken from the topic *descriptive domain. This descriptive domain represents the different topics covered by conference papers. This sequence is given in Table 1.16.*

Author	Event descriptors	Time
N. Koudas	x_1={Datastreams, Aggregation}	JUN05
	x_2={Datastreams, Top-k query}	AUG06
	x_3={Top-k query}	AUG06
	x_4={Top-k query}	SEP06
	x_5={Join query, Selection query}	SEP06
	x_6={Clustering}	SEP07

Table 1.16: Example of time sequence of events in conference proceedings

4.2 Terminology

Let Ω be the universe of discourse, i.e., the set of all categorical or numerical values that could describe a data object. In our research work, we call *descriptor* any categorical or

numerical value taken from Ω and we call *event* any data object defined as a part of Ω. The term *event* is a generic term for designating any data object, be it a simple numerical value or a complex relation defined on categorical and/or numerical attributes.

$\Omega = \bigcup_{\mathcal{A}} D_A$ is organized into several attribute domains D_A, also called *descriptive domains*, corresponding to each domain A that is used to describe the content of an event. Example 1.1 gives an example of time sequence of events in conference proceedings. For author N. Koudas, each conference publication is described by the paper's *topic*, e.g., the *topic* descriptors {Data streams, Aggregation} are used to describe the paper entitled "Multiple Aggregations Over Data Streams" presented in June 2005.

We refer to a part of Ω, i.e., a subset of descriptors taken from various descriptive domains, as *itemset* $x \in \mathcal{P}(\Omega)$. Therefore, an event e is denoted $e = (x, t)$ and is defined by (i) an itemset x that describes e and (ii) is associated with a time of occurrence, or *timestamp*, t. In our case, the paper entitled: "Multiple Aggregations Over Data Streams" is described by descriptors x={Data streams, Aggregation} and presented at SIGMOD in t="JUN05". We assume a *time sequence of events* (or *time sequence* or *sequence* for short) for a given entity s, e.g., N. Koudas, is the set of events relating to s ordered by ascending timestamp. A formal definition of a time sequence of events is given in Definition 1.5.

Definition 1.5 (Time sequence of events)
A time sequence of events s, denoted $s = \{e_1, \ldots, e_n\}$ with $n \in \mathbb{N}$, is a series of events $e_i = (x_i, t_i)$, $1 \leq i \leq n$, where x_i is an itemset in $\mathcal{P}(\Omega)$ and e_is are ordered by ascending timestamp t_i. We denote by $S = \{x_1, \ldots, x_n\}$ the support multi-set of s. A time sequence s verifies: $\forall (e_i, e_j) \in s^2,\ i < j \Leftrightarrow t_i < t_j$. We denote by $s[T]$ the set of timestamps of events in S.

It should be noted that this definition does not require an explicit timestamp to exist for a time sequence to be defined. Indeed, we make the same assumption as in the definitions of data streams given in Section 3.4. Events in a stream or sequence can have either an explicit or implicit time of occurrence. This only depends on the need or requirements from application view point. In the case events do not have an explicit time of occurrence, the associated timestamp can simply consist of an integer that corresponds to the index of the event in the sequence of events.

Also, one should note that defining the set of descriptors that describes an event as a part of Ω makes our time sequence model more generic and richer than the usual definitions that rely on relations. The usual relation-based representation of complex streams, sequences or relational databases is in a normalized form [Cod70]. Here, we allow multiple descriptors from a same descriptive domain to be used for precisely describing an event.

Definition 1.5 states that events in a sequence are ordered by ascending timestamp, be it explicit or implicit. This total order on timestamps allow us to lighten the formal expression of a time sequence. Therefore, we can equivalently use the notations:

- $s = \{e_1, \ldots, e_n\}, n \in \mathbb{N}$
- $s = \{e_i\}, 1 \leq i \leq n, n \in \mathbb{N}$

We denote by $\mathcal{S}(\Omega)$ the set of time sequences of events defined on $\mathcal{P}(\Omega)$. This notion of time sequence can be generalized and used to define a sequence of time sequences that we hereafter call *second-order time sequence*. A formal definition of second-order time sequences is given in Definition 1.6.

Definition 1.6 (Second-order time sequence)
A second-order time sequence of events defined on $\mathcal{S}(\Omega)$ is a time sequence $\overline{s} = \{(Y_1, t'_1), \ldots,$

$(Y_n, t'_n)\}$ *with* $n \in \mathbb{N}$, *or equivalently,* $\overline{s} = \{(Y_i, t'_i)\}$ *with* $1 \leq i \leq n$, $n \in \mathbb{N}$. *The second-order time sequence* $\overline{s}$ *can be understood as a series of events* (Y, t') *where each* itemset Y *is itself a regular time sequence of events in* $\mathcal{S}(\Omega)$ *and* t' *is a timestamp. Events* (Y, t') *in* $\overline{s}$ *are ordered by ascending timestamp* t'.

Since itemset Y *is a regular time sequence,* Y *can be expressed as follows:* $Y=\{e_j\}$, *where* $e_j = (x_j, t_j)$ *and* $1 \leq j \leq m$. *Then, timestamp* t' *is an aggregative timestamp value that is computed from the timestamps of events that compose* Y, *i.e.,* $t_1, \ldots, t_m$. *The aggregative function that computes* t' *can be of any nature, e.g., min, max, avg, etc..*

The set of second-order time sequences of events defined on $\mathcal{S}(\Omega)$ *is denoted* $\mathcal{S}^2(\Omega)$.

Let us give an example of second-order time sequence from Example 1.1. N. Koudas's time sequence can be reorganized as a second-order time sequence as follows:
$\overline{s} = \{(Y_1, t'_1), (Y_2, t'_2), (Y_3, t'_3))\}$ where:

- Y_1=$\{(x_1$=$\{$Datastreams, Aggregation$\}$, t_1=JUN05$)\}$ and t'_1=t_1=JUN05
- Y_2=$\{(x_2, t_2), (x_3, t_3), (x_4, t_4), (x_5, t_5))\}$ and $t'_2 = min\ \{t_2, \ldots, t_5\}$, i.e., t'_2=AUG06, where:
  - x_2=$\{$Datastreams, Top-k query$\}$ and t_2=AUG06
  - x_3=$\{$Top-k query$\}$ and t_3=AUG06
  - x_4=$\{$Top-k query$\}$ and t_4=SEP06
  - x_5=$\{$Join query, Selection query$\}$ and t_5=SEP06
- Y_3=$\{(x_6$=$\{$Clustering$\}$, t_6=SEP07$)\}$ and t'_3=t_6=SEP07

A second-order time sequence can be obtained from a time sequence s as defined in Definition 1.5 by the means of a form of *clustering* based on the semantics **and** temporal information of events e in s. Reversely, a time sequence can be obtained from a second-order time sequence $\overline{s}$ by means of *concept formation* [Mic80,MS80] computed from events (Y, t') in $\overline{s}$.

4.3 Problem statement

We have introduced in previous section all the elementary concepts required for defining a time sequence summary as understood in this thesis work. We give a formal definition of a time sequence summary using these concepts in Definition 1.7.

Definition 1.7 (Time sequence summary)
Given s *a time sequence of events in* $\mathcal{S}(\Omega)$, $s = \{e_i\}$ *with* $e_i = (x_i, t_i)$ *and* $1 \leq i \leq n$, *we define the time sequence summary of* s, *denoted* $\chi(s) = (s^2_C, s^\star_M)$ *as follows:*

- $s^2_C = \{(Y_j, t'_j)\}$ *with* $1 \leq j \leq m \leq n$, *is a second-order time sequence of events. An event* (Y, t') *in* s^2_C *is (i) a* group Y *of events* e *taken from* s *and obtained thanks to a form of* clustering C *that relies on the events' semantic and temporal information, to which is associated a time of occurrence* t'.

- $s^\star_M = \{(x^\star_j, t^\star_j)\}$ *is a regular time sequence of* concept *events, or* concepts *for short, obtained by* characterizing, *or by* forming concepts from, *events* (Y, t') *in* s^2_C. *Concept* $x^\star_j$ *is obtained from the characterization of event itemsets in group* Y_j *and* $t^\star_j$ *is obtained from the characterization of event timestamps in group* Y_j.

s^2_C *and* $s^\star_M$ *can be understood as the* extension *and the* intention, *respectively, of summary* $\chi(s)$.

For example, let $(\overline{s}, s^\star)$ be N. Koudas's time sequence summary where $\overline{s}$ is the extension of the summary as defined in Section 4.2 and $s^\star$ is intention of the summary, i.e., the sequence of concepts formed from groups in $\overline{s}$. The summary is detailed as follows:

- $\overline{s} = \{(Y_1,\text{JUN05}), (Y_2,\text{AUG06}), (Y_3,\text{SEP07})\}$
- $s^\star = \{(\{\text{Datastreams, Aggregation}\},\text{JUN05}), (\{\text{Query optimization}\},\text{AUG06}), (\{\text{Clustering}\},\text{SEP07})\}$

The given definition of a time sequence summary is purposely generic. We used terminology borrowed from conceptual clustering research since the underlying ideas are similar, i.e., group data objects based on their *similarity* or *proximity*. The novelty of time sequence summaries relies on the fact that events are grouped thanks to their semantic and temporal information. As discussed previously, traditional clustering techniques mostly rely on the joint features of the data objects considered. Their similarity is evaluated on these features thanks to a distance measure, e.g., Euclidean distance or Manhattan distance in a metric space, or entropy-based measure in categorical spaces. In time sequence summarization, a form of temporal approximation should also be applied so that events that are close from temporal view point are grouped. Consequently, local rearrangement of the data objects on the timeline should also be allowed.

From applicative view point, the objective of time sequence summarization is to support chronology dependent applications that operate on time sequences of events. For this purpose, time sequence summaries should display certain desirable properties for effectively supporting such applications. We list the desirable properties of time sequence summaries in Property 1.8.

Definition 1.8 (Properties of a time sequence summary)
The desirable properties of a time sequence summary are the following:

1. ***Brevity:*** *The number of summarized events in the output time sequence should be reduced in comparison to the number of events in the input time sequence.*

2. ***Substitution principle:*** *An application that operates on a time sequence s should be capable of seamlessly performing if the input time sequence is replaced by its summary intention.*

3. ***Informativeness:*** *Summarization should transform and reduce time sequences of events in a way that keeps the semantic content available to and understandable by the analyst without the need for* desummarization.

4. ***Accuracy and usefulness:*** *The input time sequence of events should not be overgeneralized to preserve descriptive precision and keep the summarized time sequence useful. However, ensuring high descriptive precision of events in the summarized time sequence requires trading off the* brevity *property of the summary.*

5. ***Chronology preservation:*** *The chronology of summarized events in the output time sequence should reflect the* overall *chronology of events in the input time sequence.*

6. ***Computational scalability:*** *Time sequence summaries are built to support other process-intensive applications and, thus their construction should not become a bottleneck. Applications such as data mining might need to handle very large and long collections of time sequences of events, e.g., news feeds, web logs or market data, to rapidly discover knowledge. Therefore, the summarization process should have low computational and memory requirements.*

One should note that the desirable properties given in Definition 1.8 partially overlap with those given in the definition of an event sequence summary proposed by Kiernan and Terzi in [KT08, KT09]. The main difference resides in the fact the authors impose the

summarization approach to be *parameter free*. In return, our definition of a time sequence summary impose the *chronology preservation* and the *substitution principle* properties.

We believe the chronology preservation property is of utmost importance, since it marks the delimitation between traditional summarization or clustering techniques with time sequence summarization. In the case the chronology preservation property is completely ignored, time sequence summarization degenerates into a simple summarization or clustering approach that has for sole objective to achieve or optimize a compression ratio. However, when the chronology preservation property is handled, ensuring high level of temporal accuracy requires trading off the brevity property, i.e., compression ratio, of the summary. Therefore, there exists a fragile equilibrium to maintain so that ensuring high compaction gain does not degenerate the summarization process into a traditional summarization or clustering technique.

4.4 Discussion

In a nutshell, the objective of time sequence summarization is to find the *best* method for grouping events in a time sequence of events based on events' semantic content and proximity on the timeline. Note that this general definition of time sequence summarization can partially encompass Kiernan and Terzi's work on event sequences summarization [KT08, KT09]. Indeed, when Kiernan and Terzi segment an event sequence S into k segments S_i, $1 \leq i \leq k$, this segmentation can be understood as organizing S into a second-order time sequence s^2_C where the method C is their segmentation method, e.g., Segment-DP. Note that the authors' segmentation method does not permit any form of approximation that would allow to rearrange events on the timeline: In other words, events that belong to a segment S_i can not be rearranged on the timeline to belong to a different segment S_j, e.g., S_{i-1} or S_{i+1}. Consequently, this rigid definition of segmentation could limit the compaction gain of the algorithm.

Also, the authors describe each segment S by a set of event type groups $\{X_i\}$ where each group X_i gathers event types of similar appearance rate. Hence, the model M used to describe each segment is a probabilistic model. To this point, our definition of a time sequence summary fully generalizes Kiernan and Terzi's work. However, the authors perform what we named in Section 3.5 *horizontal grouping* and associate to each event type group X_i its probability of appearance $p(X_i)$ in S. By doing so, the authors do not support (i) the chronology of appearance of each event type group nor (ii) the substitution principle.

These two property violations could be addressed simultaneously as follows. The summarization process could be adapted to our problem definition if the authors implemented a form of *vertical grouping* (as introduced in Figure 1.8(b) and Figure 1.8(c) in Section 3.5) on top of their horizontal grouping. Hence, their current horizontal grouping will simply be used to collect distribution statistics of event type groups in each segment and the vertical representation of event type groups can directly be processed by applications.

5 Chapter summary

In this chapter, we have explored and discussed the concepts that constitute "Time sequence summarization". At first glance, the notions of "Time sequence" and "Summarization" are not well defined in the common knowledge literature nor in research literature. Therefore, we introduce the definition of "Time sequence summarization" that we consider. This definition allows us to differentiate "Time series" from "Time sequences" and "Summarization" from "Compression". Defining exactly the taxonomy of "Time sequence summarization" considered allowed us to precisely delimit and pinpoint all related work in the literature.

In fact, summarization of time-varying data is a concept widely used across a broad range of research activities on data models that include: customer transaction databases, relational databases, temporal databases, data streams or event sequences. Since the data generated worldwide tends to increase at a very fast rate, researchers have focused on designing summary structures to support applications on these data sources. However, due to the lack of an unifying definition or framework, each domain has developed specific summarization methods for specific applications and needs. We studied in details the main approaches proposed in each domain and reached the following conclusion. This corpus of existing work has shown that building summaries capable of (i) handling categorical domains and (ii) the time dimension of the data are by themselves challenging tasks. However, designing summarization methods capable of simultaneously handling both types of data is a new and an even more challenging problem.

Therefore, we formally gave a definition for the concept of "Time sequence summarization" and introduced all the desirable properties a time sequence summary, that complies to our definition, should display. We showed that this definition can partially encompass previous works such as Kiernan and Terzi's most recent work on event sequence summarization. The main difference in Kiernan and Terzi's definition of summarization and ours resides in the desirable properties a summary should display.

In the following chapters, we will present different "Time sequence summarization" techniques and we will study how time sequence summaries can support in practice a chronology dependent application, namely, Sequential Pattern Mining. Hence, in Chapter 2 we present a user-oriented technique to build a time sequence summary, called TSaR. TSaR builds on top of the ideas of Attribute Oriented Induction and uses background knowledge in the form of domain specific taxonomies to represent events at higher levels of abstraction. Summarization is achieved by gathering *similar* events that occur *close* on the timeline.

Chapter 2

TSAR: a user-centered *T*ime *S*equence Summ*aR*y

1 Introduction

Data mining and summarization methods designed for operating on time sequences of events, or time-varying data in general, are often custom-tailored for specific applications. Such applications include knowledge discovery, visualization, specific queries, pattern mining, storage, etc.. Some previous summarization work [SPRM05, KT08] has attempted to propose more general purpose summarization techniques. Unfortunately, there can not exist a silver bullet method capable of addressing different users' and different applications' needs since these needs and the way summaries are built could be mutually exclusive. For instance, on the one hand, summaries produced by SUMMARY [WK06] are built on top of frequent closed itemsets mined from transaction databases. Since frequent closed itemsets are mined, SUMMARY is susceptible of loosing information. This loss of information is conditioned by the choice of the minimum support value that indicates which itemsets will be extracted from the transaction database. On the other hand, other applications, e.g., infrequent item mining [XCY06, BMS07, CYL+08] or outlier detection [HA04, CBKC07], could use summaries as a support structure. Clearly, these applications can not leverage summaries built by SUMMARY.

In [KT08], Kiernan and Terzi propose a more general purpose summarization approach that builds event sequence summaries in a parameter-free manner. The authors emphasize on the importance of unburdening the user from the need to parameterize the summarization algorithm. We actually believe this property is of importance since summarization tools are designed to assist users in their data analysis tasks and not make them loose time on this preprocessing task. Also, users are not computer scientists and do not necessarily have the technical background to parameterize and tweak complex algorithms. At least, if parameterization is required from the user, it should be simple, intuitive and comprehensive [HBMB05]. The structures produced should be informative w.r.t. the parameters chosen. For instance, if Kiernan and Terzi's algorithm [KT08] was parameterized, it should simply require the user to specify the maximum number of segments into which the timeline should be segmented.

The downside of a parameter-free methodology remains the lake of control over the output produced. For instance, let us consider in Figure 2.1 the alternative summary produced from Kiernan and Terzi's summary. This figure shows that the group of events $X_{3,3} = \{A, B, C\}$ in segment S_3 has the possibility of being grouped with other groups in segment S_2. The rationale behind this grouping could be: "The group of events $X_{3,3}$ arrived lated due to network delays, hence, it should be given a chance to be gathered with

one group in previous segment S_2". From user view point, the absolute groupings and the increased volume of information imposed by the strict segmentation of the timeline could be considered a limitation of parameter-free approaches. This limitation could be frustrating since the way events are grouped could go against the user's perception of content w.r.t. time and the way he would have grouped the events. Visually and conceptually, these event groups are identical and seem to occur *relatively close* on the timeline, why not gather them together? Actually, Miller [Mil62] found that individuals tend to focus on filtering and omitting (ignoring) information as the primary effective way of coping with this sort of information overload. This means that anyways, the user might gather $X_{3,3}$ with another group in segment S_2.

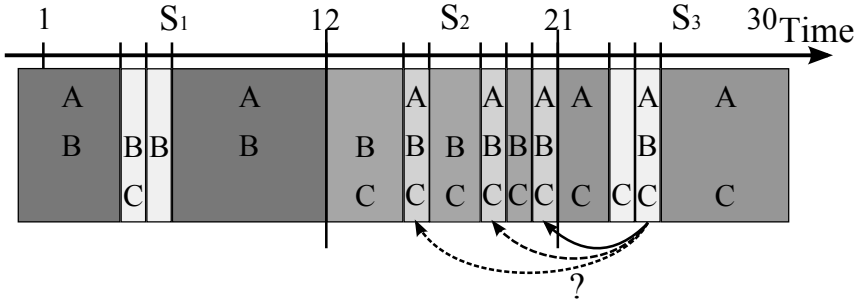


Figure 2.1: Alternate summary

In this chapter, we propose to operate time sequence summarization in a way that limits the number of parameters required from the user, i.e., limited to two parameters only. The two parameter settings required from the user should be simple and intuitive and allow him to express the way he wants the data represented w.r.t. his understanding and preferences of the domain of representation of the data. Hence, we propose a solution that involves more actively the user. This user-oriented *T*ime *S*equence summ*AR*ization approach is called TSAR. The purpose of TSAR is to provide users with a tool that can produce an informative and useful summary of time sequences of events. The idea is to allow users to customize, with the help of their knowledge and understanding of the domain, the summary produced in a simple and intuitive way. Later in Chapter 3, we will propose a full-fletched parameter-free time sequence summarization approach. The contributions discussed in this chapter are the following.

Contributions

First, we propose to build a time sequence summary that display all properties given in Definition 1.8, in Chapter 1, in a user-oriented way. We present the user-oriented time sequence summarization technique called TSAR. TSAR is a summarization method that proceeds in three phases: (i) generalization, (ii) grouping and (iii) concept formation. The generalization phase is responsible for representing the data at a higher level of abstraction while the grouping phase gathers similar events that are close on the timeline. The concept formation phase is responsible for producing a regular time sequence from the sequence of groups produced in the grouping phase. The accuracy of the summaries produced is controlled in the generalization and grouping steps by the user thanks to two parameters. These parameters reflect the user's preference in terms of: (i) the level of abstraction to which the data should be represented and (ii) the level of temporal precision of events represented on the timeline.

Second, we evaluate and validate our summarization method thanks to an extensive set of experiments on real world data: Reuters's 2003 financial news archive, i.e., approxima-

tively 1.2M news events. We define a semantic and temporal accuracy measure to evaluate the quality of summaries produced by TSaR. Hence, we summarize the input data and evaluate each approach on four dimensions: (i) computational time, (ii) semantic accuracy, (iii) temporal accuracy and (iv) compression ratio. Our experimental study shows that TSaR has linear computational complexity and can achieve high compression ratio. The user can produce summaries that achieve even higher compression ratio by trading off semantic and/or temporal accuracy. This *control* over the compression ratio is obtained by actioning the generalization and the grouping parameters.

Organization of the chapter

The remaining of this chapter is organized as follows. We present in Section 2 an overview of the TSaR approach, then we discuss in Section 3 the fundamental concepts on which TSaR relies and the work related to these fundamental concepts. Using the concepts introduced, Section 4 details the mechanisms of TSaR. Section 5 presents the implementation of TSaR and studies the technique's computational complexity and memory footprint. We present our experimental study in Section 6 and conclude this chapter in Section 7.

2 Overview of TSaR

TSaR is a technique designed to produce a concise, yet informative, time sequence summary from a time sequence of events as defined in Definition 1.5 in Chapter 1. As a reminder, events in an input time sequence of events are described thanks to a set of descriptors taken from various descriptive domains and associated to a time of occurrence. In market basket analysis terminology, an event is described by an *itemset* where items are taken from various descriptive domains and the associated time of occurrence is a *timestamp*. This timestamp is used for organizing events in ascending order. The particularity of this data representation resides in the fact an event can be described by multiple descriptors taken from a same descriptive domain. This particularity makes this representation of events richer in terms of content and more general in terms of data model. However, this step towards generality has a price in terms of complexity of the underlying concepts and in terms of computational complexity. We will discuss this aspect later in Chapter 3.

Example 2.1
Time sequence of events extracted from conference proceedings. Descriptors are taken from the topic *descriptive domain and represent the different topics covered by conference papers. This sequence is given in Table 2.1.*

Author	Event descriptors	Time
N. Koudas	x_1={Datastreams, Aggregation}	JUN05
	x_2={Datastreams, Top-k query}	AUG06
	x_3={Top-k query}	AUG06
	x_4={Top-k query}	SEP06
	x_5={Join query, Selection query}	SEP06
	x_6={Clustering}	SEP07

Table 2.1: Time sequence of events in conference proceedings for N. Koudas

We recall in Example 2.1 our example of time sequence of events extracted from conference proceedings. It will be used throughout this chapter to illustrate the notions introduced.

An overview of TSaR is given in Figure 2.2. The basic principle of TSaR is to represent event descriptors in a more abstract form using some form of *Background Knowledge* (BK). Here, this background knowledge is provided to the system in the form of domain specific taxonomies, i.e., IS-A hierarchies of concepts. These taxonomies are used to generalize event descriptors into their antecedents and consequently reduce the variability of event descriptors. Since the variability of events is reduced, some events might have *similar* descriptions. In this work, we consider for simplicity that two sets of event descriptors are *similar* if both sets are identical. If this situation occurs and such generalized events occur *close* on the timeline, they are gathered together to form a *group* of generalized events. Each group produced this way is then represented by a single event, also known as a *representative event*. This representative event is produced in our case by picking one event from the corresponding group.

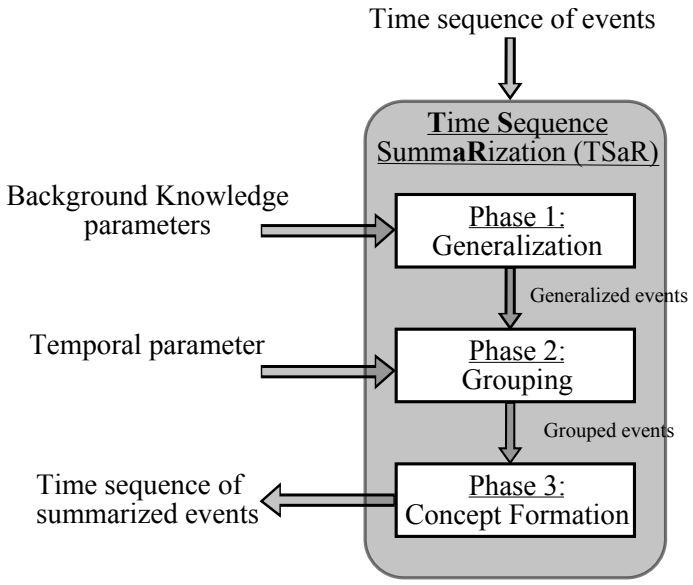


Figure 2.2: TSaR's basic principle

Putting the pieces together, TSaR is designed as a 3-phase process that relies on event descriptors *generalization*, event *grouping* and *concept formation* from groups formed. The user controls the generalization phase thanks a generalization parameter that operates with the taxonomies input into the system. This parameter expresses the level of abstraction at which the user wants the data to be represented. The *closeness* of *similar* events is controlled by a temporal parameter. This parameter allows the user to express how precisely the chronology of representative events in the output time sequence summary should reflect the chronology of their corresponding events in the original time sequence.

3 Fundamental concepts

In this section, we present the fundamental concepts involved in TSaR and discuss their related work. TSaR's overview showed that the approach represents event descriptors at different levels of abstraction thanks to the use of background knowledge provided to the system in the form of domain specific taxonomies. Here, we will justify our choice to input background knowledge in the form of domain specific taxonomies. Also, TSaR considers that events that are *similar* and *close* on the timeline should be gathered together. Hence, we discuss how this notion of temporal closeness of events on the timeline is handled by introducing the concept of *temporal locality*, borrowed from Operating System research.

3.1 Background Knowledge

Previous observations regarding parameter free summarization approaches has highlighted the necessity for separating summarization techniques into two families of approaches. On the one hand, Kiernan and Terzi impose event sequence summarization algorithms to be parameter free. This property allows summaries to be automatically built without the need for human intervention. This is a nice property since it requires no intellectual effort from the user, and, the summaries produced can eventually be exploited by other analysis tools. On the other hand, the summaries produced in such manner might not fit other users' natural way of grouping objects. Hence, in this second family of techniques, we believe the user should be given a more proactive role in the summarization process. However, the user should be involved in a simple and intuitive manner, i.e., in a way that is comprehensive and that does not require strong technical background. This approach concurs with Hiltz and Turoff for whom [HT85]:

> "Unless computer-mediated communication systems are structured, users will be overloaded with information. But structure should be imposed by individuals and user groups according to their needs and abilities, rather than through general software features."

Here we attempt to provide users with a tool that allows them to customize the way summarization is to be operated, i.e., decide the way sequence data is to be represented and the way such resulting data is to be grouped. For this purpose we go back to the fundamentals and rely on research on Cognitive Science. In the early 70's, research on the role of organization in human memory has coined the concept of *Semantic memory* in contrast with *Episodic memory*. Mainly influenced by the ideas of Reiss and Scheers who distinguished in 1959 two forms of primitive memory, namely, *remembrances* and other *memoria*, Tulving proposed to distinguish *Episodic memory* from *Semantic memory*. Episodic memory [Tul93, Tul02], also called *autobiographical memory*, allows one to remember events personally experienced at specific points in time and space. For example, an instance of such memory is the name and place of the last conference one has attended. On the other hand, semantic memory is the system that allows one to store, organize and connect one's knowledge of the world and make it available for retrieval. This is a knowledge base that anyone can obtain through learning, training, etc., that one can access quickly and without effort. In contrast with episodic memory, this knowledge does not refer to unique and concrete personal experiences but to factual knowledge of the world.

Bousfield and Cohen [BC52] advanced the idea, later shared by Mandler [Man67], that individuals tend to organize or *cluster* information into groups or subgroups. Numerous data models have been proposed to represent semantic memory, including: Feature models, Associative models, Statistical models and Network models. In the network model, knowledge is assumed to be represented as a set of *nodes* connected by links. The nodes may represent concepts, words, expressions, perceptual features or nothing at all. Links may be weighted such that some links are *stronger* than others.

A particular instance of network model are the *semantic networks* [BLV02]. In semantic networks, nodes are to be interpreted as standing for physical or non physical entities in the world, classes or kinds of entities, relationships or concepts. Links between nodes encode specific relationships between entities, concepts, etc., where the type of the relationship is a symbolic label on the link. Semantic networks can be characterized as being *general* or *restricted*.

On the one hand, general semantic networks are also known as *ontologies* in Ontology research [DF02a, DF02b]. There exists various meanings for ontologies [SKSC06]: Philo-

sophical ontology, Domain ontology or Formal ontology. In particular, Domain ontology is defined as a representation of things that exist within a particular domain of reality such as medicine, geography, ecology, law, etc.. More precisely, the most commonly used and accepted definition of a domain ontology is the one proposed by Gruber [Gru93] where:

> "An ontology is a *formal*, *explicit* specification of a *shared conceptualization*. Given a phenomenon in the world, the term *conceptualization* refers to an abstract model that identifies the relevant concepts to the phenomenon. *Explicit* means the concept used and the constraints on their use are explicitly defined. *Formal* refers to the fact an ontology should be machine readable. *Shared* reflects that ontologies should be able to capture consensual knowledge accepted by communities."

Example 2.2 (Example of conceptualization)
The object "Porsche 911" can be conceptualized as:
- IS-A "car", that IS-A "motor vehicle",
- is INST of "supercar" and
- is COMPOUND-OF {4 wheels, engine, 2 doors, breaks, etc.}

Example 2.2 gives an example of conceptualization for the object "Porsche 911". This example shows that ontologies are rich tools and have high semantic expressiveness. For instance, this conceptualization would also apply to the real world object "Porsche Cayman". In fact, the precision of this conceptualization results from the choice of the links and the associated semantics (here, the links are IS-A, INST and COMPOUND-OF). We argue that such detailed conceptualization requires (too) much effort, domain knowledge and proficiency from individuals to generate machine processable ontologies.

On the other hand, in restricted semantic networks, most common understandings of a link are: (i) IS-A relationship, i.e., generalized-to-specialized relationship, and (ii) INST relationship (standing for *instance of*), i.e., an individual-to-kind relationship. The most interesting purpose of restricted semantic networks is its strict hierarchical organization of concepts. For instance, the IS-A relationship defined in Example 2.2 can be extended as follows[1]: a "car" IS-A "motor vehicle", which IS-A "self-propelled vehicle", which IS-A "wheeled vehicle", which IS-A "vehicle", which IS-A "transport", etc.. For a given knowledge domain, an instance of such restricted semantic network is also known as a *taxonomy* over concepts of the domain.

Smith et al. [SKSC06] define a taxonomy as a tree-form graph-theoretic representational artifact with nodes representing universals or classes and edges representing IS-A or subset relations. Note that a representational artifact is a representation fixed on some medium (e.g., a text, a diagram, etc.) so that the cognitive representations that exist in an individual's mind (e.g., the concept "car") become publicly accessible in an enduring manner (e.g., a drawing representing the "car"). From this discussion, organizing knowledge in a hierarchical way is in practice a process that naturally occurs to individuals. For this reason, we believe feeding the TSAR summarization process with background knowledge in the form of domain specific taxonomies is a reasonable and simple way to allow the user to control the level of representation of the data. For instance, the descriptors of the *topic* domain in our running example in Table 2.1 can be organized into the taxonomy shown in Figure 2.3.

Therefore, the use of taxonomies allows users to control *which* event descriptors are to be abstracted and *what* they are to be abstracted into. Intuitively, abstraction is achieved

[1] Extracted from WordNet [Lab]

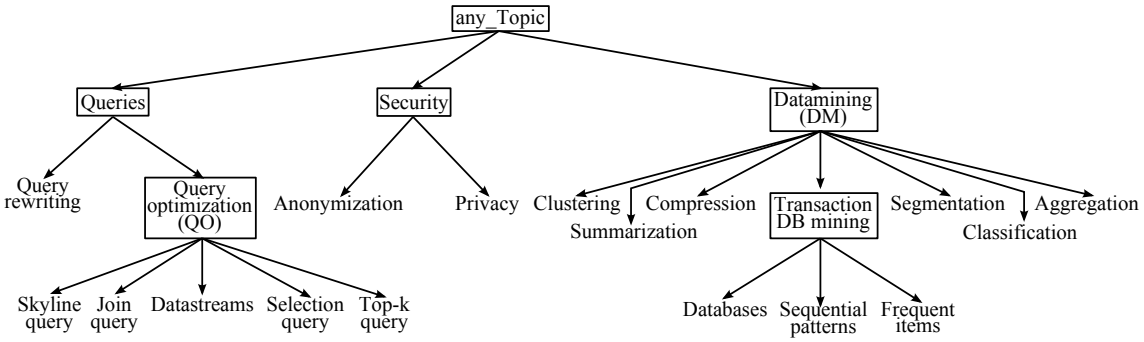


Figure 2.3: Taxonomy for the *topic* descriptive domain

by generalizing event descriptors following the IS-A relationship defined in the taxonomy. In other words, abstraction is achieved by rewriting a descriptor (one or multiple times) into its antecedent in the taxonomy. However, taxonomies alone do not specify *how*, or *to what level*, this abstraction process is carried on. We will discuss this matter in Section 4.1.

3.2 Temporal locality

The second fundamental idea on which TSAR relies on is the concept of *temporal locality* borrowed from Operating Systems research [Den05]. In fact, temporal locality is a special case of the more general concept of *locality of reference*, also known as the *Principle of locality* [ASU86]. In Operating Systems research, the locality principle is the phenomenon that a collection of data locations (in Random Access Memory – RAM), referenced in a short period of time in a running computer, often consists of relatively well predictable clusters. An example of application of this locality principle in programming languages, e.g., C, is the reuse of memory addresses to optimize matrix multiplication operations.

Temporal locality is a special case of locality of reference that assumes that if, at a given point in time, a data location is referenced, then it is likely that the same location will be referenced again in the near future. The presence of temporal locality has been recognized in a number of applications such as web server caching [CC00, SMG02, KS02, Min02] and exploited to improve performances such as the *hit ratio*, i.e., the number of times (from all accesses) a requested file was found in the cache.

Therefore, in this approach, we make the assumption that there exists a phenomenon of temporal locality in time sequences of events and transpose the temporal locality principle to summarizing time sequences of events. Hence, it this work, we assume that events that occur *close* on the timeline have high probability of referring to a same *phenomenon* or *topic* [2]. Note that these events might eventually need to be expressed at a higher level of abstraction to be *similar*.

However, events in a time sequence of events could eventually be intertwined, e.g., due to network delays, or arrive out of order, e.g., in highly distributed environments [LLD+07]. Intuitively, making the assumption that there exists a temporal locality at each point in time and estimating (or arbitrarily attributing a value to) this temporal locality would allow to gather events that actually belong together, in the sense they describe a same phenomenon or topic.

[2] In Topic Detection and Tracking (TDT) research pursued under the DARPA Translingual Information Detection, Extraction, and Summarization (TIDES) program [TID], this *phenomenon* is also called a *topic*

4 Detailed mechanisms of TSaR

We have given in previous sections an overview of TSaR and discussed the fundamental concepts on which TSaR relies to achieve time sequence summarization. In the following discussion, we detail the mechanisms of the summarization process. Hence, TSaR proceeds in 3 phases to achieve summarization: (i) generalization phase, (ii) grouping phase and (iii) concept formation phase. The detailed overview of the process is shown in Figure 2.4.

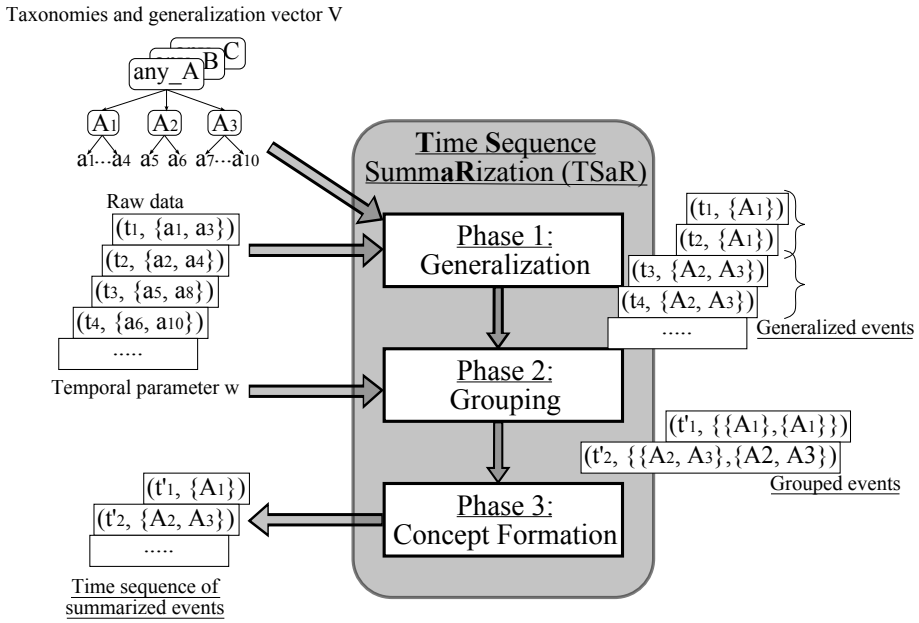


Figure 2.4: TSaR principle more detailed

The generalization phase is responsible for representing input event descriptors into a more abstract form thanks to the use of background knowledge. This background knowledge is given to the system as a collection of domain specific taxonomies, one for each descriptive domain on which event descriptors are defined. Also, the level of abstraction of each event descriptor is defined by a user defined parameter called the *generalization vector* ϑ. This vector controls for each descriptive domain, the level to which event descriptors are to be rewritten. Hence, the generalization vector allows the user to control the precision of the summary from content view point.

Since rewriting the input data reduces its variability, generalized events could eventually have *similar* descriptions. As a reminder, we assumed earlier that two sets of event descriptors are considered *similar* if they are *identical*; Hence, we will interchangeably use the terms *identical* and *similar* in the remaining of this chapter. Then, the grouping phase is responsible for grouping generalized events that have identical sets of descriptors. This grouping process is controlled by a temporal locality parameter w. Intuitively, parameter w indicates how far ahead in the time sequence an event should be looked up for grouping, i.e., w indicates the maximum *distance* on the timeline separating two similar events that allows them to be grouped. Understood under a different angle, parameter w fixes (or estimates) the temporal locality for each event. Hence, w allows the user to control the precision of the summary from temporal view point.

The last phase is the concept formation phase. This phase is responsible for generating a representative event for each group of events produced in the grouping phase. Since we consider events to be similar when event descriptors are identical at a given level of abstraction, concept formation is straightforward. We arbitrarily choose the oldest event

in a group as the representative event of the group.

We discuss in details the different phases that compose the TSAR algorithm in the following sections.

4.1 Generalization phase

The generalization phase is responsible for representing event descriptors at a higher level of abstraction. This effect is achieved by means of rewriting event descriptors into their antecedents in the taxonomies provided to the system. Here, we will discuss how these taxonomies can be obtained and how they are exploited in TSAR.

4.1.1 Obtaining Background Knowledge in the form of taxonomies

The term *taxonomy* is the practice and science of classification. The word itself derives from two ancient Greek stems: *taxis* and *nomos*, i.e., "order" or "arrangement" and "law" or "science", respectively. Organizing knowledge has attracted scientists worldwide throughout the ages, in particular in agriculture, botany or biology. Agriculture, botany or biology in general were most certainly the first domains that needed thorough knowledge organization [SheBC] (an interesting timeline of the history of taxonomy is given in [Bih]). The actual task of a taxonomist as defined in the Merriam-Webster's Dictionary online is to orderly classify plants and animals according to their presumed natural relationships. Incidentally, there naturally exists a large corpus of ontologies and taxonomies publicly available in the domains of biology, botanic or medicine. We give a summary[3] of some of the publicly available sources in Table A.1 in the Appendix of this thesis.

However, the emergence of research on the Semantic Web [BLHL01] has designated ontologies and taxonomies as natural forms of Knowledge Representation [DSS93]. The basic idea in Semantic Web research is to extend the current World Wide Web so that computers can *understand* information content. For this purpose, knowledge representations have been developed for numerous specific domains including [Ont]: Food, Countries - geography, Cyc, Enterprise, Music, Wine, Finance [Van], etc..

These observations back our assumption that users have access to publicly available domain specific ontologies and taxonomies. Even though these repositories exist, they do not necessarily cover all knowledge domains. In this case, there exists a number of tools that can help the user generate ad-hoc taxonomies. For instance, a corpus of work has focused on automatic generation of taxonomies using Wikipedia's *categories* classification [SJN06, PS07, CIK+08]. Indeed, in [PS07], Ponzetto et al. rely on the fact Wikipedia allows for structured access by means of *categories* since May 2004. These categories form a graph which can be taken to represent a conceptual network with unspecified semantic relations. The authors use this graph to derive IS-A and NOT-IS-A relations from the generic links in the graph thanks to the use of methods based on the connectivity of the network and on applying lexico-syntactic patterns (Hearst patterns [Hea92]) to very large corpora.

A state-of-the-art approach for inducing semantic taxonomies is proposed by Snow et al. [SJN06]. The authors address the problem of sense-disambiguated noun hyponym acquisition. They propose to use evidence from multiple classifiers over heterogeneous relationships in association with a novel coordinate term learning model. In other words, the authors extract hypernyms from a corpus and solve the problem of choosing the correct word sense to which to attach a new hypernym.

[3] Source: Wikipedia

Since lexico-syntactic patterns have low performances in extracting *all* relations from a *generic* corpus, Kliegr et al. [CIK+08] argue that traditional semantic taxonomies induction still yield relatively poor results. The authors propose to apply lexico-syntactic patterns on *one suitable* document with the intent to extract exactly *one hypernym* at a time. They refer to this approach as Targeted Hypernym Discovery (THD) and argue that THD can achieve much higher extraction accuracy than conventional approaches, e.g., close to 90% in image concept classification [KCN+08].

Also, an interesting work that can be leveraged for automatically generating taxonomies is the system proposed by Yao et al. [YHC09] for constructing evolutionary taxonomies from a collaborative tagging system, e.g., Del.icio.us [4] for web page bookmarking or Flickr [5] for photo sharing. The authors believe online collaborative tagging systems have become popular enough and are simple enough for acquiring user domain knowledge. Indeed, the authors argue that the tags used to describe an online content are generally correlated and evolve according to the changes to the content. Hence, the authors propose an approach based on association rule mining to discover the co-occurrences of tags and structuring tags into a hierarchy, i.e., into a taxonomy.

4.1.2 Control term abstraction thanks to the *generalization vector* ϑ

The generalization process makes use of taxonomies for abstracting event descriptors. This process is made possible thanks to the partial order that exists in IS-A hierarchies. Indeed, we assume that each descriptive domain D_A, on which event descriptors are defined, is structured into a taxonomy $H_A \in \mathcal{H} = \bigcup_{\mathcal{A}} H_A$, defining a generalization-specialization relationship between descriptors of D_A. The taxonomy H_A provides a partial ordering $\prec_A$ over D_A and is rooted by the special descriptor "any_A", i.e., $\forall d \in D_A,\ d \prec_A$ "any_A". For convenience, we assume in the following that the descriptor "any_A" belongs to descriptive domain D_A. An example of taxonomy defined for the *topic* descriptive domain in given in Figure 2.3.

The partial ordering $\prec_A$ on D_A defines a cover relation $<_A$ that corresponds to direct links between items in the taxonomy. Hence, we have $\forall (x, y) \in D_A^2,\ x \prec_A y \Rightarrow \exists (y_1, y_2, \ldots, y_k) \in D_A^k$ such that $x <_A y_1 <_A \ldots <_A y_k <_A y$. The length of the path from x to y is $\ell(x, y) = k + 1$. In other words, we need $k + 1$ *generalizations* to reach y from x in H_A. For example, given the *topic* domain taxonomy in Figure 2.3, it takes 2 generalizations to reach the concept "Queries" from the concept "Top-k query". So, thanks to this relation, we can define a containment relation $\sqsubseteq$ over subsets of Ω, i.e., $\mathcal{P}(\Omega)$, as follows:

$$\forall (x, y) \in \mathcal{P}(\Omega)^2, \quad (x \sqsubseteq y) \iff \\ \big(\forall i \in x,\ \exists i' \in y,\ \exists A \in \mathcal{A},\ (i \prec_A i') \vee (i = i')\big) \tag{2.1}$$

For the purpose of generalization, we need to replace event descriptors with their antecedents in the taxonomies, i.e., upper terms of the taxonomies. We call *generalization vector* on Ω, denoted $\vartheta \in \mathbb{N}^i$, a list of integer values. ϑ defines the number of generalizations to perform for each descriptive domain in $\mathcal{A}$. We denote by $\vartheta[A]$ the generalization level for the domain A. Equipped with this generalization vector ϑ, we are now able to define a restriction $\sqsubseteq_{\downarrow\vartheta}$ of the containment relation above:

$$\forall (x, y) \in \mathcal{P}(\Omega)^2,\ (x \sqsubseteq_{\downarrow\vartheta} y) \iff \forall i \in x, \exists i' \in y, \exists A \in \mathcal{A}, \\ \Big((i \prec_A i') \wedge \big((\ell(i, i') = \vartheta[A]) \vee ((\ell(i, i') < \vartheta[A]) \wedge (i' = \text{"any\_A"}))\big)\Big) \tag{2.2}$$

[4] http://delicious.com
[5] http://www.flickr.com

From the user's view point, ϑ represents the semantic accuracy he desires for each descriptive domain. If he is interested in the minute details of a specific domain, such as a paper's *topic* in our illustrative example, he can set ϑ to a low value, e.g., $\vartheta[topic] = 0$ or $\vartheta[topic] = 1$. The rationale of these parameters are the following: In the case $\vartheta[topic] = 0$, descriptors for the *topic* domain are not generalized and kept in their original form. This is the most accurate form of the data. This parameter should be set when one is interested in events' minute details on this particular descriptive domain. In the case $\vartheta[topic] = 1$, some approximation is tolerated: One is interested in events on this particular descriptive domain but only requires the general *trend*.

Also, an additional benefit of fixing a small number of generalizations is to allow the summarization approach to handle outliers. Indeed, in some situations, descriptors can correspond to very specialized vocabulary in specific domains. In conventional approaches these descriptors might simply be discarded. However, from taxonomy point of view, these descriptors might be *close* to other more well known or more commonly used terms. Hence, generalizing such descriptors a small number of times has the nice property of capturing those descriptors at a higher level of taxonomy and allow their comparison to more commonly used descriptors.

On the other hand, the user can set ϑ to higher values for a more abstract representation of events on this particular descriptive domain. In the most extreme case, one can set $\vartheta[topic] = \infty$. The rationale of this setting is that the user is simply not interested in the *topic* descriptive domain and any value is acceptable.

In a nutshell, the generalization vector ϑ introduced here is a tool allowing the user to control the level of generalization of event descriptors on each descriptive domain. This tool has two benefits: (i) it allows the user to express his preferences for abstracting events and (ii) it allows the user to control the trade off between personal conception of the data representation vs. lost of semantic content and summary informativeness. One might argue that defining integer values for the generalization vector is a tedious task. We believe this shortcoming can be addressed and the task simplified for the user. Indeed, the integer values that represent the level of generalization for each descriptive domain can be mapped to a set of intuitive categorical concepts such as {"Very precise", "Precise", "Fuzzy", "Gross"}. For instance, "Very precise" could be mapped to $\vartheta = \langle 0 \rangle$, "Precise" could be mapped to $\vartheta = \langle 1 \rangle$, etc..

4.1.3 Generalization process

Equipped with a set of taxonomies, i.e., one taxonomy for each descriptive domain, and a user-defined generalization vector ϑ, we can now define the generalization operator that achieves generalization on time sequences of events. This generalization operator is denoted φ_ϑ and formally defined in Definition 2.1. In this definition, an event e'_i in $\varphi_\vartheta(s)$ is called a *generalized event*, and, a time sequence of events $\varphi_\vartheta(s)$ is equivalently called a *time sequence of generalized events*, a *generalized time sequence of events*, a *generalized time sequence* or a *generalized sequence*.

Definition 2.1 (Generalization operator)
Given s a time sequence of events in $\mathcal{S}(\Omega)$, $s = \{e_1, \ldots, e_n\}$ with $e_i = (x_i, t_i)$, and a generalization vector ϑ, we define a parametric generalization function φ_ϑ that operates on s as follows:

$$\begin{array}{rccl} \varphi_\vartheta : & \mathcal{S}(\Omega) & \longrightarrow & \mathcal{S}(\Omega) \\ & s & \longmapsto & \varphi_\vartheta(s)=\{e'_1, \ldots, e'_n\},\ e'_i=(x'_i, t_i),\ \textit{such that}\ \forall i \in \{1..n\},\ x_i \sqsubseteq_{\downarrow\vartheta} x'_i \end{array}$$

As an example of generalization, we can operate φ_ϑ on our illustrative example using the *topic* domain taxonomy given in Figure 2.3. The generalized sequence obtained by generalizing Table 2.1 with $\vartheta = \langle 1 \rangle$ is given in Table 2.2. This operation has allowed to reduce the variability of event descriptors in Table 2.1 from 6 descriptors to 2 descriptors. By doing so, some event descriptors have become similar. For instance, in events (x'_3,AUG06) and (x'_4,AUG06), x'_3 and x'_4 are identical, then, events (x'_3,AUG06) and (x'_4,AUG06) can be considered for grouping.

In addition, one should note that the $\sqsubseteq\downarrow_\vartheta$ relation also allows to reduce itemsets' cardinality. Indeed, *Datastreams* and *Top-k query* both generalize into *QO*. As a result, in N. Koudas's time sequence, the event {Datastreams, Top-k query} is generalized into {QO} having one single event descriptor.

Author	**Event descriptors**	**Time**
N. Koudas	x'_1={QO, DM}	JUN05
	x'_2={QO}	AUG06
	x'_3={QO}	AUG06
	x'_4={QO}	SEP06
	x'_5={QO}	SEP06
	x'_6={DM}	SEP07

Table 2.2: Generalized events with $\vartheta = \langle 1 \rangle$

Once input time sequences of events have undergone the generalization process, the output time sequence of generalized events may present generalized events that are close on the timeline and have similar descriptors, e.g., (x'_3,AUG06) and (x'_4,AUG06). These generalized events are good candidates to be gathered together to form a group of (similar) generalized events. This group represents events that relate to a same topic, e.g., {QO}. This operation is the purpose of the grouping phase.

4.2 Grouping phase

The grouping phase is responsible for gathering *similar* generalized events that are *close* on the timeline. This phase relies on three concepts: (i) second-order time sequence as defined in Definition 1.6 in Chapter 1, (ii) event *similarity* and (iii) *temporal locality* as defined in Section 3.2. The notion of second-order time sequence is a convenient tool for producing a sequence of grouped events and the notion of temporal locality will be useful for deciding which events are eligible for grouping.

4.2.1 Temporal locality and temporal parameter w

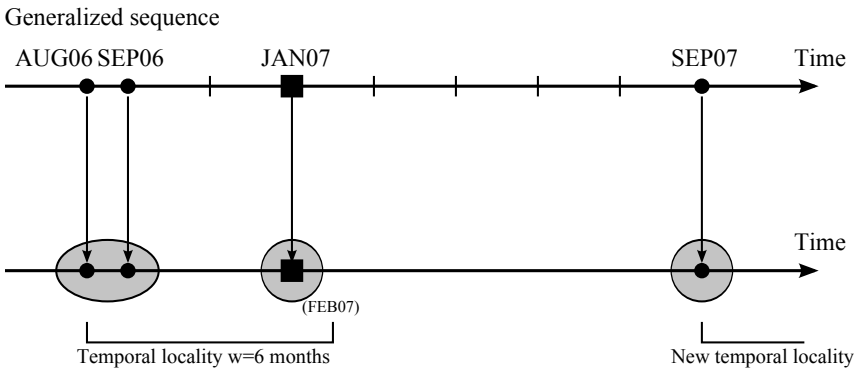
The most interesting concept on which the grouping process relies is *temporal locality*. We assume that events that occur close enough on the timeline have high probability of relating to a same topic. Therefore, the grouping process is controlled by the user thanks to a temporal locality parameter denoted w. Intuitively, parameter w is an estimation or a measurement of the temporal locality phenomenon at the time of occurrence of each generalized event. In other words, at a given instant t on the timeline, parameter w indicates the range within which generalized events are eligible for grouping to describe a same phenomenon or topic.

More formally, the temporal locality is measured as the time difference d_T, on a temporal scale T, between an incoming event and previously considered events in the output sequence of grouped events. This output sequence of grouped events is a second-order

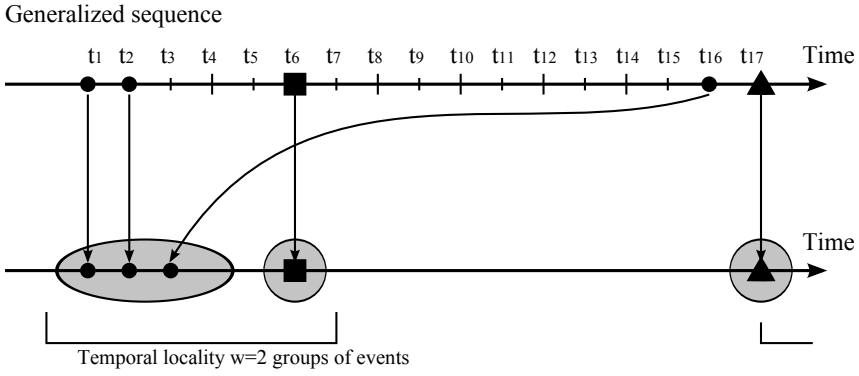
time sequence that represents the groupings of generalized events operated. This output sequence is also equivalently called a *time sequence of groups* or *sequence of groups*. Events in a sequence of groups are themselves a grouping of generalized events; Each such event is called a *group* for short.

The temporal scale T may be defined directly through timestamps if there exist explicit timestamps or through integer values if there are no timestamps, or if timestamps are not relevant, or not of interest. During the grouping process, an incoming generalized event can only be compared to previous groups within a *distance* w, i.e., $d_T \leq w$, in the sequence of groups. In fact, the temporal locality parameter w acts as a *sliding window* on the sequence of groups and its value corresponds to the user's own estimation of the temporal locality of events – then, note that w will be equivalently called *temporal window* or *temporal parameter* hereafter. Said otherwise, w can be understood as the loss of temporal accuracy the user is ready to tolerate for grouping generalized events.

This temporal window can be defined as a duration, e.g., w equals six months, or as a number of groups, e.g., w equals two groups, independently from the way temporal scale T is defined. This means that if the temporal scale T is defined through timestamps, the temporal window can be defined either as a duration or as a number of groups. However, reversely, if the temporal scale T is defined through integer values, the temporal window can only be defined as a number of groups. Indeed, in this case the notion of duration does not make sense.



(a) Temporal scale defined through timestamps and temporal locality as a duration



(b) Temporal scale defined through integer values and temporal locality as a number of groups

Figure 2.5: Example of time scales and temporal locality

Defining w as a number of groups is useful in particular when considering bursty sequences, i.e., sequences where the arrival rate of events is uneven and potentially high. The downside of this approach is the potential grouping of events distant on the timeline. This limitation could be solved by defining w as both a duration and a constraint on the number of groups. Since certain domains, e.g., networking or Finance, give more importance to the freshness of most recent information, we choose in the TSaR approach to express the temporal scale as a number of groups in order to handle bursts of events.

Let us illustrate the choice of w with an example. Figure 2.5(a) presents the grouping of events *circle* and *square* when the temporal scale is defined through timestamps and the temporal locality chosen as a duration of 6 months. Figure 2.5(b) presents the grouping of events *circle*, *square* and *triangle* when the temporal scale is defined through integer values and the temporal locality is chosen as a number of groups: The number of groups is set to two groups and is chosen in accordance to the number of groups formed in Figure 2.5(a). The interesting observation here is the difference in the number of groups obtained in total. In Figure 2.5(a), event *circle* at t=SEP07 generates a new group; In Figure 2.5(b), event *circle* at t_{16} is grouped with events *circle* at t_1 and t_2. This observation illustrates the issue of potentially grouping events that are distant on the timeline. In this illustrative example, the best choice would have been defining the temporal locality as a duration, since events are not *bursty*, or choosing a smaller parameter w, i.e., w equals one group. In the case w equals one group, the exact same groupings as in Figure 2.5(a) would have been achieved.

In short, the temporal locality parameter w allows users to control the accuracy of the summary from temporal view point. Examples in Figure 2.5 have shown that the choice of w as a duration or number of groups and their value directly impacts the groupings achieved. Our indication to users is that smaller temporal window sizes will allow to capture more details on the chronology of events. Larger window sizes will gather events on long periods of time. This is useful for instance for retrieving the very general trend of the chronology of events.

4.2.2 Grouping process

Equipped with the temporal locality parameter w, we can now define the grouping operator that gathers events from the generalized sequence output by the generalization process that are similar and close on the timeline. This grouping operator is denoted ψ_w and formally defined in Definition 2.2. Intuitively, ψ_w takes as input a time sequence of events, either generalized or not, and outputs a second-order time sequence of events. Events in this second-order time sequence of events are regular time sequences themselves and correspond to similar events within the temporal locality defined by w. In this definition, an event (Y_i, t'_i) in $\overline{s}$ is called a *group*, and, a second-order time sequence $\overline{s}$ is equivalently called a *time sequence of groups*, a *grouped sequence*, a *time sequence of groups* or a *sequence of groups*.

Definition 2.2 (Grouping operator)
Given s a time sequence of events in $\mathcal{S}(\Omega)$, $s = \{e_1, \ldots, e_p\}$, and a temporal locality

parameter w, we define a parametric grouping function ψ_w that operates as follows:

$$\begin{array}{rccl} \psi_w: & \mathcal{S}(\Omega) & \longrightarrow & \mathcal{S}^2(\Omega) \\ & s & \longmapsto & \psi_w(s) = \{(Y_1, t'_1), \ldots, (Y_n, t'_n)\} \\ & & & \textit{where } Y_i = \{e_{i,1}, \ldots, e_{i,m}\},\ e_{i,j} = (x_{i,j}, t_{i,j}) \textit{ and } t'_i = t_{i,1},\ \textit{such that:} \end{array}$$

(Part) $\forall k \in [w, p],\ \bigcup_{k-w \leq i \leq k} Y_i \subseteq S$ *and* $Y_i \cap Y_j = \emptyset$ *when* $1 \leq i < j \leq n,\ j - i \leq w$

(Cont) $(x_{i,q} \sqsubseteq x_{i,r}),\ 1 \leq r < q \leq m_i,\ 1 \leq i \leq n$

(TLoc) *if w is defined as duration,* $\big(d_T(t_{i,q}, t'_i) \leq w\big),\ 1 \leq i \leq,\ 1 \leq q \leq m_i$
if w is defined as an integer, given $Y = \{Y_i, \ldots, Y_{i+w-1}\}$, *with* $1 \leq i \leq n - w$,
then $\forall (j,k)$ *with* $i \leq j < k \leq i + w - 1,\ \bigcup_{e_u \in Y_j} x_u \cap \bigcup_{e_v \in Y_k} x_v = \emptyset$

(Max) $\forall x_q \in Y_i,\ \forall x_r \in Y_j,\ 1 \leq i < j \leq n,\ (x_q \sqsubseteq x_r) \Longrightarrow \big(d_T(t_r, t'_i) > w\big)$,
$1 \leq q \leq m_i,\ 1 \leq r \leq m_j$

Note in Figure 2.6(a) that even though events ({QO,DM},JUN05) and ({QO}, AUG06) have a common descriptor, i.e., "QO", they are not grouped together. This fact directly results from our definition of *similarity* between two events, i.e., we require *similar* events to have identical sets of descriptors. From practical view point, suppose event (x'_1,AUG05) in Table 2.2 has been processed and produces (Y_1,AUG06) with $Y_1 = \{(x'_1, AUG06)\}$. When generalized event (x'_2,AUG06) is considered, since $w = 1$ the grouping process can only consider the 1 last previously considered group within w, i.e., Y_1. Since $x'_1 \neq x'_2$, events (x'_1,AUG05) and (x'_2,AUG06) are considered non similar. So, the temporal window w *slides* on and event (x'_2,AUG06) initiates a new group in w, i.e., Y_2. Since, the temporal locality parameter w controls how well the chronology of groups should be observed, we can clearly note that when $w = 1$ only similar and *contiguous* events on the timeline are eligible for grouping. We will discuss in more details this notion of contiguity in Chapter 3.

Now, suppose that generalized event (x'_6,SEP07) is altered and the event is described by itemset x''_6={QO,DM}. Suppose we define the temporal locality parameter as an integer value set to $w = 2$. Figure 2.6(b) gives the grouped sequence obtained in this case. This example highlights the loss of chronology of grouped events when the temporal locality parameter w is increased: Event (x''_6,SEP07) normally occurs after events (x'_2,AUG06), ..., (x'_5,SEP06) but is grouped with event (x'_1,JUN05) when $w = 2$. Grouping events in this manner has the nice property of reducing the overall number of groups in the output sequence but requires to trade off the temporal accuracy of the summary, i.e., the precision of the chronology of events.

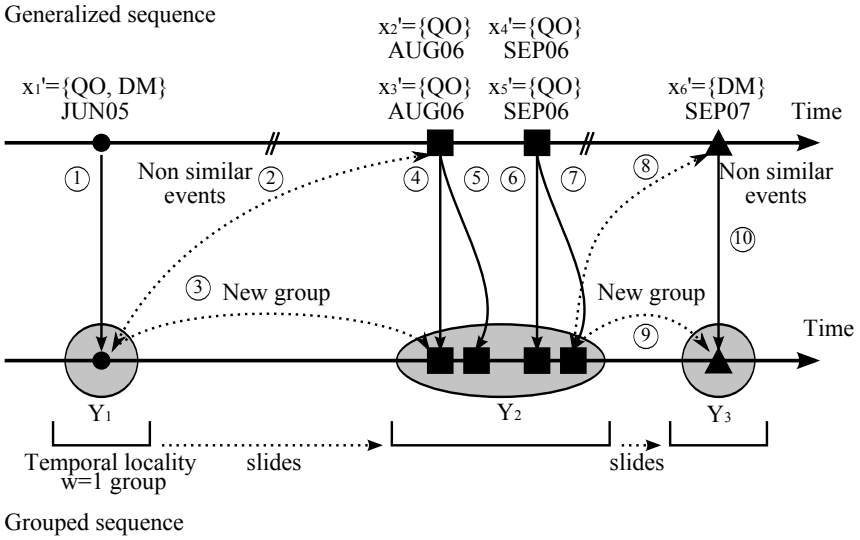
Let us give an illustration of grouping on our running example before we detail the properties *(Part)*, *(Cont)*, *(TLoc)* and *(Max)*. Suppose we are interested in the details of the chronology of events and decide to express the temporal locality as an integer value set to $w = 1$. Table 2.3 presents the expected output sequence of groups when operating ψ_w on Table 2.2 with $w = 1$ and Figure 2.6(a) shows how the temporal window w *slides* on the timeline.

Author	**Groups**	**Time**
N. Koudas	Y_1={({QO,DM},JUN05)}	JUN05
	Y_2={({QO},AUG06), ({QO},AUG06), ({QO},SEP06), ({QO},SEP06)}	AUG06
	Y_3={({DM},SEP07)}	SEP07

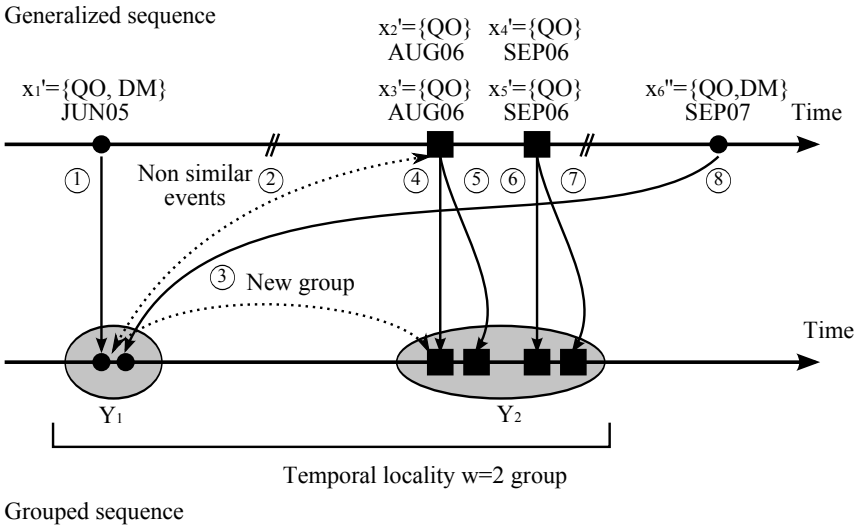
Table 2.3: Grouped events with $w = 1$

Having illustrated the mechanism of the grouping operator, we can detail the properties of the ψ_w operator:

• Property *(Part)* (standing for *Partitioning*) ensures that the support multi-set of w-contiguous groups in $\psi_w(s)$, e.g., $\{(Y_1, t'_1), \ldots, (Y_w, t'_w)\}$, $\{(Y_2, t'_2), \ldots, (Y_{w+1}, t'_{w+1})\}$, etc., is a non-overlapping part of S. This is a direct consequence of grouping events that relate to a same topic within a same temporal locality. In other words, the *(Part)* property ensures that for any sliding window, having w duration or w number of groups, taken in the sequence of groups $\psi_w(s)$, e.g., $Y = \{(Y_1, t'_1), \ldots, (Y_w, t'_w)\}$, for all $((Y_i, t'_i), (Y_j, t'_j)) \in Y^2$, $i \neq j$, $\bigcup_{e_u \in Y_i} x_u \cap \bigcup_{e_v \in Y_j} x_v = \emptyset$. For instance, in Table 2.3, since $w = 1$, any window having 1 event, i.e., $W = \{(Y_i, t_i)\}$, complies to this property. Now suppose Table 2.3 was obtained from Table 2.2 with $w = 2$ (this statement is true). Then, let us select a sequence having 2 contiguous events from $\psi_w(s)$, e.g., $Y = \{(Y_1, t'_1), (Y_2, t'_2)\}$. Property *(Part)* holds since, the set of descriptors {QO,DM} is contained in an event in Y_1 and not in any events in Y_2.



(a) Grouping of generalized sequence in Table 2.2 with $w = 1$



(b) Alternate grouping with $w = 2$

Figure 2.6: Example of groupings

• Property *(Cont)* (standing for *Containment*) gives a containment condition on events of every time sequence in $\psi_w(s)$. Given a time sequence Y_i in $\psi_w(s)$, all events $e_{i,j} \in Y_i$

are comparable w.r.t. the containment relation $\sqsubseteq$ and $e_{i,1}$ is the greatest event, i.e., $x_{i,1}$ contains all other event descriptors of events in Y_i. For instance, in Table 2.3, we have $x_{2,1}$={QO}, $x_{2,2}$={QO}, $x_{2,3}$={QO} and $x_{2,4}$={QO}, and the property *(Cont)* is verified since $x_{2,1} \sqsubseteq x_{2,2} \sqsubseteq x_{2,3} \sqsubseteq x_{2,4}$={QO}.

• Property *(TLoc)* (standing for *Temporal locality*) defines a temporal locality constraint on events that are grouped into a same time sequence Y_i in $\psi_w(s)$. When w is expressed as a duration, property *(TLoc)* enquires that Y_i only groups events $e_{i,j}$ that are within a distance d_T from timestamp $t'_i = min(Y_i[T])$, i.e., $d_T(t_{i,j}, t'_i) \leq w$. When w is expressed as a integer value, any w consecutive groups in $\psi_w(s)$ are mutually non similar.

• Property *(Max)* guaranties that the joint conditions *(Cont)* and *(TLoc)* are maximally satisfied.

When we refer to the definition of a time sequence summary in Definition 1.7 given in Chapter 1, the second-order time sequence output by the grouping operator ψ_w can be understood as the extension of the summary, i.e., s^2_C in the notation of Definition 1.7. The intention of the summary, and thus the reduced version of the input time sequence, is obtained thanks to the concept formation phase as presented in the following section.

4.3 Concept formation phase

Concept formation, also known as *concept learning* or *category learning* in cognitive science, refers to the task of *"searching for and listing of attributes that can be used to distinguish exemplars from non exemplars of various categories"*, Bruner et al. [BGA65]. Concepts are mental representations that help one to classify objects, events or ideas that have a common set of features. For example, suppose we are given two sets of words representing two categories D_1={fish, whale, shark, dolphin} and D_2={lion, bear, cougar, fox}. If an individual is asked to classify the object "cat" into one of the two categories D_1 and D_2, most likely he/she would classify "cat" into D_2. This process results from the fact a human needs to represent D_1 and D_2 by a *representative concept* d_1 and d_2, respectively, that exhibits features common to all objects in D_1 and D_2, respectively. For instance, d_1 could be an aquatic animal, that has a tail and no legs, and, d_2 could be a furry 4-legged terrestrial animal.

In 1980, Michalski [Mic80, MS80, Mic93] introduced the novel *conceptual clustering* paradigm in the area of Machine Learning research. In contrast with the traditional data clustering paradigm, a *concept description* is constructed for each class generated by the clustering method. Conceptual clustering has since attracted much research [Fis86, Fis87, GLF89, Gen89, TL91, CR04, JHC00, TB01, SPRM05]. In particular, Gennari et al. [Gen89] defined the tasks as:

- *Given:* a sequential representation of objects and their associated descriptions;
- *Find:* clusterings that group these objects into concepts;
- *Find:* a summary description for each concept;
- *Find:* a hierarchical organization for these concepts;

From this prospect, the TSAR summarization approach can be understood as a form of conceptual clustering since events in a time sequence of events are *grouped*, or *clustered* if we use clustering terminology, based on the similarity of their content and proximity on the timeline. Therefore, the concept formation phase is responsible for representing each

group obtained from the grouping process by one single *concept* or *representative event*. Hence, this phase is responsible for generating the time sequence of events $s^\star_M$, i.e., the intention of the summary as defined in Definition 1.7 given in Chapter 1, from the sequence of groups s^2_C.

Concept formation can be achieved in various ways. For instance, in traditional data clustering, the centroid of a cluster can be chosen to represent the cluster. This cluster centroid is usually computed as the *mean* of all objects in the cluster. This approach usually fits numerical attributes but does not fit categorical attributes since aggregation functions such as *average* can not be directly applied to categorical values. For instance, what is the *average* between categorical values "few" and "plenty"? Hence, in the case of categorical values, other distance measures could be leveraged, e.g., semantic measures [Bud99], or aggregation functions such as the *Most Specific Common Subsumer* (MSCS) (also known as the *Lowest Super-Ordinate* (LSO)). We will discuss these measures in details in Chapter 3.

Probabilistic methods can also be used to represent a cluster of objects. Cobweb [Fis87] must be one of the most well known of such methods. Indeed, Cobweb incrementally builds a classification tree where each node represents a class and is label by a probabilistic concept. This probabilistic concept summarizes the attribute-value distribution of objects classified under the node.

In TSAR, we gathered in the grouping phase events that have identical sets of event descriptors. Consequently, we do not need to go through the costly task of computing distances between events in a group to generate a representative event for the group. Hence, this concept formation phase is straightforward. Here, concept formation is achieved thanks to the projection operator π defined in Definition 2.3. Intuitively, π represents each group Y_i in the sequence of groups s^2_C by a representative event denoted $e^\star_i = (x^\star_i, t'_i)$: The representative event $e^\star_i$ is actually contained in Y_i and chosen as the oldest event in the group. Consequently, π produces from s^2_C a regular time sequence of events denoted $s^\star_M$ where $s^\star_M = \pi(s^2_C)$ is the intention of the summary. In this definition, an event $e^\star_i$ in $s^\star$ is equivalently called a *representative event* or a *representative concept*, and, the time sequence of events $s^\star$ is equivalently called a *representative time sequence of events*, a *representative time sequence*, a *representative sequence* or a *summarized time sequence*.

Definition 2.3 (Projection of a second-order time sequence)
Given $\overline{s}$ a second-order time sequence of events in $\mathcal{S}^2(\Omega)$, $\overline{s} = \{(Y_1, t'_1), \ldots, (Y_n, t'_n)\}$, where $Y_i = \{(x_{i,1}, t_{i,1}), \ldots, (x_{i,m_i}, t_{i,m_i})\}$, the projection operator performs as follows:

$$\begin{array}{rccl} \pi: & \mathcal{S}^2(\Omega) & \longrightarrow & \mathcal{S}(\Omega) \\ & \overline{s} & \longmapsto & \pi(\overline{s}) = \{e^\star_1, \ldots, e^\star_n\},\ e^\star_i = (x^\star_i, t^\star_i)\ \textit{and}\ x^\star_i = x_{i,1}\ \textit{and}\ t^\star_i = t'_i = t_{i,1} \end{array}$$

Let us illustrate the projection operator with our example. The projection of N. Koudas's grouped sequence in Table 2.3 occurs as follows:
$s^\star$=π({(Y_1,JUN05),(Y_2,AUG06),(Y_3,SEP07)}={($x^\star_1$,JUN05),($x^\star_2$,AUG06),($x^\star_3$,SEP07)}). The time sequence obtained is given in Table 2.4. One can note that, in fine, the projection operator is responsible for effectively reducing the numerosity of events in the representative sequence w.r.t. the number of events in the original time sequence. In this example, the number of events in the representative sequence is reduced from 6 events to 3 events. Hence, the *compression ratio* achieved by TSAR on the input time sequence is 60% (the compression ratio achieved is not 50% due to the fact 100% compression ratio corresponds to a summary containing single event - see Definition 2.7). The compression ratio measure will be formalized in details in Section 6.2.3.

Author	Event descriptor	Time
N. Koudas	$x_1^\star$={QO,DM}	JUN05
	$x_2^\star$={QO}	AUG06
	$x_3^\star$={DM}	SEP07

Table 2.4: Projection of the second-order time sequence in Table 2.3

4.4 TSaR summarization process

TSaR achieves summarization by associating the three operators presented in previous sections, namely, (i) the generalization operator, (ii) the grouping operator and (iii) the projection operator, φ_ϑ, ψ_w and π, respectively. The summarization function is formally defined in Definition 2.4.

Definition 2.4 (*T*ime *S*equence Summ*aR*ization (TSaR) operator)
Given s a time sequence of events in $\mathcal{S}(\Omega)$, a user defined generalization vector ϑ and a user defined temporal locality parameter w, the summary of s is obtained by combining a generalization φ_ϑ, followed by a grouping ψ_w and a projection π:

$$\begin{array}{rccl} \chi_{\vartheta,w}: & \mathcal{S}(\Omega) & \longrightarrow & \mathcal{S}^2(\Omega) \times \mathcal{S}(\Omega) \\ & s & \longmapsto & \chi_{\vartheta,w}(s) = (s_C^2, s_M^\star) \text{ where } s_C^2 = \psi_w \circ \varphi_\vartheta(s) \text{ and } s_M^\star = \pi(s_C^2). \end{array}$$

The association of the generalization and grouping function, φ_ϑ and ψ_w respectively, outputs the extension of the summary, i.e., s_C^2. The extension of the summary s_C^2 satisfies the conditions of the generalization-grouping process. The *(Cont)* property of ψ_w is then enforced by the generalization process φ_ϑ of events in s. Note that every element in the reduced form of the summarized time sequence, i.e., $s_M^\star$, is an element of φ_ϑ: $s_M^\star$ is a representative subsequence of the generalized sequence of s, i.e., $\varphi_\vartheta(s)$.

The reduced form of the summary, i.e., its intention $s_M^\star$, is a time sequence obtained by forming concepts from groups in s_C^2 thanks to the projection operator π. For this reason, one can note that loss of semantic content only occurs during the generalization phase, i.e., through event descriptors rewriting. The operations performed by the grouping and the concept formation operators have no incidence on the semantic content. This characteristic is given in Property 2.1. This property will be most useful for characterizing sequential patterns that can be extracted by any conventional Sequential Pattern Mining algorithm on TSaR summaries. We will detail this application in Chapter 4.

Property 2.1 (Semantic preservation)
Given s a time sequence of events in $\mathcal{S}(\Omega)$, $s = \{e_1, \ldots, e_n\}$ with $e_i = (x_i, t_i)$, we denote by s' the generalization of s, i.e., $s' = \varphi_\vartheta(s) = \{e'_1, \ldots, e'_n\}$ with $e'_i = (x'_i, t_i)$ and we denote by $s_M^\star$ the intention of s's summary, i.e., $s_M^\star = \{e_1^\star, \ldots, e_m^\star\}$ with $e_i^\star = (x_i^\star, t_i^\star)$. The following property is true:

$$\forall e'_i = (x'_i, t_i) \in s',\ \exists e_j^\star = (x_j^\star, t_j^\star) \in s_M^\star,\ x'_i = x_j^\star \text{ with } t_{j,1} = t_j^\star \text{ and } t_j^\star \leq t_i \quad (2.3)$$

$$\text{reversely, } \forall e_j^\star \in s_M^\star,\ \exists Y'_j = \{e'_{i,k} \in s',\ x'_{i,k} = x_j^\star \text{ and } t_j^\star = t'_{i,1}\} \quad (2.4)$$

Proof 1 (Proof of Property 2.1)
By definition, $s_M^\star = \pi \circ \psi_w \circ \varphi_\vartheta(s)$ or equivalently, $s_M^\star = \pi \circ \psi_w(s') = \pi(s_C^2)$ where $s_C^2 = \{(Y_1, t'_1), \ldots, (Y_m, t'_m)\}$, $Y_i = \{e'_{i,1}, \ldots, e'_{i,k}\}$ and $e'_{i,j} \in s'$. Note that all generalized events $e'_{i,j}$ in Y_i have the same set of event descriptors, i.e., $x'_i = x'_{i,1} = \ldots = x'_{i,k}$. Since $\pi(s_C^2) = s_M^\star = \{e_1^\star, \ldots, e_m^\star\}$ with $e_j^\star = (x_j^\star, t_j^\star) = (\pi(Y_j), t'_j)$, by the definition of the π operator, we have: (i) $t_j^\star = t'_j = t_{j,1}$ = and (ii) $x_j^\star = x'_j = x'_{j,1} = \ldots = x'_{j,k}$. We proved Property 2.1. □

4.5 TSAR properties w.r.t. a summary's desirable properties

From a practical view point, since $s^{\star}_{M}$ is itself a regular time sequence of events, i.e., as defined in Definition 1.5 in Chapter 1, and is the most concise format, users and applications are only given the intentional form $s^{\star}_{M}$ of s's time sequence summary $\chi_{\vartheta,w}(s)$. Additionally, being a regular time sequence of events, $s^{\star}_{M}$ complies to the substitution principle and can seamlessly replace s in any application that operates directly on s. Thus, $s^{\star}_{M}$ is the most useful form from application view point. In the rest of this dissertation, we will interchangeably use the term *summary* to designate the intention $s^{\star}_{M}$ of a summary $(s^{2}_{C}, s^{\star}_{M})$.

The representative sequence given as example in Table 2.4 highlights the compression capability of the TSAR approach. We already observed the reduction in terms of number of events (from 6 to 3 events). Also, one should notice the reduction of variability of event descriptors (from 6 to 2). Here, TSAR achieves the dual goal of numerosity reduction and domain reduction while preserving comprehensive information on events. Also, the temporal locality parameter w allows to control the numerosity reduction factor by limiting the scope or neighborhood in which the grouping process selects events for grouping. By doing so, we showed that this temporal locality parameter allows to control how well the overall chronology of events in the summarized time sequence is preserved (see Figure 2.6). In a nutshell, these compression effects obtained thanks to the two user defined parameters control the trade-off between resp. semantic accuracy vs. standardization and temporal (or chronology) accuracy vs. compression.

In the following section, we show that TSAR can be implemented as an incremental algorithm that has linear computational complexity and limited memory footprint. We will also empirically demonstrate the *computational scalability* property in our experimental study in Section 6. For all these reasons, we can argue that TSAR presents all the desirable properties expected from a time sequence summary, as given in Definition 1.8 in Chapter 1.

5 TSAR algorithm

In this section, we present the implementation of the TSAR summarization process. We propose here an incremental algorithm that has linear computational complexity and low memory footprint. In this algorithm, we assume a time sequence of events can be contained in-memory.

From an operational view point, our implementation of TSAR is given in Algorithm 1. This algorithm considers as input a time sequence of events and outputs another time sequence of events after abstracting event descriptors through generalization and grouping similar events within a certain temporal locality. The algorithm is parameterized by a set of taxonomies provided in an XML format, a generalization vector provided as an array of integers and a temporal locality parameter provided as an integer value, i.e., a number of groups. Events are considered in ascending order of timestamp, one at a time. The incremental algorithm generalizes each incoming event into a generalized event. This generalized event is then compared to previously grouped events within a temporal distance d_T equal to w.

More precisely, assume the input time sequence of events is $s = \{e_1, \ldots, e_n\}$. Assume the incoming event is $e_i = (x_i, t_i)$ and previous $i-1$ events have already been summarized in $s^{\star}_{M}$, where $s^{\star}_{M}$ is in the following state:
$s^{\star}_{M} = \chi_{\vartheta,w}(\{e_1, \ldots, e_{i-1}\}) = \{(Y_1, t'_1), \ldots, (Y_j, t'_j)\}$. The algorithm computes $\chi_{\vartheta,w}(\{e_1, \ldots, e_{i-1}, e_i\})$ with only a local update to $s^{\star}_{M}$, i.e., changes are only made within the last w groups in $s^{\star}_{M}$. So first, incoming event $e_i = (x_i, t_i)$ is generalized into generalized event

$e'_i = (x'_i, t_i)$ (Line 5).

Assume the set of groups that are included within the temporal locality window w is denoted $W = \{(Y_j, t'_j)\}$, $|W| \leq w$. Then, TSAR checks if x'_i is contained in a group $(Y, t') \in W$ (Line 6). e'_i is either incorporated into the group (Y, t') if it satisfies the *(Cont)* condition (Line 6 to Line 16), or it initializes a new group (Y_{j+w}, t'_{j+w}) in W with $t'_{j+w} = t_i$ (Line 9 to Line 15). Once all input events are processed, the last w groups contained in W are projected and added to the output summary $s^\star_M$ (Line 18 to Line 21). The final output summary is then returned and/or stored in a database.

Algorithm 1 TSAR's pseudo-code

1. INPUTS
   s: time sequence of events
   $\mathcal{H}$: taxonomies
   ϑ: generalization vector
   w: temporal locality
2. OUTPUT: $s^\star_M$: Summary
3. LOCAL: W: FIFO list containing the w last groups stored in $s^\star_M$
4. **for all** incoming event $e_i = (x_i, t_i) \in s$ **do**
5. $e'_i = (x'_i, t_i) \leftarrow \varphi_\vartheta(e_i)$ {// Generalization using ϑ and $\mathcal{H}$}
6. **if** $(\exists\ (Y, t') \in W,$ such that $\exists e'_u \in Y,$ and $x'_i = x'_u)$ **then**
7. $Y \leftarrow Y \cup \{e'_i\}$ {// Grouping}
8. **else**
9. **if** ($|W| = w$, i.e., W is full) **then**
10. Pop W's 1^{st} group (Y_j, t'_j) out of W
11. **end if**
12. $Y_{j+w} \leftarrow \{e'_i\}$
13. $t'_{j+w} \leftarrow t_i$
14. $W \leftarrow W \cup \{(Y_{j+w}, t'_{j+w})\}$
15. $s^\star_M \leftarrow s^\star_M \cup \{(\pi(Y_j), t'_j)\}$ {// Update the summary intention $s^\star_M$}
16. **end if**
17. **end for**
18. **while** $(W \neq \emptyset)$ **do**
19. Pop W's 1^{st} group (Y_j, t'_j) out of W
20. $s^\star_M \leftarrow s^\star_M \cup \{(\pi(Y_j), t'_j)\}$
21. **end while**
22. **return** $s^\star_M$

5.1 Algorithmic complexity

TSAR performs generalization, grouping and concept formation on the fly for each incoming event. The process has an algorithmic complexity linear with the number of events in the input time sequence of events. This computational cost can be refined by taking into account the computational cost induced by the generalization and grouping phases, i.e., the cost induced by generalization and for looking up for grouping candidates within a temporal locality w. Hence, the processing cost is weighted by a constant cost $c = a * b$:

- a is the cost for generalizing an event's set of descriptors and mainly depends on the number of taxonomies and their size.
- b is the cost to scan the finite list of groups previously summarized in W. This cost is

however negligible since the temporal locality window parameter w used is in general small, e.g., mostly $w \leq 25$ in our experiments.

Regarding the generalization cost, a is constrained. Indeed, taxonomies are input to the TSAR algorithm in the form of XML trees, thus, generalizing a descriptor is equivalent to looking up the descriptor in the trees and retrieving its antecedent. This can be a costly task since there does not exist any ordering in the XML tree. A first approach to mitigate this effect is to build a trie [Fre60], or prefix tree, to index the position of descriptors in the XML tree. Doing so induces a cost a' that corresponds to the cost for building the trie and allows to reduce a to $O(gm)$ where $o(m)$ is the search time of a descriptor in the trie and m is the average length of descriptors in the taxonomy, g is the level of generalization desired for the descriptor. This approach implies generalization still needs to be computed for each incoming event.

We propose to mitigate further the generalization cost by trading off memory usage and building a hashtable. The hashtable is used to precompute a generalization for each descriptor. Therefore, the hashtable's "key-value" pairs are the "descriptor-generalized descriptor" pairs. Building this hashtable requires one single scan of the input taxonomies to store the pair "descriptor-descriptor's antecedent". This corresponds to $g = 1$ level of generalization. If further generalizations are required, the hashtable is used to compute the generalizations of each concept using the "key-value" entries. This approach allows to access a descriptors's antecedent in $O(1)$ time. Our experiments showed that storing this hashtable requires a negligible amount of memory. For instance, the dataset used in Section 6.1.2 is described thanks to approximatively 7000 descriptors. Suppose the average length of a descriptor is 32 characters encoded in UTF-8, i.e., each characters is encoded with 4 bytes. In total, the memory usage for storing the hashtable is $2 \times 7000 \times 32 \times 4 = 1750kB$, which is negligible.

The pseudo-code for constructing this hashtable is given in Algorithm 9 in Appendix B. This algorithm uses the subroutine in Algorithm 8 in Appendix B that actually scans the XML tree.

5.2 Memory footprint

TSAR's memory footprint relies on two objects: (i) the in-memory hashtable that results from precomputing descriptor generalizations and (ii) $|W|$ groups within the temporal locality window w. Since groups that are at a distance d_T greater than w are popped out of window W, these can be directly written to disk or streamed to other applications. Therefore, TSAR only needs to maintain in-memory a small number of groups. This memory usage is bound by w and the average size m of an event's set of descriptors.

One should note that under certain configurations, the number of events in an in-memory group could grow very large. This happens for instance when the temporal locality parameter is chosen too large or when there exist numerous repetitions of similar events throughout the whole time sequence. This scenario can be handled by exploiting (i) the assumption that $s^{\star}_{M}$ is the most useful form of a summary and (ii) the fact the concept formation process simply selects the oldest event in a group of events to represent the group. Therefore, an incoming event e'_i that is similar to events in a group (Y_j, t'_j) in W does not need to be physically added to group (Y_j, t'_j), since group (Y_j, t'_j) will anyways be represented by the oldest event in (Y_j, t'_j) in the concept formation process. Finally, TSAR's overall memory footprint remains constant and limited compared to the amount of RAM now available on any machine.

6 Experimental study

In this section we evaluate and validate the TSAR summarization approach through an extensive set of experiments on real-world data from Reuters's financial news archives. First, we describe the dataset used and discuss how the data is preprocessed into time sequences of events that comply to Definition 1.5. Then, we present how taxonomies are acquired for the descriptive domains on which Reuters's financial news are described. Through these experiments we highlight the following results: (i) TSAR's summarization computational complexity, (ii) the loss of information from content view point, (iii) the loss of information from temporal view point and (iv) the compression ratio achieved.

The rest of this section is organized as follows. We present in Section 6.1 the hardware and software setup of this experimental work. Section 6.2 presents the metrics used to evaluate TSAR's performances and the quality of summaries produced. We report and discuss the results obtained in Section 6.3.

6.1 Experimental setup

6.1.1 Hardware et software setup

The TSAR algorithm presented in Algorithm 1 is implemented under Microsoft Visual Studio 2008 IDE in the C# language. The development environment was chosen for its ease of integration in distributed and service-oriented architectures. TSAR was deployed as a web-service in the context of the STEAD analysis framework demonstrated in [PSPB+08] and discussed in Chapter 4. One objective in this framework is to show in a small environment how summarization could be used to support applications such as data mining.

Experiments on TSAR were carried out on a Core2Duo processor running at 2GHz, 2GB of memory, 4200rpm hard drive and running Windows Vista Pro. The persistence layer, responsible for storing Reuters's time sequences of financial news and summaries built upon these time sequences, was ensured by the latest version of PostgreSQL 8.4.

6.1.2 Reuters's financial news archive

In financial applications, traders are eager to discover knowledge and eventual relationships between live news feeds and market data in order to create new business opportunities [ZZ04]. Reuters has been generating and archiving such data for a couple of decades. To experiment and validate the TSAR summarization approach in a real-world environment, we used one year of Reuters's financial news articles from 2003 and written in English. The unprocessed dataset comes as a log of 21,957,500 entries where each entry includes free text and a set of approximatively 30 attribute-value pairs of numerical or categorical values. An example of raw news event is given in Table 2.5. As provided by Reuters, the data can not be processed by TSAR for the following reasons: (i) TSAR does not process free text and (ii) there is no notion of sequence in the raw archive. Hence, the news data needs first to be cleaned and preprocessed into a collection of time sequences of financial news events computable by TSAR.

Among all the information embedded in Reuters's financial news articles we focused on 4 main components for representing the archive into a collection of time sequences of events:

- **Co_ids:** For each news article, the *co_ids* is a multivalued attribute that corresponds to the IDs of companies to which the current piece of news relates to. Therefore, this attribute is used to build time sequences. In other words, we associate

Timestamp	01 Jan 2004
Company_ID	AAH.AS ABN.N MER.N
Topic code	EUROPE USA Bank INS NL
Free text	"Dutch bank ABN AMRO said on Wednesday it had reached a preliminary agreement to sell its U.S.-based Professional Brokerage business to Merrill Lynch & Co. The Professional Brokerage business provides securities clearing, trade execution and operational support services to participants in the US options and securities markets. It operates out of San Francisco, Chicago and New York. Completion is subject to the usual closing conditions and regulatory approvals, and is expected to be finalized within three months. [...]"

Table 2.5: Example of raw news event

to each company that has a *co_ids* a time sequence of events. Events in this time sequence are taken from the news that mention the company's *co_ids*.

- **Timestamp:** This value serves for ordering news events within each company's time sequence of events. We assumed that there can not be two news events, concerning a same company, such that they have exactly the same timestamp, since timestamps are precise to the millisecond.

- **Topic_codes:** When Reuters's journalists write news articles, they are required to add *topic_codes* to describe the main content and ideas of the news article. In the technical documents received from Reuters, we have identified a total of 525 different codes relating to 17 different categories, equivalently called *descriptive domains* in our work. These categories are the following: cross-market, equities, industrial sector, economy central banking & institution, FX and money market, fixed income, commodities, energy, general news, region, country, sports, language, muni state, special muni, personal finance and new organizations. We decide to use 7 of the most popular categories to describe the data, i.e., {Country, Commodities, Economy Central Banking and Institution, Energy, Equities, Industrial sector, General news}.

- **Free text:** The textual content of a news article is a rich source of information from which precise content descriptors can be extracted. We assume that the 7 categories previously selected from the *topic_code* categories are not informative enough to give precise descriptions of the content of the financial news. Therefore, we define 5 additional categories, namely, {Business, Operation, Economics, Government, Finance}, on which we want news events to be described. For this purpose, we need to extract descriptors corresponding to these categories from the news free text. We used the WordNet [Lab] ontology as described in the rest of this section.

Extracting pertinent descriptors from free text is a non trivial task. This task is made even more difficult when one needs to organize the descriptors extracted into taxonomies. Traditional research work in Natural Language Processing (NLP) could be leveraged to extract relevant key words from the free text based on their corpus, e.g., using Term Frequency-Inverse Document Frequency (TF-IDF) weights as done in [HS02] or using online resources such as OpenCalais [Reu].

However, in this specific application, there exists a number of limitations. Suppose we can leverage NLP techniques to extract relevant descriptors from the free text. TSaR requires descriptors of each descriptive domain to be organized into a taxonomy. The

problem is dual: (i) how do we decide to which descriptive domain a descriptor should be classified? (this is the problem of sense-disambiguated noun hyponym acquisition tackled by Snow et al. [SJN06]), and (ii) how should the extracted descriptors be organized into a taxonomy? These two problems are very interesting and challenging research problems that unfortunately lie out of the scope of this thesis work.

A simple alternative approach is to work the problem the other way around. One can first define or generate taxonomies for the descriptive domains of interest, e.g., the additional categories selected {Business, Operation, Economics, Government, Finance}, then use the taxonomies generated to filter the free text: descriptors from the taxonomies that appear in the free text are extracted and used to describe the content of the news. This approach also has its shortcomings, the most evident being (i) the problem of homonymy and polysemy [Kro97], and (ii) the *coverage* of the taxonomies.

Homonymy and polysemy are common issues that originate from the fact terms used in the free text and descriptors in taxonomies might not refer to the same descriptive domain. For instance, the term "Jersey" can either refer to a knitted sweater (in the *Clothing* category) or to the British island in the English Channel (in the *Country* category). This issue could be circumvented by leveraging techniques in Word Sense Disambiguation (WSD) that use the corpus, context and/or dictionaries [Les86, YSLS07, Rad07, ADLS09] to disambiguate the sense of descriptors extracted from the free text.

The second shortcoming with this alternative approach is the *coverage* of taxonomies for each descriptive domain. In other words, each concept in a given descriptive domain may have multiple synonyms and all synonyms may or may not be captured in the associated taxonomy. The issue is double: either (i) the taxonomy is *too rich*, i.e., all synonyms are captured in the taxonomy, or (ii) the taxonomy is too poor, i.e., only few synonyms are captured in the taxonomy. In the case the taxonomy is *too rich*, multiple synonymous descriptors could be extracted from a piece of free text. In fact, this operation does not necessarily penalize the summarization algorithm since all synonyms will be generalized into one single concept. In the other case, i.e., only few synonyms are captured in the taxonomy, the descriptor extraction process might not be able to extract all relevant descriptors for a given category, e.g., due to the variability of vocabulary used by different journalists. This issue can be overcome by using very rich ontologies or taxonomies such as WordNet [Lab], i.e., bring the issue back to the first one.

In our experimental work, we decided to clean up and preprocess Reuters's financial news archives into time sequences of news events using the alternative approach. Since the domains of *topic_ codes* are relatively small, we manually generated taxonomies for the categories in {Country, Commodities, Economy Central Banking and Institution, Energy, Equities, Industrial sector, General news}. On the other hand, we generated taxonomies for the additional descriptive domains {Business, Operation, Economics, Government, Finance} using the rich general purpose ontology WordNet [Lab]. We chose to use WordNet due to the pre-existing hierarchical organization of concepts and the richness of the vocabulary. Concepts such as nouns in WordNet are linked by at least three relationships: (i) multiple senses for each concept, (ii) hypernym, i.e., antecedents, and (iii) hyponym, i.e., specialization, links. Therefore, we generate taxonomies from WordNet by choosing for a descriptive domain, e.g., Business, one sense we are interested in. The selected sense's hyponyms are then extracted and added as the concept's sons. The process is repeated until the taxonomy reaches a predefined height. Figure 2.7 gives an example showing how the *business* domain taxonomy is extracted.

In total, Reuters's 2003 financial news archive was preprocessed into 1,283,277 news events distributed over 34458 companies' time sequences. Each news event is described on

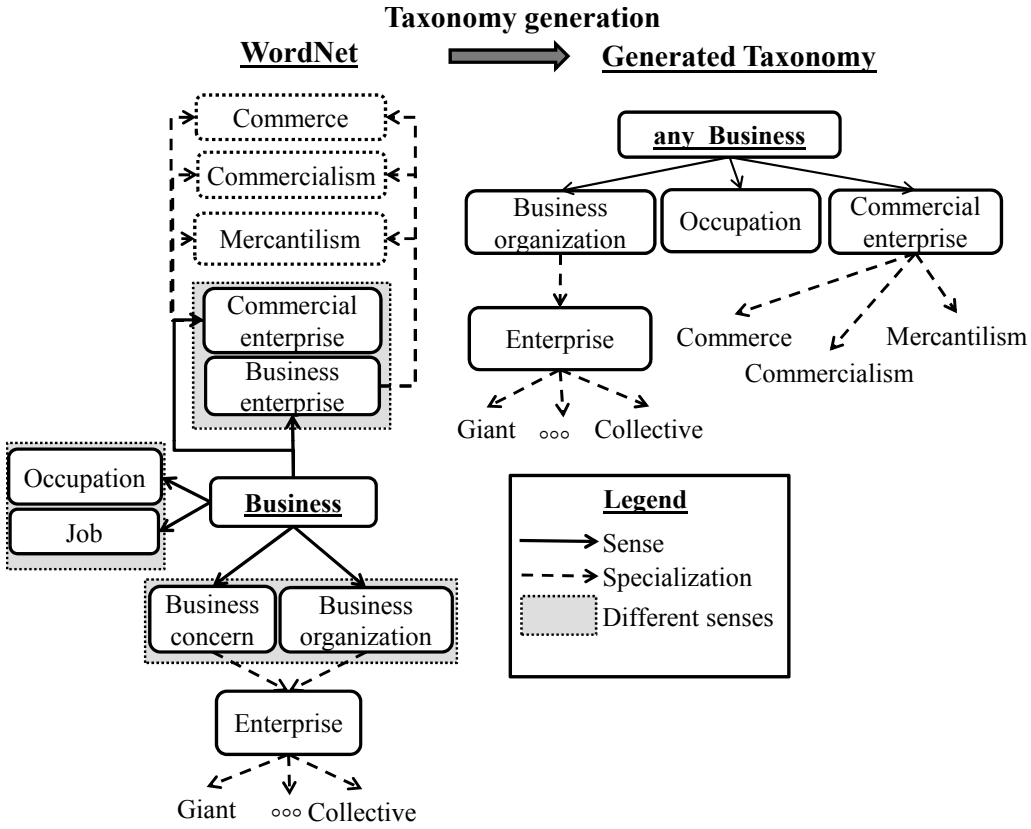


Figure 2.7: Taxonomy generated for the *business* domain

the 12 descriptive domains selected earlier. One should note that several descriptors from each domain can be used to describe a news event. We generated a taxonomy for each of these descriptive domains.

6.2 Metrics

The quality of the TSAR algorithm can be evaluated by several different methods. One method could be evaluating the summarization algorithm w.r.t. the application that it is meant to support, e.g., Sequential Pattern Mining (SPM). In this case, the summary could be evaluated based on its ability to improve the mining computational time w.r.t. not mining summaries, while generating *similar* knowledge. We will develop this aspect further in Chapter 4.

As reminder, TSAR relies on generalization to reduce the variability of event descriptors and on grouping/concept formation to reduce the numerosity of events in the output summary sequence. Also, the algorithm was designed as an incremental process with theoretical linear computational complexity. For these reasons, we propose to evaluate TSAR on its inherent features and hence, we evaluate the summarization process on the following dimensions: (i) the computational time, (ii) the loss of *semantic accuracy*, (iii) the loss of *temporal accuracy* and (iv) the compression ratio. We detail the semantic accuracy, temporal accuracy and compression ratio metrics in the following sections.

6.2.1 Semantic accuracy

TSAR's generalization phase is responsible for reducing the variability of event descriptors. For this reason, we believe it is most important to evaluate the quantity of content

information lost in the summary produced. This metric is called *semantic accuracy*.

The semantic accuracy of the summary is computed as the ratio between the number of generalized descriptors in the summary and the number of descriptors in the original time sequence. This measure is formally defined in Definition 2.5. The semantic accuracy expressed here is a simple vision of how content information is lost. Indeed, it does not take into account the provenance of descriptors nor the structure of taxonomies used to generalized the data, e.g., height and/or width of the taxonomies. In other words, the loss of content information is the same whether descriptors are taken from short or deep taxonomies. This semantic accuracy measure could take into account such these structural information of taxonomies for more refined analysis.

Also, note that this semantic accuracy measure by itself is inadequate to evaluate the quality of the summary produced. It should be considered together with the *temporal accuracy* metric defined in Definition 2.6 and the compression ratio achieved as defined in Definition 2.7.

Definition 2.5 (Semantic accuracy)
Given s a time sequence of events in $\mathcal{S}(\Omega)$ and its summary $s^\star_M$ by $\chi_{\vartheta,w}$, let $D \in \mathcal{P}(\Omega)$ be the set of descriptors on which all events in s are defined, i.e., $D = \bigcup_{e \in s} x$, $e = (x, t)$, and let $D' \in \mathcal{P}(\Omega)$ be the set of descriptors on which all events in $s^\star_M$ are defined, i.e., $D' = \bigcup_{e^\star \in s^\star_M} x^\star$, $(x^\star, t^\star)$. The semantic accuracy measure of the summary $s^\star_M$, denoted α, is defined as the following ratio:

$$\alpha = \frac{|D'|}{|D|}$$

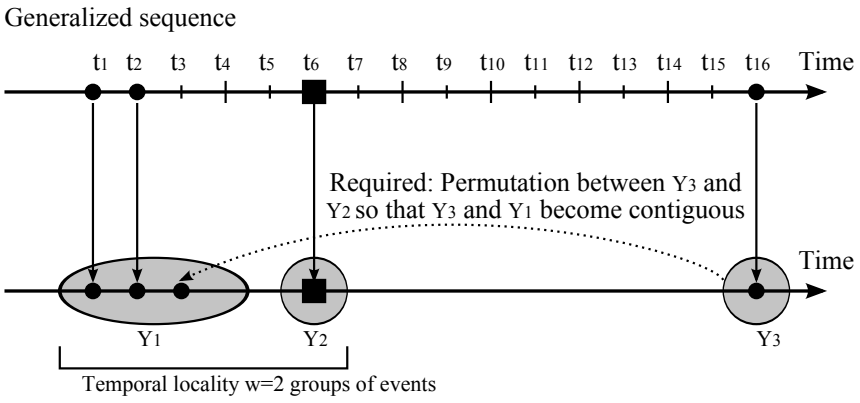
α ranges in $[0, 1]$ and the closer α to 1 the higher the semantic accuracy of the summary.

6.2.2 Temporal accuracy

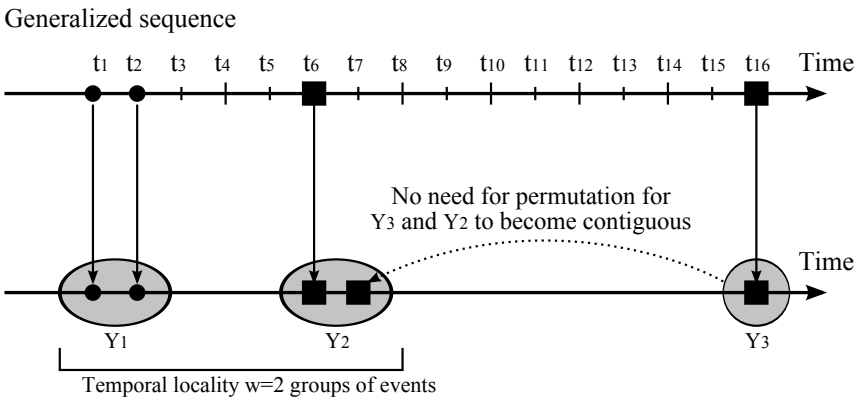
In addition to semantic accuracy, we also propose a *temporal accuracy* measure. Intuitively, the grouping process fixes a sliding temporal locality window W and groups all similar events within a distance $d_T \leq w$. This process can be understood as a form of event rearrangement on the timeline so that events *close* on the timeline, i.e., event within a same temporal locality, become *contiguous*. Intuitively, two events $e_i = (x_i, t_i)$ and $e_{i+1} = (x_{i+1}, t_{i+1})$ are considered contiguous if there does not exist an event $e_k = (x_k, t_k)$ such that $t_i < t_k < t_{i+1}$; We will develop this notion in Chapter 3. The temporal accuracy of a summary is computed as a temporal rearrangement penalty cost that reflects all the permutation operations necessary to rearrange events.

Figure 2.8 gives an example that illustrates this idea of event rearrangement. Figure 2.8(a) shows that Y_3 needs to be permutated one time with Y_2 to be *contiguous* and grouped with group Y_1. In this case, we arbitrarily assume this permutation operation comes with a cost equal to 1. On the other hand, Figure 2.8(b) shows that Y_3 does not need any permutation to become contiguous with group Y_2. Therefore, the temporal accuracy of the summary can be evaluated by summing up all permutations performed during the grouping phase of the TSaR algorithm. The temporal accuracy of a summary is best when the value $\mathcal{C}_\tau$ is closer to 0. When the temporal locality is defined as an integer value, this situation occurs when the temporal locality parameter is chosen equal to 1. When the temporal locality is defined as a duration, this situation occurs when the temporal locality parameter is chosen as the smallest duration between two similar events. The cost associated to this rearrangement operation is formalized in Definition 2.6.

Definition 2.6 (Temporal rearrangement penalty cost)
Given $\overline{s}$ a time sequence summary extent in $\mathcal{S}^2(\Omega)$, $\overline{s} = \{(Y_1, t'_1), \ldots, (Y_m, t'_m)\}$, a temporal



(a) Example of one event rearrangement



(b) Example of no event rearrangement

Figure 2.8: Examples of event arrangements

locality parameter w defined as an integer value with $w > 1$ and its associated temporal window W, the temporal rearrangement penalty cost for grouping an incoming event $e = (x, t)$ with a group $(Y_j, t'_j) \in W$ is denoted $\mathcal{C}_\tau(e, \overline{s})$. $\mathcal{C}_\tau(e, \overline{s})$ expresses the number of rearrangements necessary on the timeline of $\overline{s}$ so that event e can be grouped with group (Y_j, t'_j) in window W. $\mathcal{C}_\tau(e, \overline{s})$ penalizes incoming events that are grouped with older groups in W. On the other hand, if (Y_j, t'_j) is the most recent group in W, i.e., $(Y_j, t'_j) = (Y_m, t'_m)$, no penalty occurs since e and (Y_m, t'_m) would be contiguous on the timeline. $\mathcal{C}_\tau(e, \overline{s})$ is formally defined as follows:

$$\begin{cases} \bullet \text{ if } (\forall (Y, t') \in W,\ \forall e_u \in Y_j,\ x \neq x_u),\ \text{then } \mathcal{C}_\tau(e, \overline{s}) = 0 \\ \bullet \text{ if } (\exists (Y_j, t'_j) \in W,\ \exists e_{j,u} \in Y_j,\ x = x_{j,u}),\ \text{then:} \\ \quad * \text{ if } (\nexists k > j,\ (Y_k, t'_k) \in W),\ \text{then } \mathcal{C}_\tau(e, \overline{s}) = 0 \\ \quad * \text{ else, given } (m = |\{(Y_k, t'_k) \in W, k > j\}|, 1 \leq m \leq w - 1)\ \text{then } \mathcal{C}_\tau(e, \overline{s}) = m \end{cases}$$

The total temporal rearrangement penalty cost for summarizing s into $s^\star_M$ is denoted $\mathcal{C}_\tau(s^\star_M)$ and is given by the following equation:

$$\mathcal{C}_\tau(s^\star_M) = \sum_{e \in \overline{s}} \mathcal{C}_\tau(e, \overline{s}) \tag{2.5}$$

The absolute value of the total temporal rearrangement penalty cost is meaningless by itself and needs to be normalized so other summaries' costs are comparable. For this purpose, since to the best of our knowledge, there is no comparable summarization techniques,

we propose to normalize $\mathcal{C}_\tau(s^\star_M)$ with the temporal rearrangement penalty cost of the *worst* summary that can be produced, denoted $s^\star_{worst}$. Intuitively, given a generalization level ϑ, $s^\star_{worst}$ should be the most compact summary, i.e., $\rho(s, s^\star_{worst}) = 1$, while maximizing the total temporal rearrangement penalty cost to produce. If X is the set of distinct itemsets of events in s, i.e., $X = \{x_i \in \mathcal{P}(\Omega), (x_i, t_i) \in s\}$, then $|s^\star_{worst}| = |X|$ and each itemset in X should be used to describe only on event in $s^\star_{worst}$ and should appear in $s^\star_{worst}$ in the order in which they appear in s.

Unfortunately, building the worst summary $s^\star_{worst}$ is a difficult and non trivial task, and is itself an optimization task. An approximation of the worst summary, denoted $\overline{s^\star_{worst}}$, can be built as follows:
Let us denoted by E_i the set of all occurrences of events in s that are described by itemset x_i; Formally, $E_i = \{e \in s, x = x_i\}$. Then, the set $\mathcal{X} = \bigcup X_i$ is the collection of all X_i's in s. For each X_i, we denote by $t_{i,1}$ the first time of occurrence of events in X_i.

An approximation of the temporal rearrangement penalty cost for the worst summary $\overline{s^\star_{worst}}$ is the sum, for each X_i, of the temporal rearrangement penalty cost between events in X_i and event $(x_i, t_{i,1})$, i.e.:

$$\mathcal{C}_\tau(\overline{s^\star_{worst}}) = \sum_{X_i \in X} \sum_{(x_j, t_{j,k}) \in X_i} (d_T(t_{j,1}, t_{j,k}))$$

where $d_T(t_{j,1}, t_{j,k}) = 0$ if $(x_j, t_{j,1})$ and $(x_j, t_{j,k})$ are contiguous on the timeline. However, in practice, our experiments showed that $\mathcal{C}_\tau(\overline{s^\star_{worst}})$ is 3 to 4 orders of magnitude higher than the temporal rearrangement cost of any summary we produced in TSAR. This observation motivates or decision to choose a different normalization cost for the results to be more meaningful and tangible.

For this reason, we propose to normalize the total temporal rearrangement penalty cost $\mathcal{C}_\tau(s^\star_M)$ thanks to the range of the input parameters used in our experiments. We detail in Section 6.3, the parameters used for evaluating our algorithm. The *worst* summary that can be built with these input parameters is obtained with the strongest generalization vector and the largest temporal locality parameter, i.e., $\vartheta = \langle 3 \rangle$ and $w = 100$, respectively. Therefore, we can measure the temporal accuracy of a summary thanks to the following formula:

$$\beta = 1 - \frac{\mathcal{C}_\tau(s^\star_M)}{\mathcal{C}_\tau(\pi \circ \chi_{\{3\},100}(s))} \tag{2.6}$$

β is in the range $[0, 1]$ and the higher β, the better the temporal accuracy.

6.2.3 Compression ratio

The third measure used to evaluate the quality of a summary is its capability of reducing the number of events w.r.t. the number of events in the original time sequence of events. This measure is called the *compression ratio* ρ, also known as *compression rate*, *compaction rate* or *compaction gain*. The higher ρ, the better the compression ratio achieved. We chose to use ρ as a measure to evaluate the quality of a summary since it is a well known and well accepted measure for evaluating the compression effect of algorithms.

Definition 2.7 (Compression ratio)
Given s a time sequence of events in $\mathcal{S}(\Omega)$ and its summary by $\chi_{\vartheta,w}$, i.e., $\chi_{\vartheta,w}(s) = (s^2_C, s^\star_M)$, the compression ratio achieved by $\chi_{\vartheta,w}$, denoted $\rho(s, \chi_{\vartheta,w}(s))$, is defined as the following ratio:

$$\rho(s, \chi_{\vartheta,w}(s)) = 1 - \frac{|s^\star_M| - 1}{|s| - 1}$$

where $|seq|$ is the number of events in a time sequence seq. The ratio ρ is in the range $[0,1]$ and the closer ρ to 1, the higher the compression ratio achieved.

In contrast with traditional work on compression algorithms, we do not limit the evaluation of our time sequence summarization algorithm to the compression ratio only. We believe ρ needs to be weighted by the summary's semantic and temporal accuracy, α and β, respectively.

6.3 Results

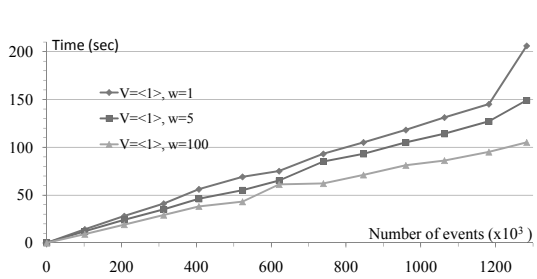
We present in this section our experimental results. We evaluate TSAR on the following four dimensions: (i) computational time, (ii) semantic accuracy of summaries generated, (iii) temporal accuracy of summaries generated and (iv) compression ratio of summaries produced.

Here, we are interested in the impact of the generalization vector and the temporal locality parameter on the computational time, semantic accuracy, temporal accuracy and compression ratio. Thus, the parameters setup used to build TSAR summaries is the following:

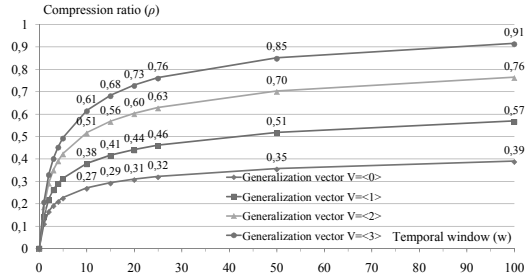
- Generalization vector: ϑ is taken in the set $\{\langle 0\rangle, \langle 1\rangle, \langle 2\rangle, \langle 3\rangle\}$.
- Temporal locality parameter: w is varied between 1, i.e., a very strict temporal locality where only similar contiguous events are considered within a same temporal locality, and 100 which reflects a very strong relaxation of the temporal locality, i.e., the algorithm degenerates into a regular compression algorithm that does not take into account the temporal aspect of data.

We start by setting the generalization vector $\vartheta = \langle 1\rangle$, i.e., all descriptors are generalized once, and $w \in \{1, 2, 3, 4, 5, 10, 15, 20, 25, 50, 100\}$. The maximum value for w was chosen equal to 100 (i) to represent a very strong temporal locality relaxation and (ii) because the quality of the summary produced with setting $w = 100$ is used as baseline for computing the relative temporal accuracy measure. Figure 2.9(a) gives TSAR's computational time for different temporal locality parameters. For the sake of readability, we only display the plots for temporal locality parameter w in $\{1, 5, 100\}$. These plots show that whatever the temporal locality parameter used, TSAR is linear in the number of input events. In addition, we can observe that computational time is almost constant whatever the temporal locality window considered. A slight variation can be observed in between different values of w and has two complementary explanations. First, it is more costly to scan large temporal locality windows during grouping. Second, a larger temporal locality window w allows to maintain more groups in-memory and, so, requires less I/O operations for writing into storage.

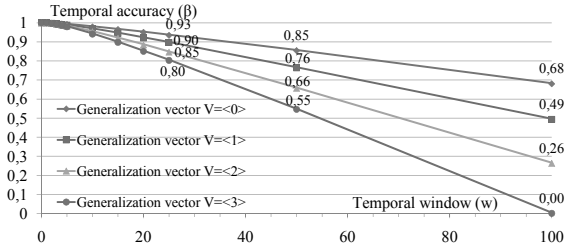
We compute the compression ratio ρ achieved by summaries built with different generalization vectors $\vartheta \in \{\langle 0\rangle, \langle 1\rangle, \langle 2\rangle, \langle 3\rangle\}$. The results are given in Figure 2.9(b). Note that the best compression ratio achieved with $\vartheta = \langle 0\rangle$ is only 0.39. For a given temporal window w, relaxing the semantic accuracy of the data by generalizing each descriptor once, twice or three times allows an average compaction gain of +46.15%, +94.87% and +133.33% respectively. In other other words, the compression ratio is approximatively doubled when increasing the generalization level. Another interesting observation is that for all generalization vector ϑ, the plots show that highest numerosity reduction is achieved with larger temporal windows while computational times are almost constant, as shown in Figure 2.9(a). This observation is very helpful from user view point for setting the summarization parameters ϑ and w. In fact, as computational times are almost constant



(a) Computational time



(b) Numerosity reduction



(c) Temporal accuracy

Figure 2.9: Summarization performances

whatever the temporal locality window considered, the user needs only to express the desired precision in terms of (i) semantic accuracy for each descriptive domain and in terms of (ii) temporal locality, without worrying about computational times.

Table 2.6 gives the semantic accuracy of the summaries produced and Figure 2.9(c) gives their temporal accuracy. Note in Table 2.6 that the number of descriptors in the raw data, i.e., $\vartheta = \langle 0 \rangle$, is 1208. When summarizing each descriptive domain once, i.e., $\vartheta = \langle 1 \rangle$, the number of descriptors in the summaries drops to 50. This loss of semantic information can be explained by our choice of using the WordNet ontology. Indeed, we mentioned earlier in Section 6.1.1 that we chose to generate very rich taxonomies (to address the issue of taxonomy coverage). For this reason, numerous descriptors extracted from the free text might in fact be synonyms and are easily generalized into one common concept. The semantic accuracy is then directly impacted. Consequently, the concepts obtained with $\vartheta = \langle 1 \rangle$ should be considered as better descriptive concepts than the raw descriptors. Hence, we choose to compute α using as baseline $|D'| = 50$, as shown in Table 2.6. In this case, each time ϑ is increased, the semantic accuracy is approximatively halved. This observation is consistent with our previous observation on the average compaction gain.

ϑ	$\lvert D' \rvert$	α
$\langle 0 \rangle$	1208	N/A
$\langle 1 \rangle$	50	1
$\langle 2 \rangle$	20	0.40
$\langle 3 \rangle$	13	0.26

Table 2.6: Semantic accuracy

Figure 2.9(c) gives the relative temporal accuracy of each summary. Higher levels of generalization reduce the temporal accuracy of the summaries. This phenomenon results from the reduction of the variability of events through generalization. Incidentally, more event rearrangements are made possible during the grouping phase. However, the temporal accuracy remains high, i.e., ≥ 0.80, for small and medium sized temporal windows, i.e.,

$w \leq 25$. The temporal accuracy only deteriorates with large windows, i.e., $w \geq 25$. This result means that the user can achieve high compression ratios without sacrificing much the temporal accuracy of the summaries.

Unfortunately, guaranteeing the compression ratio ρ achieved with TSAR is a difficult task, if not impossible, as its depend on the input parameters and on the data's distribution. Also, the user needs to weight the semantic and temporal accuracy he is ready to trade off for higher compression ratio. Therefore, guaranteeing a compression ratio becomes an optimization problem that requires the algorithm to self-tune the input parameters and take into account the user's preferences.

Through this experimental study, we have shown that TSAR is a time sequence summarization approach that achieves summarization in linear time. In order to evaluate the quality of summaries produced by TSAR, we introduce two measures, namely the semantic accuracy and the temporal accuracy of a summary. Hence, TSAR produces summaries that achieve high compression ratios while maintaining high temporal accuracy. More compression ratio can be achieved by allowing the process to generalize further event descriptors with the consequence of reducing semantic accuracy. We empirically show that increasing once the level of generalization of all event descriptors reduces the semantic accuracy of summaries produced by approximatively 50%. It is then up to the user to decide, for the purpose of achieving more compression ratio, whether he is ready (i) to accept more temporal accuracy loss and maintain high semantic accuracy or (ii) to accept more semantic accuracy loss and maintain high temporal accuracy loss.

7 Chapter summary

In this chapter, we have presented a user-oriented approach to build time sequence summaries, called TSAR. TSAR relies on the a generalization, grouping and concept formation process. Input time sequences of events are incrementally processed and event descriptors are represented at a higher level of abstraction thanks to the use of Background Knowledge expressed in the form of domain specific taxonomies.

The user has a limited number of parameters to set, i.e., two parameters, for TSAR to operate. These parameters are the generalization vector ϑ and the temporal locality parameter w. The generalization vector ϑ controls the level of abstraction of the data by indicating for each descriptive domain the number of generalizations that need to be operated on descriptors taken from that domain. The grouping process is responsible for grouping together similar events, i.e., events having same descriptors, that are *close* on the timeline. This notion of temporal closeness is captured by the temporal locality parameter. This temporal locality parameter controls how well the chronology of events on the timeline should be preserved. Finally, the concept formation process is responsible for outputting the summary in the form of a regular time sequence.

We have provided an extensive set of experiments to evaluate and validate our summarization approach. Therefore, we showed that TSAR has linear computational complexity and is capable of generating summaries that achieve high compression ratios. We provide two metrics, namely the semantic accuracy and the temporal accuracy metrics, to evaluate the quality of summaries produced by TSAR. These metrics give the user an indication on how precise summaries produced are from content and from temporal view points. Since the compression ratio of a summary can not be guaranteed, the user needs to decide whether he is ready to trade off semantic accuracy or temporal accuracy to achieve higher compression ratios.

Two interesting work orientations already addressed in the following chapters of this

thesis work are: (i) how to make time sequence summarization a full-fletched parameter-free process? and (ii) how to use TSAR summaries in real world data mining applications? On the one hand, even though parameter-free algorithms might not produce summaries that reproduce with fidelity a user's representation of the data, in some case, he might just want the data to be contained within a certain memory space. Hence, the summarization problem becomes an optimization problem: the summarization function needs to produce a summary that achieves a desired compression ratio while minimizing the loss of semantic and temporal accuracy. In other words, events on the timeline should be gathered in a way that considers their content and proximity on the timeline. By addressing this issue, we actually define a novel conceptual clustering problem. This problem will be thoroughly explored in Chapter 3.

On the other hand, we motivated the design of TSAR as a user-friendly summarization approach that as the goal of supporting chronology dependent and processing intensive applications. Still, we need to evaluate if the approach proposed is useful in practice. For instance, Sequential Pattern Mining is an application that highly depends on the sequentiality of the data. An interesting work orientation is then to study how well TSAR can benefit such application. Intuitively, since TSAR's generalization process expresses event descriptors at a higher level of abstraction, patterns discovered on TSAR summaries should give higher order knowledge of the data. Also, since TSAR's grouping process reduces the size of sequences, there should be a direct consequence on the nature and number of patterns that can be discovered on summaries w.r.t. patterns normally discovered on non summarized sequences. This study will be discussed in details in Chapter 4.

In Section 6.1.2, we have presented a simple approach to preprocess and generate taxonomies for the dataset used in our experiments. There exist a number of interesting approaches in Natural Language Processing research that address the issues of extracting descriptors from corpus data and the issues of building taxonomies, in the presence or not of corpus data. These more advanced NPL techniques could be leveraged to preprocess Reuters's financial news archives and to build more refined taxonomies for the descriptive domains of interest. One direction of our future work is to study the impact of such methods on summarization. In particular, we are interested in knowing if *higher quality* preprocessed data and *higher quality* taxonomies could allow to produce summaries having higher semantic and/or temporal accuracy while achieving more compression.

Chapter 3

Time sequence summarization as a novel clustering problem

1 Introduction

The TSaR summarization technique presented in Chapter 2 proceeds in three steps: (i) generalization of event descriptors, (ii) grouping of events having identical generalized descriptors within a certain temporal locality and (iii) representation of each group formed by a representative event. TSaR is tuned by the user thanks to several input parameters, i.e., (i) taxonomies are used to abstract event descriptors and it is required from the user (ii) a generalization parameter and (iii) a temporal parameter that control the content and temporal accuracy, respectively, of summaries produced. In some situations, e.g., when the user does not know at what level of abstraction the data should be represented, these parameters may not be trivial to set and require the user to try different combinations of settings. In addition, if the user is interested in having a summary of a given size, since each parameter setting induces a different compression ratio, that desired compression ration can not be guaranteed beforehand. TSaR does not satisfy the parameter-free property introduced by Kiernan and Terzi [KT08] for event sequence summarization. This parameter-free property is a nice feature for unburdening the user with choosing these input parameters.

Now, if one looks into minute details, one can see that summaries produced by TSaR strikingly resemble clusters that could be produced by a traditional data clustering algorithm. In fact, the underlying ideas are very similar, i.e., TSaR divide or *partition* events in a time sequence into *dissimilar* groups of *similar* events. Indeed, TSaR only gathers together events whose generalized descriptors are identical. In other words, TSaR gathers events whose event descriptors are *similar* at a given level of abstraction; This level of abstraction is fixed by the generalization process. The grouping operation typically corresponds the methodology of traditional data clustering techniques. Most traditional data clustering techniques rely on the joint features of two objects and compute pair-wise distances, or *linkage distances*. Data objects are grouped when their similarity matches a given condition. Also, in TSaR, the only events eligible for grouping should be located within a certain temporal locality defined by the temporal locality parameter w. This condition can be understood as a form of temporal segmentation of the timeline.

Here, we address the shortcoming of TSaR, i.e., the approach not being parameter-free, and propose to redefine the time sequence summarization problem so that summaries can be built in a parameter-free manner. For this purpose, we translate the time sequence summarization problem from the *generalization grouping and concept formation* paradigm into the *clustering and concept formation* paradigm. In this prospect, we show that time

sequence summarization can be understood under the angle of clustering where the objective function to optimize takes into account both the categorical content and the temporal information of event in the sequence. Once clusters of events are defined, a concept formation technique is operated to provide for each cluster a representative event to which is associated a timestamp. Hence, the set of representative events are ordered on the timeline and define the time sequence summary. More formally, given an input time sequence and a desired compression ratio, time sequence summarization then becomes the problem of producing a summary that achieves the desired compression ratio while minimizing a cost function. Designing a time sequence summary that displays all desirable properties listed in Chapter 1 in a parameter-free manner is a challenging task, and is, to the best of our knowledge, a problem that has not been addressed so far.

Under the light of these observations, designing a time sequence summary that displays all these properties is equivalent to designing a data clustering method that needs to fulfill two conditions: (i) handle categorical features and (ii) handle the time associated to events. Regarding categorical data, there exists a wealth of research work on clustering categorical data. We discuss these techniques in Section 7. In fact, we believe a most challenging issue in this new problem formulation is the way temporal information is handled. The most intuitive and straightforward manner to handle the time associated to events in a time sequence is to consider the time dimension as a numerical feature that will be treated as any other feature. However, let us show the limitations of this assumption through the two following examples.

On the one hand, suppose the time dimension is considered as a feature equivalent to any other and events are described on numerous features, e.g., 100 features. Mechanically, the time dimension will have less weight than in a situation where events were described on a smaller number of features, e.g., 10 features. Consequently, two very similar events that occur very distantly on the timeline could be grouped together. This situation is illustrated in Figure 3.1: Events e_1 and e_{10} are both dark circles but are very distant on the timeline, i.e., events very similar on every feature except on the time dimension. Under this assumption, the dark circles will have very high probability to be clustered together regardless of the time that separates them.

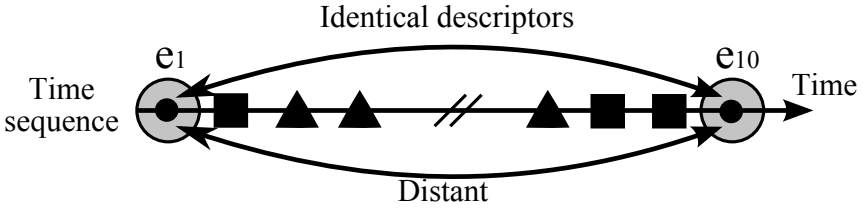


Figure 3.1: Example of similar but distant events

On the other hand, suppose the time dimension is considered as a discriminative dimension. In other words, the clustering method will first segment the timeline then cluster events within, and limited to, each segment of the timeline based on their joint features. This approach is exactly the one adopted by Kiernan and Terzi in [KT08] as described in Chapter 1. The idea is interesting but the temporal segmentation operated could be too crisp and might prevent more compact clusters to be formed. This situation is illustrated in Figure 3.2: The timeline is segmented into three segments S_1, S_2 and S_3 and the clustering of each segment produces in total 5 clusters. However, one should notice that the dark squares in segment S_1 and S_2 generate two individual clusters, while their distance on the timeline does not seems more important than the distance between the first and third dark circles in segment S_1. Therefore, the alternate clustering given in Figure 3.2

where the two dark squares are grouped into one single cluster seems to be a more *natural* clustering.

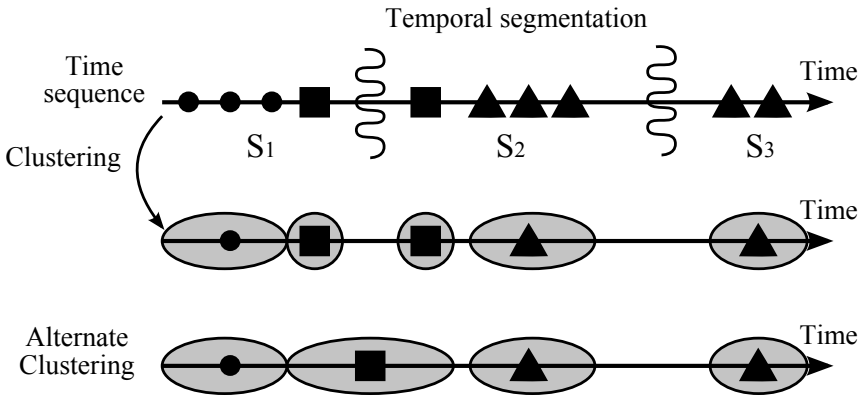


Figure 3.2: Example of clustering with time as a discriminative dimension

Intuitively, a solution to this issue is to attribute an appropriate weight to the time dimension. Therefore, the challenge of designing a time sequence summary as a parameter-free clustering problem comes down to the problem of handling categorical descriptors and, most importantly, to handling the weight of the time dimension. We intend to tackle this challenge in this chapter. For this purpose, we reformulate time sequence summarization in a way that completely includes the temporal information associated to events in a time sequence. By doing so, we present time sequence summarization under the light of a novel conceptual clustering problem where the objective function to optimize relies on a measure or cost that determines how *close* two events are thanks to their content and their time of occurrence. The contributions discussed in this chapter are the following.

Contributions

First, we redefine time sequence summarization using data clustering terminology and reformulate time sequence summarization as a novel conceptual clustering problem. For this purpose, we define a new but simple cost function to evaluate the distance between events on the timeline. The novelty of this cost function lies in the dual consideration of the *similarity* of events from content view point and the *proximity* of events on the timeline. Thus, we present this novel conceptual clustering problem as follows: Given a desired compression ratio, an optimal time sequence summary is a summary that achieves the desired compression ratio while globally minimizing the cost function.

Second, we propose three solutions to build a time sequence summary under this new definition, namely N-TSS, G-BUSS and GRASS. N-TSS is a naive solution that builds all candidate summaries and then chooses the one that minimizes the cost function. Since this approach is the simplest, common sense would entice us to choose this solution as the baseline approach for comparison. However, we will show that approach has prohibitive performances. Hence, we also propose two greedy algorithms that build locally optimal solutions, namely, G-BUSS and GRASS. G-BUSS is a greedy hierarchical ascending clustering algorithm that aggregates clusters based on their content and temporal information. Since, G-BUSS has quadratic computational complexity, we propose a pruning technique that relies on the characteristics of the cost function used to reduce in practice G-BUSS's computational time. GRASS is a greedy parallel random-seeded algorithm that heavily uses the pruning technique introduced for G-BUSS and that explores the search space at multiple sites.

Third, we evaluate and validate the algorithms proposed thanks to an extensive set of experiments on real world data: Reuters's 2003 financial news archive, i.e., approximatively 1.2M news events. We propose three measures to evaluate the quality of summaries produced by the approaches proposed: (i) computational time, (ii) total clustering cost and (iii) quality of clusters produced. Therefore, we summarize the input data and evaluate each approach on these three dimensions. Our preliminary experimental study shows that the N-TSS solution's computational and memory footprints are prohibitive. For this reason, G-BUSS is in turn chosen as the baseline approach. Hence, GRASS's performances are compared to G-BUSS's performances. We show that GRASS reduces in practice G-BUSS's computational time by two to three orders of magnitude while producing clusters of equivalent quality.

Organization of the chapter

The remaining of this chapter is organized as follows. We define time sequence summarization using clustering terminology in Section 2 and we present in Section 3 the problem of building optimal summaries (or clusters) under this new definition. Section 4 presents a naive solution, called N-TSS, to build optimal time sequence summaries. Section 5 presents our two greedy solutions, namely G-BUSS and GRASS. We provide in Section 6 an extensive set of experiments to evaluate and validate the algorithms proposed. Related work is then discussed in Section 7. We conclude this chapter in Section 8 and discuss some perspective work.

2 Time sequence summarization formulated in clustering terminology

We have highlighted the existence of a very close link between time sequence summarization and data clustering. This observation has motivated us to reformulate time sequence summarization as a conceptual clustering problem where an objective function needs to be optimized. Here, this objective function to optimize is a cost function to globally minimize. The originality of this cost function resides in the fact it considers the content and the time associated to events. Hence, we provide in this section a reformulation of a time sequence summary, inspired and revisited from Definition 1.7 given in Chapter 1.

Events in a time sequence are defined on two orthogonal dimensions: (i) their content, i.e., event descriptors, and (ii) their time of occurrence, i.e., a timestamp. Then, the *similarity* between two events $e_i = (x_i, t_i)$ and $e_j = (x_j, t_j)$ should be computed by evaluating (i) the similarity between their descriptive content, i.e., x_i and x_j, in conjunction with (ii) their *proximity* on the timeline using their temporal information, i.e., t_i and t_j. Intuitively, the situation that is easily understood and well addressed is when events are similar and close on the timeline or the opposite situation, i.e., dissimilar and distant on the timeline. The murkier situation is when events are *quite* similar and *quite* close on the timeline. We believe such events should be given a chance to be grouped together. However, if such grouping occurs, a cost expressed as a loss of temporal accuracy should also be taken into account by the clustering process. In total, we propose to devise a methodology to evaluate what it would cost in terms of *semantic accuracy* and in terms of *temporal accuracy* for these events to be considered *similar* on both dimensions. This mechanism is then leveraged as the cost function to minimize for operating time sequence summarization as a data clustering task.

As a reminder, a time sequence summary of a sequence s, as defined in Chapter 1, is denoted $\chi_{\vartheta,w}(s) = (s^2_C, s^\star_M)$. Under this definition, s^2_C is a second-order time sequence that is the extension of the summary and $s^\star_M$ is a sequence of representative events that is the intent of the summary. In Chapter 1, the intent $s^\star_M$ is obtained by building for each group Y in s^2_C an event $e^\star = (x^\star, t^\star)$ where $x^\star$ is a set of descriptors taken from any event in Y, since all events in Y have identical sets of descriptors, and $t^\star$ is the timestamp of the first event in Y.

In fact, defining an intent for the summary of a sequence s requires to characterize subsequences of s both on their content and their position on the timeline. Thus, achieving the summarization task as a clustering task requires to define a *concept formation* model that is able to provide a single generalized event $e^\star$ from a group of events $Y = \{e\}$ with $e = (x, t)$. Depending on the targeted application, this model could take various forms. A basic but realistic model is for instance to generalize every descriptor in the x's by the way of IS-A hierarchies on domains D_A, until they all become the same, denoted $x^\star$ for each Y. Then, the generalized description $x^\star$ is the description associated to the generalized event $e^\star$.

From temporal view point, the time of occurrence t of events e in Y should be used to generate a timestamp $t^\star$ that represents the time of occurrence of event $e^\star$. The aggregation function used to compute $t^\star$ deeply depends on the semantics one wants to give to the summary. For instance, $t^\star$ could be computed by taking the minimum timestamp t_{min} in Y. This is the choice done in TSaR. This choice reflects our preference for the information that the series of events in Y started at t_{min}, i.e., the time of occurrence of the first event represented by $e^\star$. On the other hand, if $t^\star$ is computed by taking the maximum timestamp t_{max} in Y, it would mean that event $e^\star$ represents a series of events that end at the time of occurrence of the last event represented by $e^\star$. Eventually, $t^\star$ could not represent a time of occurrence but a duration and inform of the length of time on which a series of events spans.

More formally, we need a function to characterize events' content and temporal information. This characterization function is described in Definition 3.1.

Definition 3.1 (Dual characterization function $F = (f, \mathsf{T})$)
Given s a time sequence of events in $\mathcal{S}(\Omega)$, $s = \{e_1, \ldots, e_m\}$ with $e_i = (x_i, t_i)$, the dual characterization function F, defined on $\mathcal{S}(\Omega)$, produces a single event $e^\star = (x^\star, t^\star)$ from s, $F(s) = e^\star = (x^\star, t^\star)$. F is composed of a couple of function, i.e., $F = (f, \mathsf{T})$ where:

- $f(\{x_1, \ldots, x_m\}) = x^\star$, $x^\star \in \mathcal{P}(\Omega)$ *and*
- $\mathsf{T}(\{t_1, \ldots, t_m\}) = t^\star$, t *is a timestamp, i.e., value on the timeline, or eventually a duration*

Equipped with this dual characterization such function, we are now able to propose the cluster-based definition of a time sequence summary. This definition is given in Definition 3.2.

Definition 3.2 (Time sequence summary revisited)
Given a time sequence $s \in \mathcal{S}(\Omega)$; Let F be a dual characterization function defined on $\mathcal{S}(\Omega)$. Using clustering terminology, we define the time sequence summary of s, denoted $\chi_F(s) = (s^2_C, s^\star_M)$, as follows:

- $\Pi(s) = \{Y_k\}$ *is a partition of events in s where every cluster $Y_k = \{e_{ki}, 1 \leq i \leq n\} \in s^2_C$ is a subset of events e_{ki} of s and $\bigcup_{\Pi(s)} Y_k = s$ and $Y_k \cap Y_\ell = \emptyset$ pairwise.*

- $s^\star = \{e_k^\star = F(Y_k)\}$ *is the time sequence of representative events* $e_k^\star = (x^\star, t^\star)$ *formed from every cluster* Y_k *in* $\Pi(s)$*, with* $(x_k^\star = f(x_{ki}, 1 \leq i \leq n), t_k^\star = \mathsf{T}(\{t_{ki}, 1 \leq i \leq n\}))$.

Given a partition $\Pi(s) = \{Y_k\}$ of a time sequence s ; Let F be a dual characterization function that provides a way of building a generalized event from an entire sequence. Then, we can compute the intent $s^\star$ of $\chi_F(s)$ such that for every Y_k in $\Pi(s)$ we have $e_k^\star = F(Y_k)$. For instance, we used in TSaR, a *generalization and merging* paradigm to compute the summaries, and F has been dually defined as (i) f : the *Most Specific Common Subsumer* (MSCS) or *Lowest Super-Ordinate* (LSO) of the descriptors in the IS-A hierarchy and (ii) T : the minimum timestamp of each group formed in s_C^2. Hence, we can claim that our new definition of a time sequence summary generalizes the definition of time sequence summarization given in Chapter 1.

From the above definition, $\Pi(s)$ and $s^\star$ can be understood as the *extent* and the *intent*, respectively, of summary $\chi_F(s)$. In other words, every $e^\star$ in $s^\star$ is the *concept*, represented with a single generalized event, of a group Y of raw events of s. Besides, observe that every Y is also a subsequence of s if we rank the events e_i by means of their timestamps t_i as for regular time sequences.

Finally, let us be given the summary of a time sequence s, i.e., $\chi_F(s) = (\Pi(s), s^\star)$ where $\Pi(s) = \{Y_1, \ldots, Y_m\}$ and $s^\star = \{e_1^\star, \ldots, e_m^\star\}$ and $e_i^\star = (x_i^\star, t_i^\star)$. Notice that we can build a canonical second-order time sequence denoted S from $\chi_F(s)$. For this purpose, it suffices to associate each cluster Y_k of $\Pi(s)$ with the corresponding timestamp $t_k^\star$ in $s^\star$ such that: $S = \{E_1, \ldots, E_m\}$ with $E_k = (Y_k, t_k^\star)$ for $1 \leq k \leq m$. We will extensively use this canonical structure in the following sections. The main reason for such trick is that we need to combine clustering techniques on sets, i.e., $Y_k = \{e_{k1}, \ldots, e_{km}\}$, with temporal analysis on sequences, i.e., s, and we have to go back and forth between both concepts. This idea is at the heart of our contribution.

3 The problem of building optimal summaries

We have previously reformulated time sequence summarization using data clustering terminology. In this section, under this new definition of a summary, we formalize the time sequence summarization task as a clustering activity where an objective function needs to be optimized. This objective function is expressed as a cost that takes into account the content and the time associated to events to cluster. For this purpose, we adopt a constructive methodology where we provide elementary operators to iteratively transform one sequence into another. The more iterations the less similar sequences become. The objective of the clustering problem is then to minimize the global cost for transforming a sequence into another until the summary is obtained. Our constructive methodology is largely inspired from usual edit distances [Ham50, Lev65] between structures such as strings, genes [PWP08] or trees, and is adapted to our summarization problem.

We give in Section 3.1 the basic concepts necessary to build the elementary clustering operation. Then, we formally define in Section 3.2 the elementary clustering operation itself and its associated cost to compare two time sequence summaries. Equipped with all the necessary material, we formally state the problem of building optimal time sequence summaries in Section 3.3.

3.1 Preliminaries

Here are introduced both content and temporal components of the elementary clustering operation used for defining the objective function that needs to be optimized in our clus-

tering problem.

3.1.1 Illustrative example

We illustrate all concepts and properties introduced hereafter by reusing the running example provided in Chapter 2. For commodity, we recall in Table 3.1 the time sequence of events extracted from conference proceedings. Each event is a publication whose topic is described thanks to descriptors taken from the *topic* domain taxonomy also recalled in Figure 3.3.

Author	Event descriptors	Time
N. Koudas	x_1={Datastreams, Aggregation}	JUN05
	x_2={Datastreams, Top-k query}	AUG06
	x_3={Top-k query}	AUG06
	x_4={Top-k query}	SEP06
	x_5={Join query, Selection query}	SEP06
	x_6={Clustering}	SEP07

Table 3.1: Time sequence of events in conference proceedings

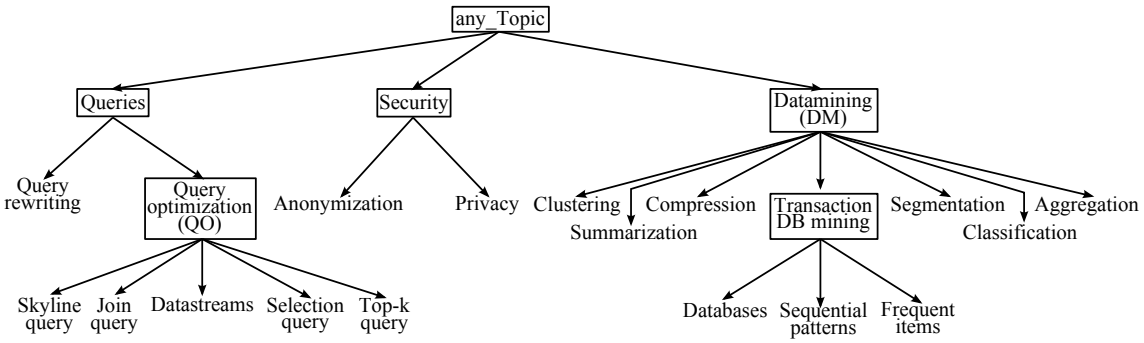


Figure 3.3: Taxonomy for the *topic* descriptive domain

3.1.2 Event content representation

When partitioning a set of data objects, it is necessary to evaluate the similarity between objects from content view point. Traditionally, a good partition is obtained when objects are grouped while minimizing intra-class inertia and maximizing inter-class inertia, i.e., form *dissimilar* groups of *similar* objects. In time sequence summarization, since events are described by a set of categorical descriptors, the general notion of inertia is difficult to grasp. Usual techniques rely on properties of contiguous domains to define distance-based measurement functions, e.g., the weighted total, average distance between pairs of objects, etc.. These metrics defined for numerical values can not apply, or poorly apply, to categorical values. As a reminder, it is for instance not trivial to define the *average* between descriptors "few" and "plenty".

Here, we define the *distance* between two sets of event descriptors x_i and x_j as the number of operations necessary to represent x_i and x_j at the level of abstraction where they become strictly identical. Intuitively, two different sets of event descriptors have a common representation if they are substituted by ancestors at a higher level of taxonomy. This common representation is also known in literature on ontologies as the *Most Specific Common Subsumer* (MSCS) or the *Lowest Super-Ordinate* (LSO). We present in Definition 3.4 the

basic concept of MSCS of two event descriptors and in Definition 3.5 the cost associated to the MSCS operation. This concept relies on the idea of *Atomic Generalization Operation* (AGO) defined in Definition 3.3. An AGO allows to represent an event descriptor at a superior level of abstraction using a taxonomy, i.e., by rewriting the descriptor with its ancestor in the taxonomy. The cost associated to each AGO is also given in Definition 3.3 and simply corresponds to 1 if a descriptor is successfully rewritten into its ancestor in the taxonomy. For instance, in the *topic* domain taxonomy, it takes two AGOs for descriptor "Frequent itemset" to be generalized into "Datamining"; This operation then comes with a cost of 2.

Definition 3.3 (Atomic generalization operator (AGO) Gen**)**
Let d be an event descriptor taken from descriptive domain D_A and suppose descriptive domain D_A is structured into taxonomy H_A. The atomic generalization operation, denoted $Gen(d)$, that rewrites d into its ancestor in H_A is defined as follows:

$$\begin{array}{rccl} Gen: & D_A & \longrightarrow & D_A \\ & d & \longmapsto & Gen(d) = \left\{ \begin{array}{ll} \text{“any\_}A\text{”}, & \text{if } d = \text{“any\_}A\text{”} \\ d', & \text{otherwise} \end{array} \right. \end{array}$$

where $d' \in D_A,\ d \prec_A d'$ and $\nexists d'' \in D_A,\ d \prec_A d'' \prec_A d'$. The cost associated to an atomic generalization operation is denoted $\mathcal{C}_{Gen}$ and defined as follows:

$$\mathcal{C}_{Gen}(d) \quad = \quad \left\{ \begin{array}{ll} 0, & \text{if } d = \text{“any\_}A\text{”} \\ 1, & \text{otherwise} \end{array} \right.$$

Definition 3.4 (Most specific common subsumer $mscs$**)**
Let d_i and d_j be two event descriptors taken from descriptive domain D_A and suppose descriptive domain D_A is structured into taxonomy H_A. The most specific common subsumer (MSCS) of descriptors d_i and d_j is denoted $d = mscs(d_i, d_j) \in H_A$ and is defined as follows:

$$\begin{array}{c} \forall (d_i, d_j) \in D_A^2,\ d = mscs(d_i, d_j) \ \Leftrightarrow\ \big((d_i \prec_A d) \ \vee\ (d_i = d)\big) \ \wedge \\ \big((d_j \prec_A d) \ \vee\ (d_j = d)\big) \ \wedge\ \big(\nexists d' \in D_A, (d_i \prec_A d') \ \wedge\ (d_j \prec_A d') \ \wedge\ (d' \prec_A d)\big) \end{array}$$

Note that the mscs operator is symmetric, i.e., $mscs(d_i, d_j) = mscs(d_j, d_i)$.

Definition 3.5 (MSCS cost)
The cost for representing two descriptors d_i and d_j, taken in a single descriptive domain D_A, by their MSCS is denoted $\mathcal{C}_{mscs}(d_i, d_j)$ and defined as follows:

$$\mathcal{C}_{mscs}(d_i, d_j) = N_i + N_j$$

where N_i is the length of the path from d_i to d in H_A, i.e., $N_i = \ell(d_i, d)$, and where N_j is the length of the path from d_j to d in H_A, i.e., $N_j = \ell(d_j, d)$. Note that the MSCS cost is symmetric, i.e., $\mathcal{C}_{mscs}(d_i, d_j) = \mathcal{C}_{mscs}(d_j, d_i)$, and the closer $\mathcal{C}_{mscs}(d_i, d_j)$ to 0, the more related d_i and d_j. The situation where $\mathcal{C}_{mscs}(d_i, d_j) = 0$ occurs when $d_i = d_j$.

Suppose d is the MSCS of the pair of descriptors (d_1, d_2), i.e., $d = mscs(d_1, d_2)$. Since we denote by $N_i = \ell(d_i, d)$ the length of the path from d_i to d in H_A, we can also say that d_i requires N_i atomic generalization operations (AGO) to be generalized into d. Hence, the *distance* between two descriptors d_1 and d_2 is evaluated as the sum of all AGOs necessary for representing both event descriptors d_1 and d_2 into their MSCS d. This distance can also be understood as the shortest path, in terms of number of edges, from d_i to d_j in taxonomy H_A. In literature on semantic distances, this generalization cost is comparable

to Rada et al's [RMBB89] simple edge counting method. We discuss different semantic distances available in the literature in Section 7.

Definition 3.6, 3.7 and 3.8 are generalizations of the MSCS concept. These generalizations are necessary since two events in a time sequence can be defined thanks to descriptors taken from multiple descriptive domains and several descriptors from a same descriptive domain could be used to describe an event. Therefore, Definition 3.6 generalizes the basic notion of MSCS to a set of descriptors taken from one single descriptive domain. Definition 3.7 generalizes MSCS to two sets of descriptors where descriptors are taken from multiple descriptive domains. Definition 3.8 is the most general definition and defines MSCS for multiple sets of descriptors where descriptors are taken from multiple descriptive domains. These properties will be helpful in Section 4 for computing the generalization cost for clustering multiple events.

One should note that the *mscs* operator has some nice properties, i.e., it is symmetric, reflexive and associative. These properties are recapitulated in Property 3.1. These properties are important for defining the MSCS costs for the generalized definitions of MSCS.

Definition 3.6 (Generalized MSCS - form 1)
We are given a set $x = \{d_1, \ldots, d_m\}$ of m event descriptors taken from one single descriptive domain D_A, the MSCS of set x is $d \in H_A$, denoted $d = mscs(x)$, defined as follows:

$$d = mscs(d_1, mscs(x - \{d_1\}))$$

where $\forall d_j \in D_A, mscs(d_j, \emptyset) = d_j$. The cost for representing x by its MSCS, denoted $\mathcal{C}_{mscs}(x)$, is defined as the sum of the cost for representing each descriptor d_i by its MSCS:

$$\mathcal{C}_{mscs}(x) = \sum_{i=1}^{m} \mathcal{C}_{mscs}(d_i, mscs(x))$$

Definition 3.7 (Generalized MSCS - form 2)
We are given two sets of descriptors $x = \{d_1, \ldots, d_m\}$ and $x' = \{d'_1, \ldots, d'_n\}$ of m and n event descriptors, respectively. Suppose x is partitioned into M subsets x_A, where every $x_A \subseteq D_A$ and $x = \bigcup_{A \in \mathcal{A}} x_A$. Also suppose x' is partitioned into N subsets x'_A, where every $x'_A \subseteq D_A$ and $x' = \bigcup_{A \in \mathcal{A}} x'_A$. The generalized MSCS of two sets of descriptors, where descriptors are taken from several different descriptive domains, is defined as follows:

$$\begin{array}{l} mscs(x, x') = \bigcup_{A \in \mathcal{A}} mscs(x_A, x'_A) \text{ where} \\ \forall A \in \mathcal{A}, mscs(\emptyset, \emptyset) = \emptyset \text{ and} \\ \forall A \in \mathcal{A}, mscs(x_A, \emptyset) = mscs(\emptyset, x_A) = \{\text{“}any\_A''\} \text{ and} \\ \forall A \in \mathcal{A}, mscs(x_A, x'_A) = \begin{cases} x_A, & \text{if } x_A = x'_A \\ mscs(x_A \cup x'_A), & \text{otherwise} \end{cases} \end{array}$$

The total cost induced for generalizing x and x' into their MSCS is defined as follows:

$$\begin{array}{l} \mathcal{C}_{mscs}(x, x') = \sum_{A \in \mathcal{A}} \mathcal{C}_{mscs}(x_A, x'_A) \text{ where} \\ \forall A \in \mathcal{A}, \mathcal{C}_{mscs}(\emptyset, \emptyset) = 0 \text{ and} \\ \forall A \in \mathcal{A}, \mathcal{C}_{mscs}(x_A, \emptyset) = \mathcal{C}_{mscs}(\emptyset, x_A) \text{ and} \\ \forall A \in \mathcal{A}, \mathcal{C}_{mscs}(\emptyset, x_A) = \mathcal{C}_{mscs}(x_A \cup \{\text{“}any\_A''\}) \text{ and} \\ \forall A \in \mathcal{A}, \mathcal{C}_{mscs}(x_A, x'_A) = \begin{cases} 0, & \text{if } x_A = x'_A \\ \mathcal{C}_{mscs}(x_A \cup x'_A), & \text{otherwise} \end{cases} \end{array}$$

Definition 3.8 (Generalized MSCS - form 3)
We are given n sets of event descriptors $x_1, \ldots, x_n$ where event descriptors are taken from multiple descriptive domains. We denote by X the set of all sets of event descriptors, i.e., $X = \{x_1, \ldots, x_n\}$. The generalized MSCS of X is defined as follows:

$$mscs(X) = mscs(x_1, mscs(X - x_1))$$

The total cost induced for generalizing X into its MSCS is defined as follows:

$$\mathcal{C}_{mscs}(X) = \sum_{i=1}^{n} \mathcal{C}_{mscs}(x_i, mscs(X))$$

For simplicity, in the rest of this chapter, these costs will be interchangeably called the MSCS cost *or the* generalization cost.

Property 3.1 (Properties of the *mscs* operator)
Given two sets of descriptors $x = \{d_1, \ldots, d_m\}$ and $x' = \{d'_1, \ldots, d'_n\}$ of m and n event descriptors, respectively, taken from descriptive domains D_A in $\mathcal{A}$, suppose each descriptive domain D_A is organized into taxonomy H_A. The mscs operator has the following properties:

- *Symmetry:* $mscs(x, x') = mscs(x', x)$
- *Reflexivity:* $mscs(x, x) = x$
- *Associativity:* $\forall x'' \in \mathcal{P}(\Omega),\ mscs(mscs(x, x'), x'') = mscs(x, mscs(x', x''))$

Proof 2 (Proof of Property 3.1)
Symmetry and reflexivity: By construction these two properties are verified.
Associativity: Supposing $mscs(x, x') = x^\star$, we have the following equalities:

$$mscs(mscs(x, x'), x'') = mscs(x^\star, x'') \text{ where } \forall d \in x \cup x',\ \exists d^\star \in x^\star,\ d \prec_A d^\star$$
$$mscs(mscs(x, x'), x'') = x^+ \text{ where } \forall d \in x^\star \cup x'',\ \exists d^+ \in x^+,\ d \prec_A d^+$$

On the other hand, supposing $mscs(x', x'') = x^\dagger$ we have the following equalities:

$$mscs(x, mscs(x', x'')) = mscs(x, x^\dagger) \text{ where } \forall d \in x' \cup x'',\ \exists d^\dagger \in x^\dagger,\ d \prec_A d^\dagger$$
$$mscs(x, mscs(x', x')) = x^\diamond \text{ where } \forall d \in x \cup x^\dagger,\ \exists d^\diamond \in x^\diamond,\ d \prec_A d^\diamond$$

Therefore, $\forall d \in x \cup x' \cup x''$, suppose d has a MSCS in x^+ and in $x^\diamond$, i.e., $\exists d^+ \in x^+, d \prec_A d^+$ and $\exists d^\diamond \in x^\diamond, d \prec_A d''$. Suppose $d^+ \neq d^\diamond$. Then either $d^+ \prec_A d^\diamond$ or $d^\diamond \prec_A d^+$. If the situation $d^+ \prec_A d^\diamond$ occurs, then by the definition of a MSCS, $d^\diamond$ is not the MSCS of $x \cup x' \cup x''$. If the situation $d^\diamond \prec_A d^+$ occurs, then by the definition of a MSCS, d^+ is not the MSCS of $x \cup x' \cup x''$. In any way, we proved that $d^+ = d^\diamond$ and by extent we proved Property 3.1. □

3.1.3 Rearrangement of events on the timeline

The second component of the objective function relies on the notion of *proximity* between events on the timeline. The novelty in time sequence summarization is the fact that events are timestamped and chronologically ordered. As a reminder, we assume two events need to be *similar* and *close* on the timeline to be candidates for clustering. Suppose the two events e_i and e_j are to be grouped together in a sequence S: A mechanism to virtually rearrange events e_i and e_j on the timeline, so the grouping can occur, needs to

be defined. In addition, a cost should be associated to the event rearrangement operated. Hereafter, since we consider on the one side clusters, i.e., collections of events, and on the other side sequences, i.e., lists of events, we will extensively use the canonical second-order time sequence notation introduced earlier. This notation is important for keeping the consistency of all notations used.

Under our time sequence summarization problem reformulation, we assume that two events $e_i = (x_i, t_i)$ and $e_j = (x_j, t_j)$ are eligible for clustering i.f.f. e_i and e_j satisfy two conditions: (i) x_i and x_j are *similar* and (ii) times of occurrence t_i and t_j are *close* on the timeline. The notion of *closeness* on the timeline can be expressed by different manners: (i) the time difference between t_i and t_j can be compared to a given threshold w, i.e., $t_j - t_i \leq w$, (ii) the number of events that separates (x_i, t_i) and (x_j, t_j) is lower than a given threshold, etc.. Here, we provide a simple way of evaluating the temporal proximity of two events by means of the number of events between them.

In fact, to support a constructive definition of the summarization problem, we decide that two candidate events for clustering are *close* on the timeline when they are *contiguous* on the timeline. This notion of *contiguity* is defined in Definition 3.9. For instance, in Example 3.1, events ($\{e_3\}$,AUG06) and ($\{e_4\}$,SEP06) are contiguous but events ($\{e_3\}$,AUG06) and ($\{e_6\}$,SEP07) are not contiguous on the timeline. This example is true even when the notion of *closeness* is chosen as a time duration w, for example w=1 month. Indeed, SEP06-AUG06=1 month and events ($\{e_3\}$,AUG06) and ($\{e_4\}$,SEP06) are contiguous; However, SEP07-AUG06=12 months and events ($\{e_3\}$,AUG06) and ($\{e_6\}$,SEP07) are not contiguous.

Definition 3.9 (Contiguous events)
Given s a time sequence of events in $\mathcal{S}(\Omega)$, let $\chi_F(s) = (\Pi(s), s^\star)$ be its summary and let S be the canonical second-order time sequence built from the summary $\chi_F(s)$, i.e., $S = \{E_1, \ldots, E_m\}$ with $E_k = (Y_k, t_k^\star)\}$ and $1 \leq k \leq m$. Given two events E_i and E_j in S, with $1 \leq i < j \leq m$, event E_i is said to be contiguous *to event E_j (and equivalently, "E_j is contiguous to E_i" or "E_i and E_j are contiguous") i.f.f.:*

$$\nexists u \in \mathbb{N}, \text{ such that } E_u = (Y_u, t_u^\star) \in S \text{ and } t_i^\star < t_u^\star < t_j^\star$$

In practice, two events are clustered by a simple set union operation. However, we believe this operation should be detailed and corresponds to a series of virtual *Atomic Permutation Operations* (APOs) of events on the timeline until the events become contiguous. Hence, we provide an operator denoted τ, that virtually rearranges events on the timeline so they become contiguous by a series of APOs. By rearranging events on the timeline in such way, operator τ is responsible for degrading the temporal accuracy of the sequence.

The atomic permutation operator is denoted $\sigma_{i^-}(s)$ and defined as follows:

$$\sigma_{i^-}(s) = \begin{cases} s, & \text{if } i = 1 \\ \langle e_1, \ldots, e_{i-2}, e_i, e_{i-1}, \ldots, e_m \rangle, & \text{if } 1 < i \leq n \end{cases}$$

Note that we could also define $\sigma_{i^+}(s)$ as the opposite permutation operation. However, for simplicity, we will focus only on the operator σ_{i^-}. Following up this choice, we lighten the notation and equivalently denote $\sigma_i = \sigma_{i^-}$. Also, when an event e_i undergoes k successive APOs σ_i, the operation is called a *k-permutation* and is denoted $\sigma_i^k(s)$.

The definition of the atomic permutation operator allows us to give an alternate definition of contiguous events. Intuitively, two events E_i and E_j in a canonical second-order time sequence S are contiguous events in S, if when E_j is permutated and then E_i is permutated, the resulting permutated sequence is identical to S. For instance, suppose $S = \langle E_1, E_2, E_3 \rangle$ where events E_2 and E_3 are clearly contiguous events. Notice that $\sigma_3(S) = \langle E_1, E_3, E_2 \rangle$

and $\sigma_2(\sigma_3(S)) = \langle E_1, E_2, E_3 \rangle$, i.e., $\sigma_2 \circ \sigma_3(S) = S$. On the other hand, events E_1 and E_3 are not contiguous events. Notice that $\sigma_1(\sigma_3(S)) = \langle E_1, E_3, E_2 \rangle$, i.e., $\sigma_1(\sigma_3(S)) \neq S$. We formalize this alternate definition of contiguous events in Definition 3.10. This definition is useful for proving Property 3.2.

Definition 3.10 (Contiguous events - Alternate definition)
Given s a time sequence of events in $\mathcal{S}(\Omega)$, let $\chi_F(s) = (\Pi(s), s^\star)$ be its summary and let S be the canonical second-order time sequence built from the summary $\chi_F(s)$, i.e., $S = \{E_1, \ldots, E_m\}$ with $E_k = (Y_k, t_k^\star)\}$ and $1 \leq k \leq m$. Given two events E_i and E_j in S, with $1 \leq i < j \leq m$, event E_i is said to be contiguous *to event E_j (and equivalently, "E_j is contiguous to E_i" or "E_i and E_j are contiguous") i.f.f.:*

$$\sigma_i \circ \sigma_j(S) = S$$

Property 3.2 (Existence of a unique event rearrangement)
Given s a time sequence of events in $\mathcal{S}(\Omega)$, let $\chi_F(s) = (\Pi(s), s^\star)$ be its summary and let S be the canonical second-order time sequence built from the summary $\chi_F(s)$, i.e., $S = \{E_1, \ldots, E_m\}$ with $E_k = (Y_k, t_k^\star)$ and $1 \leq k \leq m$. Given two events E_i and E_j in S, with $1 \leq i < j \leq m$, there exists a unique k-permutation σ_i^k with $0 \leq k \leq m-2$, so that E_i and E_j are contiguous events in $\sigma_j^k(S)$.

Proof 3 (Proof of Property 3.2)

Existence:

Suppose $S = \langle E_1, \ldots, E_i, E_j, \ldots, E_m \rangle$. Trivially, E_i and E_j are contiguous events (and require 0 permutation).

Suppose $S = \langle E_1, \ldots, E_i, \underbrace{\ldots}_{n \text{ events}}, E_j, \ldots, E_m \rangle$ with $n \geq 1$. Hence, we can hence write the following expressions:

$$\sigma_j(S) = \langle E_1, \ldots, E_i, \underbrace{\ldots}_{n\text{-}1 \text{ events}}, E_j, E_{j-1}, E_{j+1}, \ldots, E_m \rangle$$
$$\vdots$$
$$\underbrace{\sigma_j \circ \ldots \circ \sigma_j}_{n \text{ permutations}}(S) = \sigma_j^n(S) = \langle E_1, \ldots, E_i, E_j, E_{i+1}, \ldots, E_m \rangle$$

Consequently: $\sigma_i \circ \sigma_j(\sigma_j^n(S)) = \sigma_j^n(S)$ (Definition 3.10)

This is the condition for E_i and E_j to be contiguous. In this case, E_j required $k = n$ permutations to be contiguous to E_i. We proved the existence of the k-permutation.

Unicity:

Suppose $S = \langle E_1, \ldots, E_i, \ldots, E_j, \ldots, E_m \rangle$ and suppose there exists at least two k-permutations of E_j in S, denoted $\sigma_j^{k_1}$ and $\sigma_j^{k_2}$ with $k_1 < k_2$, so that E_j becomes contiguous to E_i. We can hence write the following expressions:

$$\sigma_j^{k_1}(S) = \langle E_1, \ldots, E_i, E_j, E_{i+1}, \ldots, E_m \rangle$$
$$As\ k_2 > k_1, \qquad \sigma_j^{k_2}(S) = \underbrace{\sigma_j \circ \ldots \circ \sigma_j}_{k_2 - k_1 \text{ permutations}} \quad \sigma_j^{k_1}(S)$$
$$Then\ \sigma_j^{k_2}(S) = \underbrace{\sigma_j \circ \ldots \circ \sigma_j}_{k_2 - k_1 \text{ permutations}} \quad (\langle E_1, \ldots, E_i, E_j, E_{i+1}, \ldots, E_m \rangle).$$
$$If\ i < (k_2 - k_1), \quad \sigma_j^{k_2}(S) = \langle E_j, \underbrace{E_1, \ldots}_{i \text{ events}}, E_i, \ldots, E_m \rangle.$$
$$If\ i \geq (k_2 - k_1), \quad \sigma_j^{k_2}(S) = \langle E_1, \ldots, E_j, \underbrace{\ldots}_{k_2 - k_1 \text{ events}}, E_i, \ldots, E_m \rangle.$$

In both cases, E_i and E_j are not contiguous in $\sigma_j^{k_2}(S)$ and necessarily $k_1 = k_2$. We proved the unicity of the k-permutation. □

Equipped with the notion of event permutation and the existence and unicity properties given in Property 3.2, we can finally give a formal definition of the event rearrangement operator τ in Definition 3.11. The cost induced by this event rearrangement operator is expressed as a loss of temporal accuracy and given in Definition 3.12.

Definition 3.11 (Event rearrangement operator $\tau_{i,j}$)
Given s a time sequence of events in $\mathcal{S}(\Omega)$, let $\chi_F(s) = (\Pi(s), s^\star)$ be its summary and let S be the canonical second-order time sequence built from the summary $\chi_F(s)$, i.e., $S = \{E_1, \ldots, E_m\}$ with $E_u = (Y_u, t_u^\star)\}$ and $1 \leq u \leq m$. The event rearrangement operator $\tau_{i,j}$, $1 \leq i < j \leq m$, virtually performs the unique k-permutation, with $k = j - 1 - 1$, on E_i and E_j such that they become contiguous in the canonical second-order time sequence S. $\tau_{i,j}$ operates as follows:

$$\tau_{i,j}(S) = \langle E_1, \ldots, E_i, E_j, E_{i+1}, \ldots, E_{j-1}, E_{j+1}, \ldots, E_n \rangle$$

Definition 3.12 (Event rearrangement cost $\mathcal{C}_\tau$)
Given s a time sequence of events in $\mathcal{S}(\Omega)$, let $\chi_F(s) = (\Pi(s), s^\star)$ be its summary and let S be the canonical second-order time sequence built from the summary $\chi_F(s)$, i.e., $S = \{E_1, \ldots, E_m\}$ with $E_u = (Y_u, t_u^\star)\}$ and $1 \leq u \leq m$. Let E_i and E_j be two events in S to rearrange. E_i and E_j require a k-permutation with $k = j - i - 1$ to become contiguous, so, the event rearrangement cost is defined as follows:

$$\mathcal{C}_\tau(i, j, s) = j - i - 1$$

One should note that this definition of the event rearrangement cost is no different from the temporal rearrangement penalty cost given in Definition 2.6 in Chapter 2. Here, we detailed this definition with all necessary concepts introduced in this chapter for rigorously defining the *Elementary Clustering Operator* (ECO).

Also, one should note that in Definition 3.12, $\mathcal{C}_\tau$ is a very first attempt to estimate the loss of temporal accuracy since the linearity of $\mathcal{C}_\tau$ does not penalize event rearrangements that require a large number of permutations. $\mathcal{C}_\tau$ could be improved and refined to reflect different time decay models. For instance, $\mathcal{C}_\tau$ could be expressed as an exponential function, which is most commonly used, or as a polynomial function, which is less common, of time. We present in Section 7 some interesting directions taken in time decay modeling that could be implemented. Anyway, the basic temporal cost function keeps the model consistent and does not yield any loss of generality in the approach.

3.2 Elementary Clustering Operator (ECO)

Previously, we showed that building a summary $\chi_F(s)$ from s comes down to partitioning s and providing a characterization function F to describe every cluster. Since we adopt a constructive approach to define the summarization problem, partitioning s is equivalent to iteratively grouping two clusters Y_i and Y_j in $\Pi(s)$. The process is initialized from a time sequence $s = \langle e_1, \ldots, e_n \rangle$ with the partition of singletons $\Pi(s) = \{\{e_1\}, \ldots, \{e_n\}\}$. Then, the elementary clustering operator f_{ECO}, defined in Definition 3.14, performs the grouping on a summary with n clusters and provides a summary with $n - 1$ clusters. The cost of f_{ECO}, denoted $\mathcal{C}_{\text{ECO}}$, is introduced in Definition 3.15. Intuitively, $\mathcal{C}_{\text{ECO}}$ is the MSCS cost for representing $Y_i \cup Y_j$ by its MSCS to which is added the cost to virtually rearrange Y_i and Y_j. We assume that the grouping operation, defined in Definition 3.13, is *free* of cost since it does not involve any update neither on the content nor on the timestamps of the sequence.

Definition 3.13 (Set union operator $\psi_{i,j}$)
Given S a collection of clusters, with $S = \{Y_1, \ldots, Y_n\}$ and where each cluster is a collection of objects, the set union operator is defined as follows:

$$\psi_{i,j}(S) = \langle Y_1, \ldots, Y_i = Y_i \cup Y_j, \ldots, Y_{j-1}, Y_{j+1}, \ldots, Y_n \rangle$$

Definition 3.14 (Elementary clustering operator f_{ECO})
Given s a time sequence of events in $\mathcal{S}(\Omega)$, let $\chi_F(s) = (\Pi(s), s^\star)$ be its summary. Let Y_i and Y_j be two elements in $\Pi(s)$. The elementary clustering operator f_{ECO} that groups Y_i and Y_j is defined as follows:

$f_{\text{ECO}}(i, j, \chi_F(s)) = \chi'_F(s) = (\Pi'(s), s^\star{}')$ *where*
$\Pi'(s) = \psi_{i,j}(\Pi(s)) = \{Y_i \cup Y_j, Y_1, \ldots, Y_{i-1}, Y_{i+1}, \ldots, Y_{j-1}, Y_{j+1}, \ldots, Y_n\}$ *and*
$s^\star{}' = \{F(Y),\ Y \in \Pi'(s)\}$

Definition 3.15 (Elementary clustering operation cost $\mathcal{C}_{\text{ECO}}$)
Given s a time sequence of events in $\mathcal{S}(\Omega)$, let $\chi_F(s) = (\Pi(s), s^\star)$ be its summary and let S be the canonical second-order time sequence built from the summary $\chi_F(s)$, i.e., $S = \{E_1, \ldots, E_m\}$ with $E_u = (Y_u, t^\star_u)\}$ and $1 \leq u \leq m$. Let Y_i and Y_j be two elements in $\Pi(s)$. The clustering cost induced by the ECO $f_{\text{ECO}}(i, j, \chi_F(s))$ is denoted $\mathcal{C}_{\text{ECO}}(i, j, \chi_F(s))$ and is defined as follows:

$$\mathcal{C}_{\text{ECO}}(i, j, \chi_F(s)) = \mathcal{C}_\tau(i, j, S) + \mathcal{C}_{mscs}(Y_i \cup Y_j) + 0$$

The last null term corresponds to cost of the grouping operation.

Finally, we propose to measure the overall cost of all elementary clustering operations necessary for computing a summary $\chi_F(s)$ from the raw sequence s. This cost is denoted $\mathcal{TC}_{f_{\text{ECO}}}(s, \chi_F(s)) = \sum \mathcal{C}_{\text{ECO}}(i, j, \chi_F(s))$. Roughly speaking, we define here a counting of successive elementary editions, on content and time, to transform a sequence s into its summary $\chi_F(s)$. Hence, this cost can be understood as the edit distance $d(s, \chi_F(s))$ between a sequence s and its summary $\chi_F(s)$.

Let us give an example of elementary clustering using our running example in Table 3.1. Suppose $\Pi(s) = \{Y_1, \ldots, Y_6\}$, with $Y_i = \{e_i\} = \{(x_i, t_i)\}$, e.g., $e_1 = (\{\text{Datastreams, Aggregation}\}, \text{JUN05})$, is the partition of singletons built from N. Koudas's time sequence of events. We compute the following elementary clustering operations:

- $f_{\text{ECO}}(3, 4, \Pi(s))$: This elementary clustering operation comes with a cost $\mathcal{C}_{\text{ECO}}(3, 4, \chi_F(s)) = 0$ as Y_3 and Y_4 are contiguous and $mscs(Y_3 \cup Y_4) = Y_3 = Y_4$ (no AGO needed).

- $f_{\text{ECO}}(3, 6, \Pi(s))$: This elementary clustering operation comes with a cost $\mathcal{C}_{\text{ECO}}(3, 6, \chi_F(s)) = 2 + (2+3) + 0 = 7$. Indeed, Y_3 and Y_6 require 2 APOs to become contiguous. In addition, in the *topic* domain taxonomy in Figure 3.3, the MSCS of "Top-k query" and "Clustering" is "any_Topic"; Hence, there are (i) a path of length 3 between "Top-k query" and "any_Topic" and (ii) a path of length 2 between "Clustering" and "any_Topic", which totals to 5 AGOs.

3.3 Problem statement

Using the elementary clustering operator defined in Section 3.2, we can state the problem of finding an *optimal time sequence summary* in Definition 3.16.

Definition 3.16 (Optimal Time Sequence Summarization Problem Statement)
Given s a time sequence of events in $\mathcal{S}(\Omega)$ and a dual characterization function F defined on $\mathcal{S}(\Omega)$, let ρ_{obj} be the desired compression ratio. The optimal time sequence summarization problem consists in building the summary $\chi_F^+(s)$ of s that satisfies:

1. $\rho(s, \chi_F^+(s)) \geq \rho_{obj}$, *and*
2. $\mathcal{TC}_{f_{\text{ECO}}}(s, \chi_F^+(s))$ *is minimum.*

Given a time sequence s; let F be a dual characterization function used to define summary $\chi_F(s)$. For the sake of simplicity, and without any loss of generality, we indistinctly adopt in the following the canonical second order sequence S to represent the summary $\chi_F(s)$. The search space for finding the optimal time sequence summary is a lattice $\mathcal{L}$ defined as follows:

- The lowest node in lattice $\mathcal{L}$ is S.
- The highest node in lattice $\mathcal{L}$ is denoted $\overline{S}$, where $\overline{S} = \{\bigcup_{Y \in S} Y\}$.
- Two nodes u and v in lattice $\mathcal{L}$ are linked by an arc i.f.f. there exists an elementary clustering operation such that $v = f_{\text{ECO}}(i, j, u)$ with $1 \leq i < j \leq n$ and having clustering cost $\mathcal{C}_{\text{ECO}}(i, j, u)$. An illustration of the search space is given in Figure 3.4.

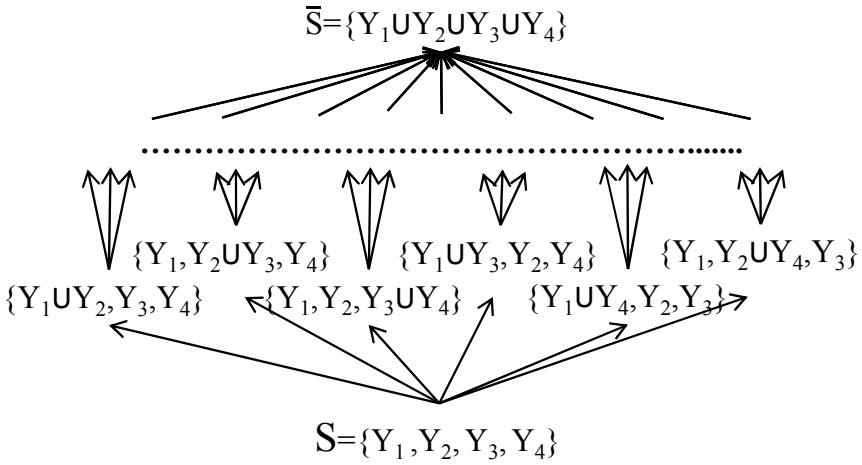


Figure 3.4: Search space of optimal time sequence summaries

The problem statement given here rises several legitimate questions listed as follows:

1. Does an optimal summary exist?
2. Is this optimal summary unique?
3. Is it always possible to compute an optimal summary?

Theorem 3.1 (Existence of a summary)
Given a time sequence s and a characterization function F; let ρ_{obj} be the desired compression ratio. There exists at least one time sequence summary $\chi_F(s)$ that achieves the compression ratio ρ_{obj}.

Sketch of the proof: The ECO is monotone in the cardinality of the summary: Each time we perform an ECO, the size of the summary decrements by 1. Hence, if we apply successively the ECO on a given summary, $[f_{\text{ECO}}(\_, \_, \chi_F(s))]^{N-1}$, with N the length of

s, the resulting summary has a partition $\Pi(s)$ with a single cluster equal to the set of events in s. It is exactly the upper bound in the lattice $\mathcal{L}$, denoted by $\overline{S}$. In this case, the compression ratio is maximal, i.e., $\rho(s, \overline{S}) = 1$. Hence, for every desired compression ratio $\rho_{obj} \in [0, 1]$, we can build at least summary $\overline{S}$ to meet the problem requirements. □

Contrarily, *unicity* of the answer to the problem of Definition 3.16 is unlikely observed. It is easy to see that different instantiations of an elementary clustering operation f_{ECO} could be associated to equal costs. For instance, the sequence (a, a, b, b) is reduced to (a, b, b) or (a, a, b), both with a cost equal to zero.

Theorem 3.1 states that it is always possible to achieve the desired compression ratio. In fact, as the compression ratio ρ defined in Definition 2.7 in Chapter 2 is connected to the number of events in the output time sequence summary, a time sequence summarization algorithm needs only to perform a finite number of elementary clustering operations to achieve the desired compression ratio ρ_{obj}. This minimum number of elementary clusterings is denoted min_{ECO} and detailed in Theorem 3.2.

Theorem 3.2 (Min. nb. of ECOs vs. desired compression ratio)
Given s a time sequence of events in $\mathcal{S}(\Omega)$ and a desired compression ratio ρ_{obj}, there exists a finite number of elementary clusterings operations, denoted min_{ECO}, necessary to achieve the desired compression ration ρ_{obj}. min_{ECO} is given by the following formula:

$$min_{\text{ECO}} = \lceil \rho_{obj} \times (|s| - 1) \rceil$$

Proof 4 (Proof of Theorem 3.2)
Theorem 3.2 is proved with the definition of the compression ratio. Given $\chi_F(s)$=$(\Pi(s), s^\star)$ the summary of s:

$$\rho_{obj} = 1 - \frac{|s^\star|-1}{|s|-1}$$
$$\rho_{obj} = \frac{|s|-1-|s^\star|+1}{|s|-1}$$
$$|s| - |s^\star| = \rho_{obj} \times (|s| - 1)$$

The quantity $|s| - |s^\star|$ represents the number of events that summary $s^\star$ has less w.r.t. the original sequence s. Hence, it also represents the number of elementary clustering operations performed to achieve ρ_{obj} since for each event less in $s^\star$ corresponds to one elementary clustering operation performed. □

Therefore, given a desired compression ratio ρ_{obj}, Theorem 3.2 implies that the optimal time sequence summaries are found in the upward closure of nodes at a distance of $min_{\text{ECO}} - 1$ elementary clustering operations from S in lattice $\mathcal{L}$. We denote by $\mathcal{Z}$ the set of time sequence summaries that can be generated with min_{ECO} elementary clustering operations. $\mathcal{Z}$ can also be understood as the set of summaries that contain exactly $|S| - min_{\text{ECO}}$ events.

4 Basic solution

The most basic solution to build an optimal time sequence summary from time sequence s is to generate the set of all candidate summaries $\mathcal{Z}$. Then, the optimal time sequence summary is the one obtained with the lowest total clustering cost, also called the *canonical cost*. Given a candidate summary S' in $\mathcal{Z}$, it is adamant to be capable of computing the canonical cost for partitioning S into S'. Since there exist multiple paths in lattice $\mathcal{L}$ from S to the candidate summary S', each path induces a different clustering cost. However, with the only knowledge of S and candidate summary S', computing the canonical cost

for generating S' from S is difficult and itself an optimization problem. A heuristic can be devised to approximate this canonical cost. We propose a greedy top-down algorithm for this purpose. Still, note that this basic approach to compute an optimal summary is natively combinatorial since we have first to enumerate $\mathcal{Z}$, that is an exponential task, and then, decide based on the lowest cost computation whether each candidate summary is the optimal solution.

The rest of this section is organized as follows. We detail in Section 4.1 the basic structural characteristics of clusters in a summary. These characteristics will be used for building the heuristic to approximate the canonical cost. We present our heuristic in Section 4.2. Finally, we put the pieces together and propose in Section 4.3 the N-TSS algorithm, and its performances are studied in Section 4.4.

4.1 Characteristics of clusters

In this section, we present some characteristics of summaries in lattice $\mathcal{L}$. The characteristics presented in this section will be useful for deriving an estimation of the canonical cost for producing a candidate summary S' from S. Hence given S and candidate summary S', each cluster $Y_i \in S'$ required exactly $|Y_i| - 1$ elementary clustering operations to be formed. Definition 3.15 states that the clustering cost for producing cluster Y_i depends on the number of atomic permutation operations necessary to group events into Y_i and the MSCS cost to represent all events in Y_i by a representative event. However, the definition of the MSCS cost given in Definition 3.8 allows us to state that this MSCS cost for cluster Y_i is constant and we denote this constant cost c_i.

As a consequence, the clustering cost for generating cluster Y_i only varies with the cost $\mathcal{C}_\tau$ necessary for rearranging events on the timeline, i.e., the number of atomic permutation operations. Let us present from our illustrative example the cost induced by two different summarizations: Suppose $S = \{\{e_1\}, \ldots, \{e_6\}\}$ and its summary is $S' = \{Y_1 = \{e_1, e_4, e_6\}, Y_2 = \{e_2, e_3, e_5\}\}$ where the MSCS cost of Y_1 is c_1 and the MSCS cost of y_2 is c_2. Here are two different sequences of ECOs that produce S' and their associated clustering cost:

1. The first algorithm χ_1 performs all ECOs necessary to generate Y_1 then performs all ECOs necessary to generate Y_2. The sequence of ECOs is broken down as follows:

   - $f_{\text{ECO}}(1, 4, S)$=S'_1=$\{Y_1$=$\{x_1, x_4\}, \{x_2\}, \{x_3\}, \{x_5\}, \{x_6\}\}$ and requires 2 APOs.
   - $f_{\text{ECO}}(1, 6, S'_1)$=$S'_2$=$\{Y_1$=$\{x_1, x_4, x_6\}, \{x_2\}, \{x_3\}, \{x_5\}\}$ and requires 3 APOs.
   - $f_{\text{ECO}}(2, 3, S'_2)$=$S'_3$=$\{Y_1$=$\{x_1, x_4, x_6\}, Y_2$=$\{x_2, x_3\}, \{x_5\}\}$ and requires 0 APO.
   - $f_{\text{ECO}}(2, 5, S'_3)$=$S'$ and requires 0 APO.

   The total clustering cost is thus $\mathcal{TC}_{\chi_1}$=$c_1 + c_2 + 2 + 3 + 0 + 0$=$c_1 + c_2 + 5$.

2. The second algorithm χ_2 performs all ECOs necessary to generate Y_2 then performs all ECOs necessary to generate Y_1. The sequence of ECOs is broken down as follows:

   - $f_{\text{ECO}}(2, 3, S)$=S'_1=$\{\{x_1\}, Y_2$=$\{x_2, x_3\}, \{x_4\}, \{x_5\}, \{x_6\}\}$ and requires 0 APO.
   - $f_{\text{ECO}}(2, 5, S'_1)$=$S'_2$=$\{\{x_1\}, Y_2$=$\{x_2, x_3, x_5\}, \{x_4\}, \{x_6\}\}$ and requires 1 APO.
   - $f_{\text{ECO}}(1, 4), S'_2$=$S'_3$=$\{Y_1$=$\{x_1, x_4\}, Y_2$=$\{x_2, x_3, x_5\}, \{x_6\}\}$ and requires 1 APO.
   - $f_{\text{ECO}}(2, 5, S'_3)$=$S'$ and requires 1 APO.

   The total clustering cost is thus $\mathcal{TC}_{\chi_2}$=$c_1 + c_2 + 0 + 1 + 1 + 1$=$c_1 + c_2 + 3$.

These simple examples of summarization show that there is a difference of 2 APOs between the clustering costs of χ_1 and χ_2. These two examples also show it is possible to lower bound the clustering cost for forming each cluster $Y_i \in S'$. Indeed, for any given cluster $Y_i \in S'$, a minimum of $m-1$ elementary clustering operations on S are required to form Y_i. The lower bound is attainable when exactly $m-1$ ECOs are performed successively to form y_i.

For instance, if $S = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}\}$ and $S' = \{Y_1 = \{x_1, x_4\}, Y_2 = \{x_2, x_3\}\}$, Y_1 requires one single ECO to be formed, i.e., $f_{\text{ECO}}(1, 4, S)$, and induces a clustering cost of $c_1 + 2$. However, if Y_2 is formed first, i.e., $f_{\text{ECO}}(2, 3, S) = S_1$ is performed (with clustering cost c_2+1), then followed by $f_{\text{ECO}}(1, 2, S_1)$, the actual clustering cost for forming Y_1 is c_1+1 and has required 2 ECOs. These observations allow us to characterize in Definition 3.17 the lower bound of the clustering cost for forming any cluster $Y_i \in S'$.

Definition 3.17 (Clustering cost lower bound)
Given s a time sequence of events in $\mathcal{S}(\Omega)$, let $\chi_F(s) = (\Pi(s), s^\star)$ be its summary and let S be the canonical second-order time sequence built from the summary $\chi_F(s)$, i.e., $S = \{E_1, \ldots, E_m\}$ with $E_u = (Y_u, t_u^\star)\}$ and $1 \leq u \leq m$. Let S' be a candidate summary in $\mathcal{Z}$. For any $Y_i \in S'$ with $Y_i = \{e_{i_1}, \ldots, e_{i_m}\}$, the clustering cost induced to form cluster Y_i with exactly $|Y_i| - 1$ successive ECOs is denoted $CLB(Y_i)$. This cost is lower bound by the following formula:

$$\begin{cases} CLB(Y_i) = 0, & \text{if } |Y_i| = 1 \\ CLB(Y_i) = (i_m - i_1 - 1) - (|Y_i| - 2) & \text{otherwise} \end{cases}$$

Let us explicit the rationale of Definition 3.17. Events e_{i_1} and e_{i_m} in cluster Y_i can be understood as the boundary events of cluster Y_i in s, i.e., all events in cluster Y_i are temporally located in between events e_{i_1} and e_{i_m} on the timeline. There exists a number of events e_j in between events e_{i_1} and e_{i_m} that do not belong to cluster Y_i. In fact, there are exactly $(i_m - i_1 - 1) - (|Y_i| - 2)$ events temporally located in between events e_{i_1} and e_{i_m} that do not belong to cluster Y_i. Therefore, forming cluster Y_i from S in exactly $|Y_i| - 1$ ECOs requires to reduce this number of intermediary events to zero. This can be achieved by performing exactly $(i_m - i_1 - 1) - (|Y_i| - 2)$ APOs and $|Y_i| - 1$ grouping operations.

From practical view point, this lower bound can be achieved by means of an algorithm that iteratively clusters Y_i's *boundary* event. Informally, one starts by adding the cost for clustering in S the most recent cluster in Y_i, i.e., e_{i_m}, with its second most recent event, i.e., e_{i_m-1}. The cluster obtained is denoted Y_{i_m-1}. The process is repeated and cluster Y_{i_m-1} is clustered with the third most recent event in Y_i, and so on until the least recent event e_{i_1} in Y_i is reached. Figure 3.5 gives an example of how this algorithm operates. Here, the cluster to form is $Y_1 = \{e_1, e_5, e_8, e_{10}\}$ where the boundary events are e_1 and e_{10}, i.e., $e_{i_m} = e_{10}$. Y_1's clustering cost lower bound is computed in 3 iterations and gives a total cost of 6. Note that Y_1 is formed in exactly $|Y_1| - 1$ operations and the resulting cost perfectly corresponds to $CLB(Y_1) = (10 - 1 - 1) - (4 - 2) = 6$. Also, visually, this cost corresponds to the number of events in between e_1 and e_{10} that are not contained in Y_1.

This notion of clustering cost lower bound for forming a cluster Y allows us to characterize the temporal relatedness of events in Y. Indeed, Property 3.3 shows that if the number of events in a cluster Y is smaller than $CLB(Y)$, some events in Y must be contiguous. Since contiguous events do not induce an event rearrangement cost, this property is most interesting for deciding the order of the ECOs necessary to generate S'.

Property 3.3 (Contiguous events in cluster Y vs. $CLB(Y)$ vs. $|Y|$)
Given s a time sequence of events in $\mathcal{S}(\Omega)$, let $\chi_F(s) = (\Pi(s), s^\star)$ be its summary and let S be the canonical second-order time sequence built from the summary $\chi_F(s)$, i.e.,

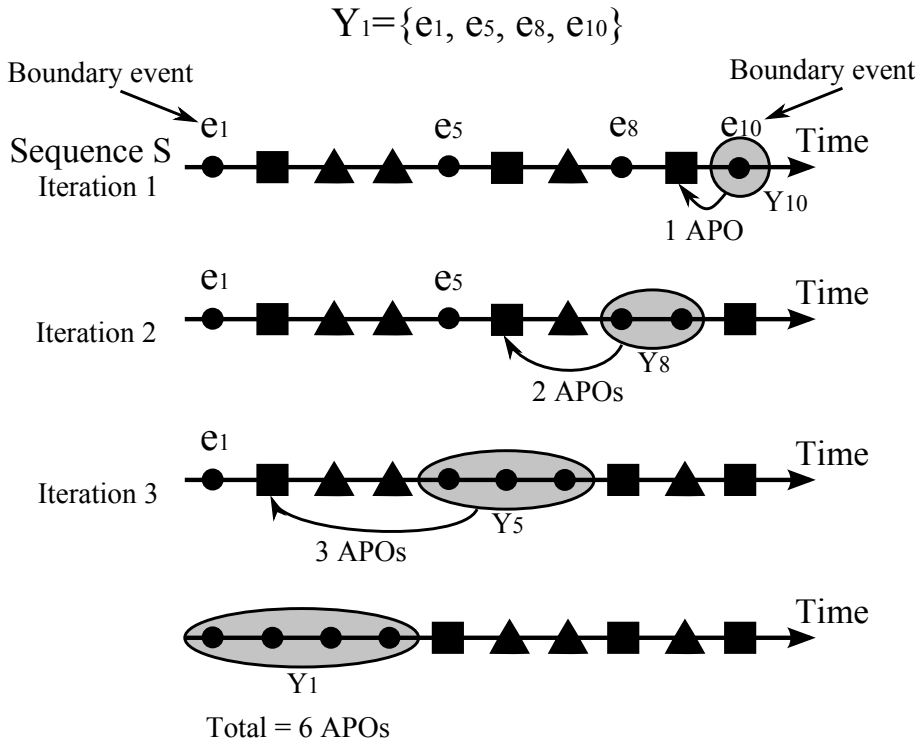


Figure 3.5: Example of CLB computation for forming Y_1

$S = \{E_1, \ldots, E_m\}$ with $E_u = (Y_u, t_u^\star)\}$ and $1 \leq u \leq m$. Let S' be a summary in $\mathcal{Z}$. The following property is true:

$$\forall Y \in S',\ (|Y| > 1)\ \wedge\ (CLB(Y) < |Y| - 1)\ \Rightarrow$$
there exist at least $\big(|Y| - CLB(Y) - 1\big)$ *couples of contiguous events in* Y

Consequently, there exist at least $|Y| - CLB(Y) - 1$ ECOs of events in Y that require zero APO.

Proof 5 (Proof of Property 3.3)
Previously, we described the quantity $CLB(Y)$ as the number of events in s in between the boundary events of cluster Y, i.e., events e_1 and e_m, that are not contained in Y. We denote by $\overline{CLB}(Y)$ this set of events. Suppose $CLB(Y) = |Y| - 1$. This situation can only be true when events in Y and $\overline{CLB}(Y)$ alternate on the timeline. For instance, suppose $s = \langle e_1, e_2, e_3, e_4, e_5 \rangle$ and $Y_1 = \{e_1, e_3, e_5\}$, then, $CLB(Y_1) = 2$ and $\overline{CLB}(Y) = \{e_2, e_4\}$.

Therefore, when $CLB(Y) = |Y| - 1 - 1$, there exists at least one pair of events in Y that are contiguous. Recursively, when $CLB(Y) < |Y| - 1$ there exist at least $|Y| - CLB(Y) - 1$ pairs of events in Y that are contiguous. □

4.2 Greedy canonical clustering cost evaluation algorithm

Equipped with Property 3.3, we can present the greedy algorithm that computes an approximation of the canonical clustering cost for producing a candidate summary S' from a sequence S. As mentioned earlier, since the MSCS cost for forming each cluster $Y \in S'$ is constant, the total cost for forming Y mainly relies on the order in which events are clustered (or equivalently, the order in which events are rearranged). The intuition behind the algorithm is that the lowest clustering cost is obtained when the *cheapest* elementary clustering operations are performed first.

Also, we observe that when Y's CLB is low and the number of events in Y is important, many events in Y must be contiguous on the timeline. This observation allows us to introduce the *density* index of a cluster Y. The idea of the density index is to indicate how dense on the timeline are events within a cluster Y. The density index of a cluster is given in Definition 3.18 and used as an indicator for our greedy algorithm.

Definition 3.18 (Density index of a cluster)
Given s a time sequence of events in $\mathcal{S}(\Omega)$, let $\chi_F(s) = (\Pi(s), s^\star)$ be its summary and let S be the canonical second-order time sequence built from the summary $\chi_F(s)$, i.e., $S = \{E_1, \ldots, E_m\}$ with $E_u = (Y_u, t^\star_u)\}$ and $1 \leq u \leq m$. The density index of each cluster Y_i in S, such that $|Y_i| \geq 2$, is denoted r_i and computed by the following ratio:

$$r_i = \frac{CLB(Y_i)}{|Y_i|}, \text{ with } 0 \leq r_i \leq 1$$

The closer r_i to zero, the more events $e_{i,j} \in Y_i$ are close on the timeline and require less APOs to be clustered.

We chose to define the density index as a ratio based on the CLB measure due to its relationship with the notion of contiguity of events on a timeline. Other more common measures could have been used to compute the density of the cluster, e.g., entropy-based measures.

The intuition behind our greedy algorithm is to proceed in a top-down fashion and guess from candidate summary S' which was possibly the min^{th}_{ECO} (or equivalently, the last) elementary clustering operation performed that produced S'. Since, r_i gives the density of clusters Y_i, we choose the cluster whose density index is the closest to zero: The closer to zero, the more chances events in the cluster are close on the timeline. Let us denote by Y_j this cluster. Suppose, $f_{\text{ECO}}(j,k,u)$ was the min^{th}_{ECO} elementary clustering operation performed to produce S', i.e., $S' = f_{\text{ECO}}(j,k,u)$ where u is the intermediary summary in lattice $\mathcal{L}$ used for clustering and Y_j and Y_k the clusters grouped. Suppose events $e_{j,i}$ and $e_{j,k}$ are the couple of events in Y_j that induces the lowest clustering cost, or equivalently, the couple of events in Y_j that induces the smallest event rearrangement cost, in S. Therefore, we assume that $Y_k = \{e_{j,k}\}$ and split $e_{j,k}$ from cluster Y_j to rebuild intermediary summary u. This process is repeated min_{ECO} times so that at the last iteration, the intermediary summary u is S. We present in Algorithm 2 the pseudo-code of our algorithm.

First, the cluster density index r_i is computed for each cluster $Y_i \in S'$ (Line 6 to Line 13). Then, we choose cluster $Y_j \in S'$ that has the smallest density index (Line 14). Property 3.3 indicates that if $CLB(Y_j) < |Y_j| - 1$ there exist $|Y_i| - 1 - CLB(Y_i)$ ECOs of events in cluster Y_j that can be performed with no APO cost. Any one of these operations can be performed first. Otherwise, i.e, there are no contiguous events, we choose the couple of events $e_u \in Y_j$ and $e_v \in Y_j$ that minimizes the event rearrangement cost $\mathcal{C}_\tau(u,v,S)$ (Line 15 to Line 16). The total clustering cost is increased with $\mathcal{C}_\tau(u,v,S)$ (Line 17). This process is repeated min_{ECO} times. Note that sequence S is updated by deleting event e_v from S. This operation is permitted, since, from event rearrangement point of view, deleting event e_v is equivalent to grouping event e_v with event e_u.

4.3 N-TSS: Naive Time Sequence Summarization approach

Now that we have defined the function that computes an approximation of the canonical clustering cost, we can give the details of our naive algorithm called N-TSS. N-TSS is an Exhaustive Enumeration (EE) approach, i.e., N-TSS generates all candidate summaries

and then determines which is the optimal summary. Algorithm 3 gives the pseudo-code of N-TSS. First, the set $\mathcal{Z}$ of all time sequence summaries possibly obtained with min_{ECO} elementary clustering operations is generated (Line 6). Then, for each candidate summary $S' \in \mathcal{Z}$, the approximate canonical cost to produce S' from S (Line 8) is computed and added to the MSCS cost of clusters in S'. If this cost is the lowest computed so far, it is recorded and S' is marked as the (local) optimal summary.

Algorithm 2 Greedy Canonical clustering Cost approximation function *G-CC*

1. INPUTS:
   $S = \{Y_1 = \{e_1\}, \ldots, Y_n = \{e_n\}\}$: Input sequence
   $S' = \{Y_1, \ldots, Y_m\}$, $Y_i = \{e_{i,1}, \ldots, e_{i,m_i}\}$: Summary
   min_{ECO}: Minimum number of ECOs to perform
2. OUTPUT: C: Approximate canonical clustering cost
3. LOCALS:
   R: Array of density ratios of size m
   C: Approximate canonical clustering cost
   min: Index of the cluster Y_i with minimum density index r_i

4. $C \leftarrow 0$ {// Initialize clustering cost}
5. **for** $cpt = 1$ to min_{ECO} **do**
6. **for** $i = 1$ to m **do**
7. {// Compute density index of each cluster Y_i}
8. **if** $(|Y_i| = 1)$ **then**
9. $R[i] \leftarrow -1$ {// Default value for singleton clusters}
10. **else**
11. $R[i] \leftarrow \frac{CLB(Y_i)}{|Y_i|}$
12. **end if**
13. **end for**
14. $min \leftarrow \text{argmin}_i(R[i])$ with $R[min] \neq -1$
15. Generate the set Y of all couples of events in Y_{min}, i.e.,
   $Y \leftarrow \{(e_i, e_j) \in Y_{min}^2, i < j\}$
16. Find $(e_u, e_v) \in Y$ such that
   $(e_u, e_v) \leftarrow \text{argmin}_{(e_i,e_j)\in Y}(\mathcal{C}_\tau(i, j, S))$
17. $C \leftarrow C + \mathcal{C}_\tau(u, v, S)$ {// Update clustering cost}
18. $S \leftarrow S - \{e_v\}$ {// Update sequences}
19. $Y_{min} \leftarrow Y_{min} - e_v$
20. **end for**
21. **return** C

4.4 Complexity

The N-TSS algorithm is an exhaustive and very costly approach from computational and memory usage view point. Indeed, step 6 in Algorithm 3 requires the Exhaustive Enumeration (EE) and storage in-memory of all candidates summaries. Given the number of elementary clustering operations to perform on a sequence S to achieve the desired compression ratio min_{ECO}, the number of clusters formed in S is $k = |S| - min_{\text{ECO}}$. Since, there are approximatively $\frac{k^n}{k!}$ ways of partitioning a set of n data points into k clusters [DH73], the computational and memory footprint of the N-TSS are exponential in $O(\frac{k^n}{k!})$ and $O(R \times \frac{k^n}{k!})$, respectively, where R is the memory necessary for storing a summary in-memory.

Algorithm 3 Naive Time Sequence Summarization algorithm *N-TSS*

1. INPUTS:
   $S = \{Y_1 = \{e_1\}, \ldots, Y_n = \{e_n\}\}$: Input sequence
   ρ_{obj}: Objective compression ratio
2. OUTPUT: S_{opt}: (Local) Optimal time sequence summary
3. LOCALS:
   min_{ECO}: Minimum number of ECOs to perform
   $\mathcal{Z}$: Set of candidate summaries
   $minCost$: Minimum clustering cost computed so far
   S_{opt}: Optimal time sequence summary identified so far

4. $min_{\text{ECO}} \leftarrow \lceil |S| \times (1 - \rho_{obj}) \rceil$
5. $minCost \leftarrow \infty$
6. Generate the (lattice) set $\mathcal{Z}$ with S and min_{ECO}
7. **for** $i = 1$ to $|\mathcal{Z}|$ **do**
8. {// Compute the approximate canonical clustering cost of S_i}
   $C \leftarrow$ G-CC$(S, S_i, min_{\text{ECO}})$
9. **if** $((C + \sum_{Y_j \in S_i} \mathcal{C}_{mscs}(Y_j)) < minCost)$ **then**
10. $minCost \leftarrow C + \sum_{Y_j \in S_i} \mathcal{C}_{mscs}(Y_j)$ {// Update lowest clustering cost}
11. $S_{opt} \leftarrow S_i$ {// Update optimal summary}
12. **end if**
13. **end for**
14. **return** S_{opt}

Let us give a more tangible example of the memory usage: Suppose the input sequence contains 12 events, i.e., $|S| = 12$, and we require 60% compresion ratio, i.e., $min_{\text{ECO}} = 8$ operations and thus $k = 12 - 8 = 4$. Suppose each event is described by a set of descriptors of 128 characters encoded in UTF-8, i.e, each event takes 128*4=512 bytes in memory. Generating the set $\mathcal{Z}$ requires $512 \times \frac{4^{12}}{4!} \approx 341$MB; This is a prohibitive amount of memory usage for such a small input sequence.

Once all candidate summaries are generated, each candidate summary needs to be inspected to approximate its canonical clustering cost. The computational cost for processing G-CC is upper bound by (i) the number min_{ECO} of ECOs necessary to generate S', (ii) the number of clusters in a summary, i.e., $|S| - min_{\text{ECO}}$, and (iii) the maximum possible number of events in a cluster, i.e., $min_{\text{ECO}} + 1$. Hence, this computational cost is bound by $O(min_{\text{ECO}}^2 \times n)$. In total, the overall computational footprint of the N-TSS algorithm is $O(min_{\text{ECO}}^2 \times n \times \frac{k^n}{k!})$. N-TSS could give very high quality results but, in practice, the approach is infeasible except for extremely small values of n and k. For these reasons, even though N-TSS is the most basic solution, its prohibitive performances do not allow us to use it as the baseline technique for comparing other solutions. Instead, we will use as baseline the greedy algorithm presented in Section 5.1.

5 Greedy algorithms to build locally optimal solutions

In this section, we present two time sequence summarization algorithms that build locally optimal summaries. These summaries are approximations of the optimal time sequence summary. The first approach presented in Section 5.1 is a straightforward greedy ascending clustering algorithm that performs in a bottom-up fashion. At each iteration, the algo-

rithm performs the ECO that induces the lowest elementary clustering operation cost and works its way up to the summary that achieves the desired compression ratio. The second approach presented in Section 5.2 is an algorithm that randomly selects one (or multiple) cluster(s) Y_i in S and decides if Y_i should be clustered with any one of its neighboring clusters on the timeline that minimizes the clustering cost $\mathcal{C}_{\text{ECO}}$. The process is repeated until the desired compression ratio is achieved.

5.1 G-BUSS: Greedy Bottom-Up time Sequence Summarization

We propose a **G**reedy **B**ottom-**U**p time **S**equence **S**ummarization algorithm called G-BUSS to build a locally optimal time sequence summary. G-BUSS is a hierarchical ascending clustering algorithm that has the particularity of using our objective function defined upon the content and temporal information of events in the sequence. Hence, G-BUSS uses the elementary clustering cost $\mathcal{C}_{\text{ECO}}$ to measure the distance between clusters. The algorithm achieves the desired compression ratio by processing the input sequence in a bottom-up fashion in four steps that are repeated min_{ECO} times:

1. Enumerate all possible clusterings between any 2 candidate clusters Y_i and Y_j in S.
2. Evaluate $\mathcal{C}_{\text{ECO}}(i, j, S)$ for each couple (Y_i, Y_j).
3. Perform the actual ECO on the couple that induces the lowest clustering cost.
4. Update the total clustering cost with $\mathcal{C}_{\text{ECO}}(i, j, S)$.

The pseudo-code of this approach is given in Algorithm 4.

Algorithm 4 G-BUSS algorithm

1. INPUTS:
   $S = \{Y_1, \ldots, Y_n\}$: Input sequence
   ρ_{obj}: Objective compression ratio
2. OUTPUT: Summary
3. LOCALS:
   C: Clustering cost
   min_{ECO}: Minimum number of ECOs to perform
   CM: Cost matrix storing the clustering cost of clusters in S
   X: Set of couples (u, v) for which CM is minimal
4. $C \leftarrow 0$ {// Initialize clustering cost}
5. $min_{\text{ECO}} \leftarrow \lceil |S| \times (1 - \rho_{obj}) \rceil$
6. **for** $cpt = 1$ to min_{ECO} **do**
7. **for** $i = 1$ to $|S|$ **do**
8. **for** $j = i + 1$ to $|S|$ **do**
9. $CM[i, j] \leftarrow \mathcal{C}_{\text{ECO}}(i, j, S)$ {// Compute clustering costs}
10. **end for**
11. **end for**
12. $X \leftarrow \text{argmin}_{(k,l) \in [1 \ldots |S|]^2} (CM[k, l])$ with $k > l$
13. $(i, j) \leftarrow$ first couple in X
14. $C \leftarrow C + CM[i, j]$ {// Update clustering cost}
15. $S \leftarrow f_{\text{ECO}}(i, j, S)$ {// Perform the actual ECO}
16. **end for**
17. **return** S

This greedy algorithm chooses at each iteration the couple of candidate clusters that minimizes the cost $\mathcal{C}_{\text{ECO}}$ (Line 6 to Line 12). When input sequences are very large, it is possible different candidates induce the same minimal clustering cost. As it is defined, G-BUSS arbitrarily chooses the first couple of candidate (Line 13). This observation alone allows us to state that G-BUSS does not necessarily build the optimal time sequence summary but produces a local optimum. Since algorithm N-TSS has prohibitive theoretical performances, we assume that the local optimum solution produced by G-BUSS is a good enough solution. In the rest of this chapter, G-BUSS will be used as the baseline for performance comparisons.

5.1.1 Complexity

G-BUSS's computational footprint is bound by (i) the number of iterations min_{ECO} and (ii) n the number of clusters in S, i.e., the algorithm is bound by $O(min_{\text{ECO}} \times n^2)$. G-BUSS's memory footprint is bound by $O(\frac{1}{2}n^2)$ and corresponds to the storage of the matrix that stores the clustering cost, i.e., float values, of all pairs of clustering candidates.

5.1.2 Pruning technique

G-BUSS's computational complexity does not allow the technique to scale and process very large datasets. However, since the clustering cost $\mathcal{C}_{\text{ECO}}$ has two components, i.e., $\mathcal{C}_{\text{ECO}}$ is based on the (i) content and (ii) temporal information, it is possible to use the temporal component to reduce the number of costs $\mathcal{C}_{\text{ECO}}$ to compute between pairs of clusters.

Intuitively, for a given cluster Y_i in S, Y_i has a limited number of possible clustering candidates. These candidates are located in a neighborhood of m clusters around Y_i on the timeline. Beyond that neighborhood, it becomes more costly to rearrange a candidate cluster Y_j than clustering Y_i with one of its neighbors within the neighborhood of m clusters. In this case, there is no benefit in computing the clustering cost between Y_i with any cluster Y_j such that $j > i + m$ or $0 \leq j < i - m$. One possible way to determine m is to compute the $\mathcal{C}_{mscs}$ between Y_i and its contiguous clusters since these clusters do not induce an event rearrangement cost. This property is formally given in Property 3.4.

Property 3.4 (Event's clustering candidate neighborhood)
Given s a time sequence of events in $\mathcal{S}(\Omega)$, let $\chi_F(s) = (\Pi(s), s^\star)$ be its summary and let S be the canonical second-order time sequence built from the summary $\chi_F(s)$, i.e., $S = \{E_1, \ldots, E_m\}$ with $E_u = (Y_u, t_u^\star)\}$ and $1 \leq u \leq m$. Let Y_i and Y_j be two elements in $\Pi(s)$. The clustering candidates for a cluster Y_i in S that minimize the elementary clustering cost $\mathcal{C}_{\text{ECO}}$ are located in a limited neighborhood. Y_j is a clustering candidate i.f.f.:

$$\text{for } m = \min(\mathcal{C}_{mscs}(Y_{i-1}, Y_i), \mathcal{C}_{mscs}(Y_i, Y_{i+1})),\ i - m < j < i + m \text{ is true}$$

Proof 6 (Proof of Property 3.4)
Suppose there exists a cluster Y_b in S that is outside the neighborhood of m clusters and that is a better clustering candidate for Y_i, and $b < i$ and $\mathcal{C}_\tau(b, i, S) > m$. Therefore the following inequality is true: $\mathcal{C}_{\text{ECO}}(b, i, S) = \mathcal{C}_\tau(b, i, S) + \mathcal{C}_{mscs}(Y_b \cup Y_i) \leq m$ (where $m = \min(\mathcal{C}_{mscs}(Y_{i-1}, Y_i), \mathcal{C}_{mscs}(Y_i, Y_{i+1}))$). Since Y_b is outside the neighborhood of m clusters, $m \leq \mathcal{C}_\tau(b, i, S)$. Putting the pieces together, we should have the following truths: $\mathcal{C}_{mscs}(Y_b \cup Y_i) = 0$ and $m \leq \mathcal{C}_\tau(b, i, S) \leq m$, i.e., $\mathcal{C}_\tau(b, i, S) = m$. By hypothesis, $\mathcal{C}_\tau(b, i, S) > m$. We proved Property 3.4. □

Let us give an explicit example. Figure 3.6 illustrates this notion of neighborhood in which clustering candidates are located. In this example, we assume $\mathcal{C}_{\text{ECO}}(i, i + 1, S) =$

$\mathcal{C}_{mscs}(Y_i \cup Y_{i+1}) = 2$. Therefore, the best clustering candidates for cluster Y_i can only be in a neighborhood of $m = 2$ clusters. Note that since $\mathcal{C}_{\text{ECO}}(i, i+2, S) = \mathcal{C}_\tau(i, i+2, S) + \mathcal{C}_{mscs}(Y_i \cup Y_{i+1}) = 1 + 2 = 3$, cluster Y_{i+2} is not a better clustering candidate than Y_{i+1}. Hence, we can not reduce the size of the neighborhood m. Also, notice cluster Y_{i-3} that is outside the neighborhood of $m = 2$ clusters. Cluster Y_{i-3} requires an event rearrangement cost of $\mathcal{C}_\tau(i-3, i, S) = 2$ to be eligible for clustering with Y_i. Y_{i-3} and Y_i could only be grouped i.f.f. $\mathcal{C}_{mscs}(Y_{i-3} \cup Y_i) = 0$. This grouping is unlikely and counter-productive since it requires to evaluate the cost $\mathcal{C}_{mscs}(Y_{i-3} \cup Y_i)$. This observation highlights our intuition that in some case, it is more costly to rearrange clusters than actually grouping clusters within a restricted neighborhood.

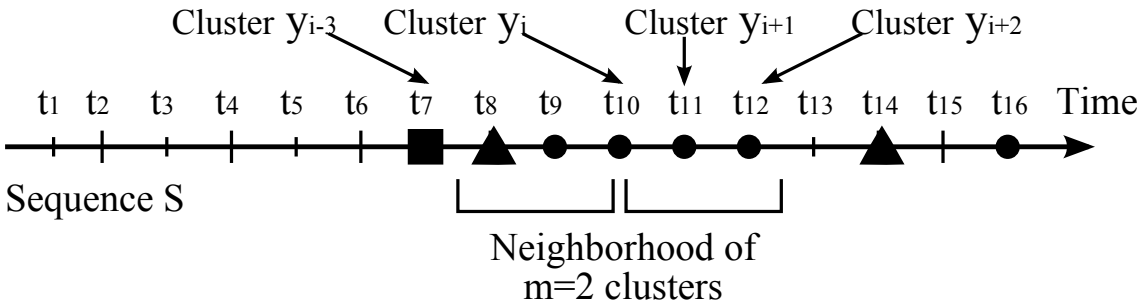


Figure 3.6: Example of neighborhood for searching clustering candidates

Therefore, Algorithm 4 can take into account this pruning technique to reduce the time spent on computing the costs $\mathcal{C}_{\text{ECO}}$ between all pairs of clusters in S. Algorithm 4 can make use of this improvement simply by replacing Line 8 in Algorithm 4 by the following instruction:

"**for** $j = i + 1$ to $\min(\mathcal{C}_{mscs}(Y_{i-1}, Y_i), \mathcal{C}_{mscs}(Y_i, Y_{i+1}))$"

We will show in our experimental study that this simple pruning technique allows to reduce in practice G-BUSS's computational time by approximatively one order of magnitude.

5.2 GRASS: Greedy RAndom-seeded time Sequence Summarization

The second greedy algorithm we present here relies on the observation that G-BUSS does not scale to address very large data sources. When performing tasks such as knowledge discovery, indexation or summarization on very large data sources, it is most desirable if each input data item is considered a limited number of times, ideally once or less. We propose another greedy algorithm called GRASS. The idea in GRASS is to reduce the practical complexity of G-BUSS while maintaining high quality clusters. This is achieved by selecting at each iteration multiple *seed* clusters and exploring each seed cluster's neighborhood for good clustering candidates. This approach heavily relies on the pruning technique introduced in Property 3.4 for G-BUSS.

5.2.1 Algorithm

The intuition behind GRASS relies on two assumptions: (i) events close on the timeline are more likely to relate to a same topic and (ii) the best clustering candidate for each event Y_i is located in a restricted neighborhood on the timeline (Property 3.4).

Suppose Y_j is the best clustering candidate for cluster Y_i within a neighborhood of m events. This statement does not imply that Y_i is the best clustering candidate for Y_j, i.e., there might exist another event Y_k in Y_j's neighborhood such that $\mathcal{C}_{\text{ECO}}(j, k, S) < \mathcal{C}_{\text{ECO}}(i, j, S)$. In this case, since Y_j's neighborhood provides a better clustering candidate,

one should not cluster Y_j with Y_i and should instead consider exploring cluster Y_j's context. This step is reiterated until there is no improvement of the clustering cost or when a predefined number of *hops* is reached. It is possible to limit the number of clusters to explore for searching for a better clustering candidate. We call this parameter the *depth* of the search and we denote it D with $D \geq 1$. When $D = \infty$, the search step is reiterated until the clustering cost does not improve. Note that these steps guarantee the algorithm to converge towards a local optimum since at each step, the clustering cost is reduced by at least 1. When Y_i and Y_j are their mutual best clustering candidates, they are grouped together. Therefore, our GRASS algorithm proceeds as follows:

1. Randomly select a cluster Y_i in S, also called *seed* hereafter.
2. Find Y_i's best clustering candidate Y_j within its neighborhood.
3. Determine if Y_i is Y_j best clustering candidate, i.e., perform step 2 on Y_j.
4. Cluster Y_i and Y_j if Y_i is Y_j's best clustering candidate. Otherwise, repeat from step 2 with Y_j at most D times.

At first glance, GRASS might appear to suffer from several shortcomings:

1. GRASS has quadratic computational complexity. In the worst case scenario, at each iteration, the clustering cost of all pairs of clusters in the input sequence is computed. If this scenario occurs, then GRASS degenerates into the G-BUSS algorithm and has a computational complexity in $O(min_{\text{ECO}} \times n^2)$.
2. Randomly selecting a seed and exploring its neighborhood might not give *good* clusters as the seeds chosen could be outliers or lead to bad local optima.
3. The *quality* of the summary can not be guaranteed.

For the first shortcoming, indeed, GRASS has in theory quadratic computational complexity. However, it is possible to characterize the sequence S that will lead GRASS to degenerate into the G-BUSS algorithm. The only scenario that makes possible for GRASS to achieve quadratic computational complexity occurs when the MSCS cost between all contiguous pairs of clusters on the timeline monotonously decreases by at least 1, i.e., $\forall Y_i \in S, \mathcal{C}_{mscs}(Y_i \cup Y_{i+1}) = \mathcal{C}_{mscs}(Y_{i+1} \cup Y_{i+2}) + k$ with $k \geq 1$. In this situation, the seed selected by the algorithm should always be the first cluster in the sequence, i.e., Y_1.

However, our preliminary experiments on our real world dataset show that this situation is a very unlikely situation and hardly happens. In fact, the GRASS algorithm has in practice much lower computational complexity. We show in our experiments that GRASS's computational time actually improves G-BUSS's computational time by 2 to 3 orders of magnitude. In fact, GRASS performs well on real-life scenarios.

Also, it is possible to alleviate the second and third shortcoming by not limiting GRASS to using one single seed at each iteration. Indeed, when GRASS explores a cluster's neighborhood, sequence S remains unchanged. The ECO is performed and sequence S is altered only when a best local clustering candidate is found. Hence, it is possible to randomly chose P different seeds in the sequence and find the best clustering candidates generated from each seed. Then, the algorithm consolidates the results produced by the P different seeds and the best pair of clustering candidates is chosen for the actual ECO. The overhead induced by this approach is the time necessary for synchronizing all P threads responsible for finding the clustering candidates. In comparison to the base GRASS algorithm that randomly selects only one seed, this approach has the additional benefit of allowing

the algorithm to explore more clustering possibilities in the search space and improving the quality of clusters formed. The pseudo-code of this multi-seed version of GRASS is given in Algorithm 5. This algorithm relies on the procedure named *BestCandidate* that searches for each seed a pair of best clustering candidates. The details of the *BestCandidate* procedure are given in Algorithm 6.

Algorithm 5 GRASS Algorithm

1. INPUTS:
   $S = \{Y_1, \ldots, Y_n\}$: Input sequence
   ρ_{obj}: Objective compression ratio
   D: Depth of local search; P: Number of processing units or threads
2. OUTPUT: Summary
3. LOCALS:
   Y_{seed}: Seed cluster
   (Y_i, Y_j): Best clustering candidates
   min_{ECO}: Minimum number of ECOs to perform
   $PTab$: Table storing best clustering candidates for each seed

4. **while** $min_{\text{ECO}} \neq 0$ **do**
5. **for** $i = 1$ to P **do**
6. Randomly select one seed Y_{seed} from S
7. {// BestCandidates: routine computed in separate thread}
    $PTab[i] \leftarrow$ BestCandidates(Y_{seed},D,S)
8. **end for**
9. Wait for all threads to complete
10. {// Get best clustering candidates from $PTab$}
    $(Y_i, Y_j) \leftarrow \text{argmin}_{k=(Y_l,Y_l')\in PTab}\, \mathcal{C}_{\text{ECO}}(l, l', S)$
11. $S \leftarrow f_{\text{ECO}}(i, j, S)$ {// Perform ECO}
12. $min_{\text{ECO}} \leftarrow min_{\text{ECO}} - 1$ {// Update min_{ECO}}
13. **end while**
14. **return** S

We argue that this approach is realistic, feasible and comes with low overhead. Indeed, it has become a widespread trend for workstations to have hardware with a minimum of two to four physical cores (up to four to eight virtual cores using Intel's HyperThreading technology). In most cutting-edge single server hardware, it is even possible to find configurations having 8*8 (Nehalem-EX[1]) or 1*16 (Niagara 3[2]) physical cores capable of handling 128 to 512 threads, respectively. Also, note that the number of cores usable for scientific computation is multiplied 10- to 100-fold thanks to General-Purpose computing on Graphics Processing Units (GPGPU) technology on newest GPU architectures such as Fermi [Nvi] from Nvidia. Therefore, we believe that it is not a receivable argument to consider this approach as a *simple* parallelization of the algorithm. Here, parallelism is inherent to the process and is a mechanism that uses the available hardware to support and improve the quality of summaries produced.

[1] http://www.intel.com/p/en_US/products/server/processor/xeon7000
[2] http://en.wikipedia.org/wiki/UltraSPARC_T3

Algorithm 6 BestCandidates routine

1. INPUTS:
   Y_{seed}: Seed event; D: Depth of local search
   $S = \{Y_1, \ldots, Y_n\}$: Original sequence
2. OUTPUT: (Y_i, Y_j): Best clustering candidates
3. LOCALS:
   m: Neighborhood of clustering candidates
   Y_j: Best clustering candidate for Y_{seed}
4. {// Compute neighborhood size of Y_{seed}}
   $m_i \leftarrow \min(\mathcal{C}_{mscs}(Y_{seed-1}, Y_{seed}),\ \mathcal{C}_{mscs}(Y_{seed},\ Y_{seed+1}))$
5. {// Get the best clustering candidate for Y_{seed}}
   $Y_j \leftarrow \text{argmin}_{Y_k \in \{Y_{seed-m+1}, \ldots, Y_{seed+m-1}\}} \mathcal{C}_{\text{ECO}}(seed, k, S)$
6. **if** $(D = 1)$ **then**
7. **return** (Y_i, Y_j)
8. **else**
9. **if** $(\mathcal{C}_{\text{ECO}}(seed, j, S) < \mathcal{C}_{\text{ECO}}(j, \text{BestCandidates}(y_j, D-1, S)))$ **then**
10. **return** (Y_i, Y_j)
11. **else**
12. **return** BestCandidates$(y_j, D-1, S)$
13. **end if**
14. **end if**

6 Experiments

In this section we evaluate and validate the time sequence summarization algorithms proposed throughout this chapter. We provide an extensive set of experiments on real-world data from Reuters's financial news archives. We reuse again the dataset extracted from Reuters's 2003 financial news archives as described in Section 6.1.2 in Chapter 2. The N-TSS, G-BUSS and GRASS algorithms were implemented under Microsoft Visual Studio 2008 IDE in C#.

However, since the GRASS algorithm uses hardware specificities, we changed the hardware configuration so that more processing units could be used in our experiments. Experiments on N-TSS, G-BUSS and GRASS were carried out on a Core2Quad at 2.4GHz workstation with 4GB of memory running Windows Seven Pro 64bit. The persistence layer, responsible for storing Reuters's time sequences of financial news and summaries built upon these time sequences, was ensured by PostgreSQL 8.4.

6.1 Metrics

The measures used for evaluating N-TSS, G-BUSS and GRASS are:

1. Computational time
2. Total clustering cost
3. Average MSCS cost for forming a representative event for clusters formed

The computational time is a trivial measure. Since different summarization approaches can achieve the desired compression ratio and could eventually have comparable computational time, it is interesting to evaluate the quality of the summaries produced.

Here, we propose two measures to evaluate the quality of a summary. We evaluate (i) the effort necessary to build the summary and (ii) the effort necessary to represent

each cluster by a representative event. Therefore, the second measure on which N-TSS, G-BUSS and GRASS are evaluated is the total clustering cost induced for building a summary. Indeed, we mentioned earlier that $\mathcal{TC}_{f_{\text{ECO}}}$ could be understood as the edit distance $d(s, \chi_F(s))$ between a sequence s and its summary $\chi_F(s)$, i.e., the effort in terms of elementary operations to build the summary. Naturally, the smaller the total clustering cost, the better.

However, the total clustering cost does not indicate if clusters formed are homogeneous. This is the purpose of our third measure. Here, we chose to evaluate the homogeneity of clusters by evaluating the cost for representing each cluster by a representative event. Therefore, we use the MSCS cost and compute the average MSCS cost for forming a representative event from all clusters in a summary. This average MSCS cost is then denoted avg_{MSCS} and defined as follows:

$$avg_{MSCS} = \frac{1}{|S|} \sum_{Y_i \in S} \mathcal{C}_{mscs}(Y_i) \tag{3.1}$$

Hence, if events in a cluster have *similar* descriptors, the MSCS cost for representing that cluster by a unique representative event should be low, i.e., close to zero. In total, if all clusters produced by a summarization algorithm are homogeneous, then the average MSCS cost should also be low: The lower avg_{MSCS} on the plots, the more homogeneous the clusters. An observation one could make is that avg_{MSCS} is an absolute value. For instance, if the average MSCS cost of a cluster equals 12, there is no indication on how *good* the quality of this cluster is. We can only compare this value to the quality of clusters obtained in other summaries. Ideally, a relative value would be more comprehensive. However, since we can not generate an optimal time sequence summary and compute its exact canonical cost, we do not have a reference cluster quality to normalize this metric. Eventually, we could use the average MSCS cost of clusters produced by G-BUSS as a reference.

6.2 Evaluation of N-TSS, G-BUSS and GRASS

We sum up in Table 3.2 the summarization setup on which N-TSS, G-BUSS and GRASS are evaluated. Since N-TSS has exponential computational complexity and since G-BUSS has quadratic computational complexity, we do not perform summarization on the entire sequence of news events preprocessed. We limit experiments on N-TSS to sequences of 5, 10 and 20 events, and G-BUSS to sequences of 20, 50, 100, 200, 300, 400 and 500 events. The desired compression ratio ρ_{obj} for N-TSS is taken from the set {5%, 10%, 15%} and for G-BUSS, ρ_{obj} is taken in the set {5%, 10%, 15%, 20%, 25%, 50%}. We described in Section 5.2 the worst input sequence for GRASS. In practice, our preliminary experiments show that the worst case scenario is unlikely to occur, at least in our dataset. For this reason, we use a larger dataset and also experiment GRASS on sequences of size up to 10^4. Since we perform all experiments on a quad-core microprocessor, we varied the number of processing units P up to 4. Also, as GRASS randomly selects seed events, we consolidate GRASS performance results by running the algorithm 10 times on each setting and we report the average computational time, clustering cost and average cluster quality.

Summarization computational times are plotted in Figure 3.7, Figure 3.8 and Figure 3.9 for algorithms N-TSS, G-BUSS and GRASS respectively. Figure 3.7 gives N-TSS's computational time with $n = 20$ events in comparison to G-BUSS's and GRASS's computational time for $n = 500$ and $n = 100,000$, respectively. This figure shows that with parameters as small as $n = 20$ and $\rho_{obj} = 15\%$, N-TSS could not complete even after approximatively 23 hours of run time. These preliminary results confirm that N-TSS can

Algorithm	N-TSS	G-BUSS	GRASS
Sequence size n	$1 \leq n \leq 20$	$100 \leq n \leq 500$	$100 \leq n \leq 10^4$
Compression ratio ρ_{obj}	$5\% \leq \rho_{obj} \leq 15\%$	$5\% \leq \rho_{obj} \leq 50\%$	
Processing units P			$P \in \{1, 2, 3, 4\}$
Depth			$D \in \{1, 5, 10, \infty\}$

Table 3.2: Experiments setup

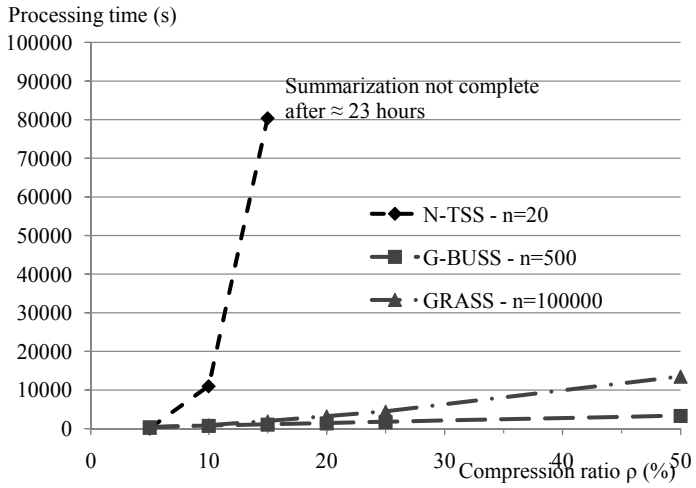


Figure 3.7: N-TSS computational time w.r.t. G-BUSS and GRASS

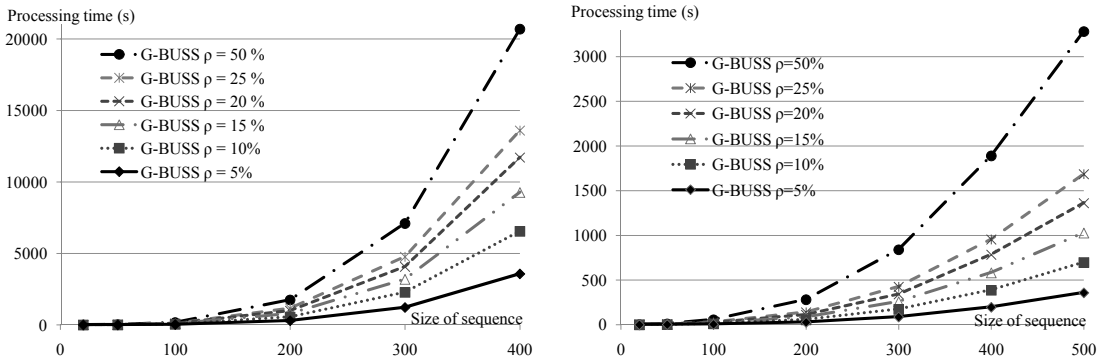
not be reasonably used as a baseline approach. So, we will mainly focus on G-BUSS and GRASS, and use G-BUSS's performances as baseline.

The total clustering cost of G-BUSS and GRASS are given in Figure 3.11. The average MSCS cost for forming a representative event for clusters formed by both techniques is given in Figure 3.12. Since G-BUSS is used as baseline instead of N-TSS, G-BUSS's clustering cost and average MSCS cost also appear in all subfigures of Figure 3.11 and Figure 3.12.

Figure 3.8(a) gives G-BUSS's computational time without using the pruning technique of Property 3.4 and Figure 3.8(b) gives the computational time with the use of the pruning technique. Figure 3.8 confirms our complexity study and shows that G-BUSS has quadratic computational time. On the other hand, Figure 3.8(b) shows that this computational time can be reduced by more than one order of magnitude thanks to Property 3.4. The computational time for each plot is clearly reduced by a factor of approximatively 10. For instance, for $\rho_{obj} = 10\%$ in Figure 3.8(a) the computational time is approximatively 6000 seconds, while in Figure 3.8(b) the computational time is approximatively 600 seconds. Note that these improvements did not alter the total clustering cost or the average MSCS cost of the clusters produced.

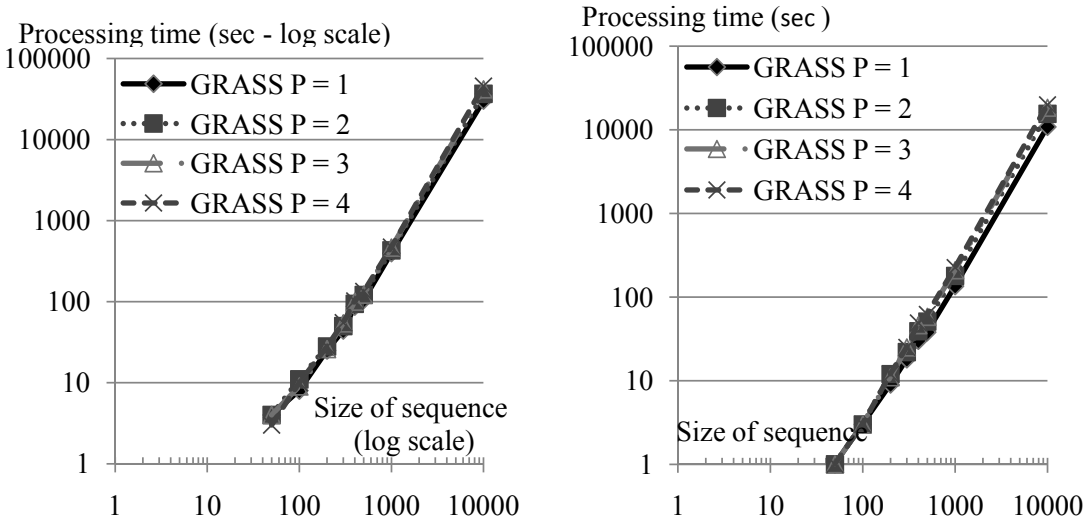
Figure 3.9 gives GRASS's computational times when the desired compression ratio is set to $\rho_{obj} = 50\%$ and the depth setting is varied. More precisely, Figure 3.9(a) corresponds to $D = 1$, Figure 3.9(b) corresponds to $D = 5$, Figure 3.9(c) corresponds to $D = 10$ and Figure 3.9(d) corresponds to $D = \infty$. Sub-figures in Figure 3.9 show that GRASS seems to have linear computational complexity on the logarithmic scale. This observation is mainly influenced by the scale used in the plots. Figure 3.10(b) presents a zoom on the computational time of GRASS for sequences of size smaller than 500 events: This figure shows that GRASS does have a quadratic evolution.

Also, when the number of processing units used increases, the computational time slightly increases. However, this overhead is progressively reduced for each additional

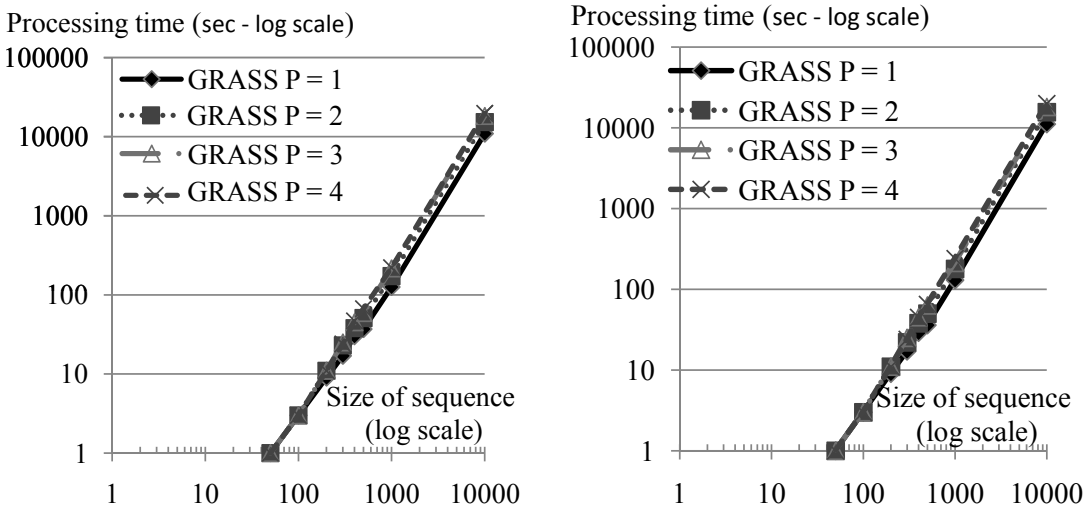


(a) Basic G-BUSS (b) Improved G-BUSS

Figure 3.8: G-BUSS computational time

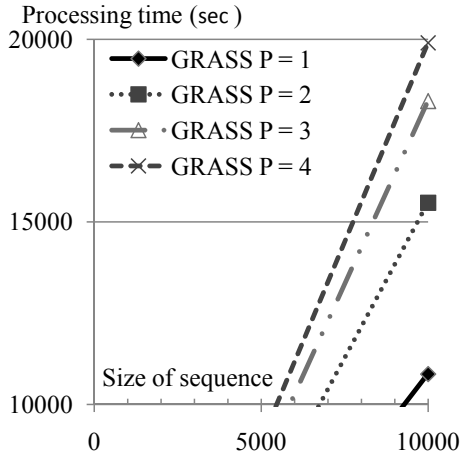


(a) $(D = 1, \rho = 50\%)$ (b) $(D = 5, \rho = 50\%)$

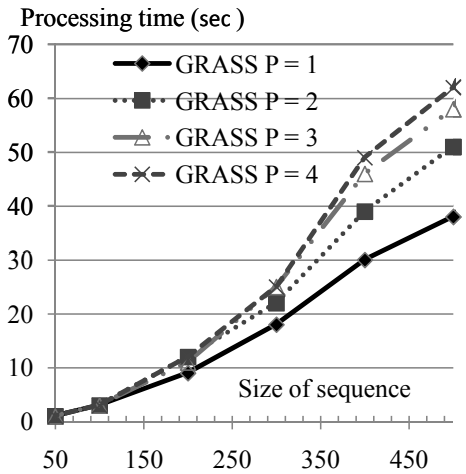


(c) $(D = 10, \rho = 50\%)$ (d) $(D = \infty, \rho = 50\%)$

Figure 3.9: GRASS computational time

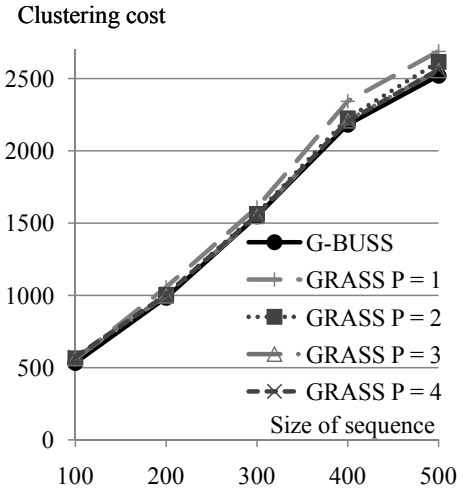


(a) $(D = 5, \rho = 50\%)$ and $n \geq 5,000$

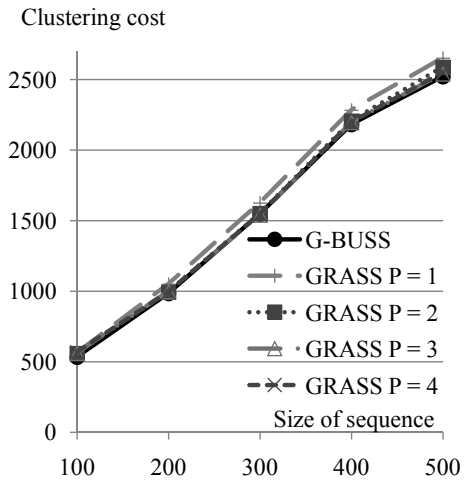


(b) $(D = 5, \rho = 50\%)$ and $n \leq 500$

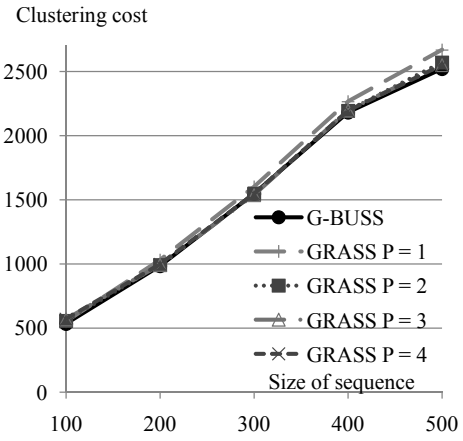
Figure 3.10: GRASS computational time: zooms



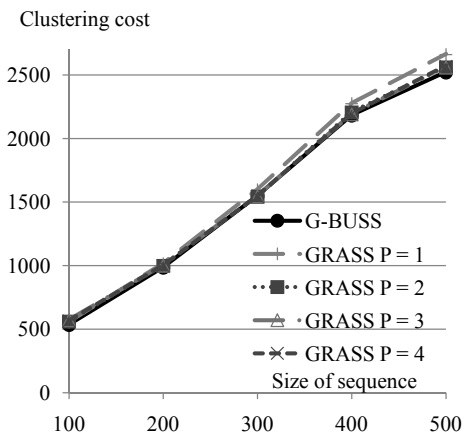
(a) $(D = 1, \rho = 50\%)$



(b) $(D = 5, \rho = 50\%)$

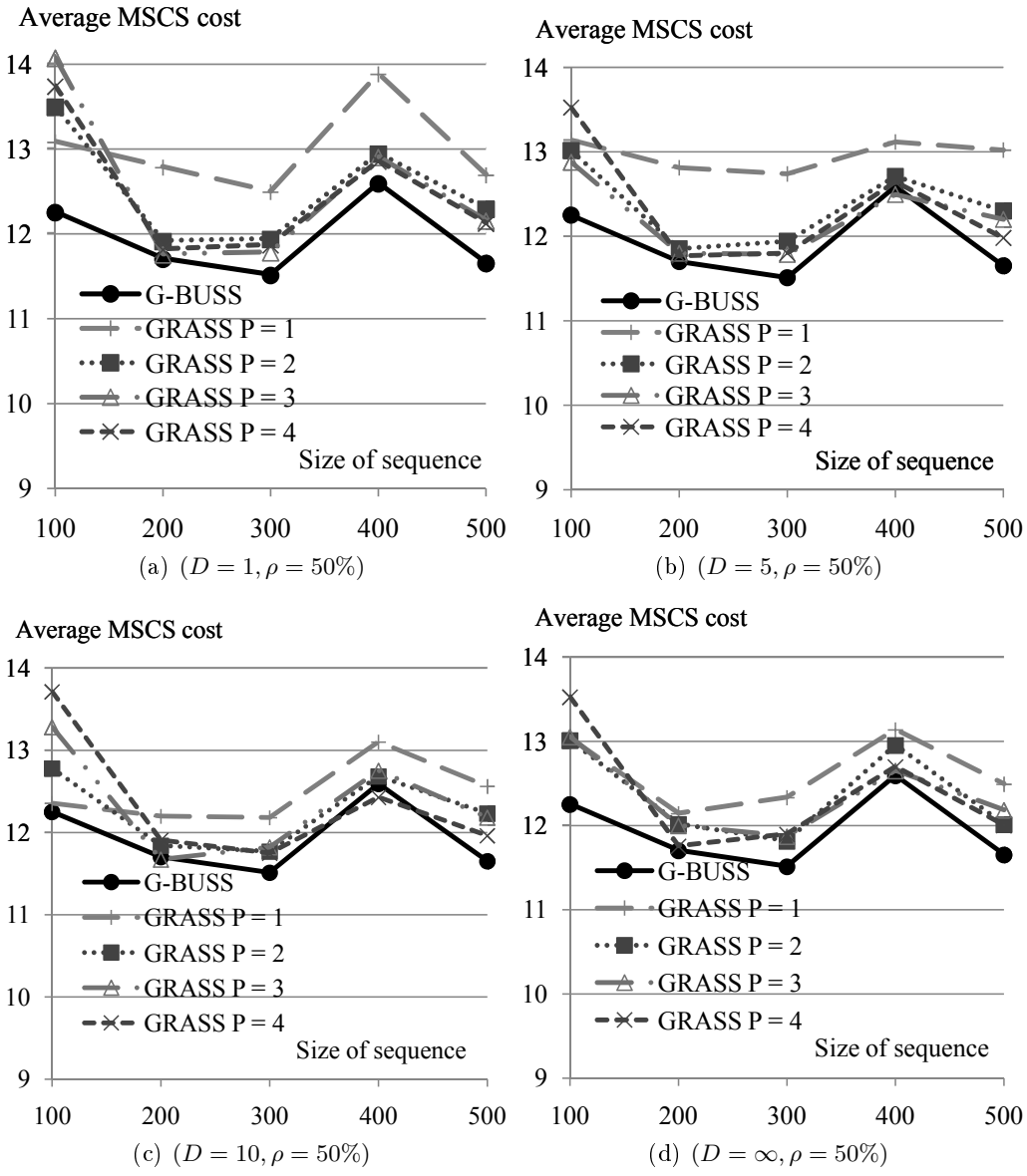


(c) $(D = 10, \rho = 50\%)$



(d) $(D = \infty, \rho = 50\%)$

Figure 3.11: Clustering cost



(a) $(D = 1, \rho = 50\%)$ (b) $(D = 5, \rho = 50\%)$

(c) $(D = 10, \rho = 50\%)$ (d) $(D = \infty, \rho = 50\%)$

Figure 3.12: Average MSCS cost

processing unit added. For instance, Figure 3.10(a) presents a zoom of Figure 3.9(b) at $n = 10^4$. The computational time overhead between $P = 3$ and $P = 4$ is less significant w.r.t. the computational time overhead between $P = 1$ and $P = 2$. This effect can be explained by the fact the algorithm needs to synchronize more threads, i.e., wait for more threads to complete. When more seeds are added (limited to the number of available physical processing units), the overhead converges towards the sum of the computational time of the slowest thread at each iteration.

Nevertheless, the overall computational times presented in these figures confirm our intuition that even though GRASS has quadratic complexity in theory, in practice, GRASS reduces G-BUSS's computational time by several orders of magnitude. Now, we evaluate the quality of the summaries produced by GRASS.

Figure 3.11 presents the clustering cost for each dataset. For the sake of clarity, we do not plot all results for all compression ratios and only present the results obtained for $\rho_{obj} = 50\%$. However, the results obtained for all other compression ratios are consistent with those obtained with $\rho_{obj} = 50\%$. Hence, Figure 3.11(a) give the clustering cost when the depth parameter is set to $D = 1$, Figure 3.11(b) when $D = 5$, Figure 3.11(c) when $D = 10$ and Figure 3.11(d) gives the clustering cost is set to $D = \infty$. First, note that in all sub-figures in Figure 3.11, the total clustering cost for both algorithms increases almost linearly with the size of input sequences. Second, note that the total clustering cost for building summaries with GRASS is almost identical to the total clustering cost induced by building summaries with G-BUSS. In fact, summaries built by GRASS with $P = 1$ processing unit induce the highest total clustering cost. When using at least $P = 2$ processing units, the total clustering cost induced becomes equivalent to the total clustering cost induced by G-BUSS.

Figure 3.12 presents the average MSCS cost for forming representative events from clusters built by G-BUSS and GRASS. Again, for similar reasons, we only plot here the results obtained when the compression ratio is $\rho_{obj} = 50\%$. Note that for almost all datasets G-BUSS produces summaries where clusters are the most homogeneous. In most cases, clusters formed by GRASS with $P = 1$ processing unit are the less homogeneous. This is consequent to the poor seach space coverage when only $P = 1$ processing unit is used. On the other hand, when P increases, the homogeneity of clusters increases and converges towards the homogeneity of clusters formed by algorithm G-BUSS.

Interestingly, these results show that the clustering cost and the homogeneity of clusters in summaries produced by GRASS are only worse than those produced by G-BUSS by an average of 1% to 3%. On some datasets, GRASS actually slightly reduces the clustering cost and improves the homogeneity of clusters w.r.t. G-BUSS, e.g., $n = 400$ in Figure 3.12(c). This observation confirms again the fact that G-BUSS does not generate the optimal time sequence summary but a local optimum. Also, this observation shows that randomly selecting seeds in GRASS can benefit in finding other local optima and globally improve the quality of summaries produced. Therefore, given a desired compression ratio ρ_{obj}, GRASS has achieved the goal of building very high quality summaries while improving G-BUSS's computational time by more than two orders of magnitude.

7 Related work

Data clustering in databases, data warehouses, data streams, etc., is an important data mining task that has been well studied [JMF99,Ber02] when data is represented in numerical form. The essence of data clustering is to form *dissimilar* groups of *similar* objects by

optimizing a predefined distance measure or objective function. Time sequence summarization formulated as a clustering problem is most relevant to the corpus of work in data clustering that addresses categorical data or mixed numerical and categorical data.

However, the clustering task is rendered more difficult when the data considered is defined on categorical domains. Traditional data clustering techniques mostly rely on continuous domains and on the existence of a metric space in these domains to compute the similarity/distance between two objects. When data is defined on categorical domains, the usual distances are no more available, e.g., what is the *average* between descriptors "few" and "plenty"? A corpus of work has emerged and addressed the issues of categorical data clustering without using traditional distance measures. We will discuss these approaches in Section 7.1.

The time sequence summarization algorithms we proposed in this chapter rely on the notion of most specific common subsumer. The cost associated to the *mscs* operator as defined in Definition 3.6, 3.7 and 3.8 is in fact a special instance of *semantic distances*, namely Rada et al's [RMBB89] simple edge counting method. There exists a wealth of research work in Natural Language Processing, from word sense disambiguation to text summarization and speech recognition that make use of the ability to measure the *semantic relatedness* or distance between words of a natural language [Bud99]. We will discuss in Section 7.2 semantic distance metrics from this literature that are most relevant to the algorithms we proposed in this chapter.

When discussing the cost $\mathcal{C}_\tau(i, j, S)$ associated to the event rearrangement operator $\tau_{i,j}$ in Section 3.1.3, we mentioned the fact that $\mathcal{C}_\tau(i, j, S)$ was defined as a function linear with the time dimension. There exist a number of time decay functions widely adopted across a broad spectrum of systems, e.g., data warehouses, data streaming systems or sensor networks. We discuss these alternative approaches in Section 7.3.

7.1 Categorical clustering

Clustering categorical data has attracted many researcher efforts, e.g., [Hua98, GRS99, GGR99, YGY02] and [BLC02, ATMS04, ZP05], and the first methods available in the literature are adaptations of techniques designed for traditional data clustering [Hua98]. Some approaches build summary information [GRS99, GGR99, ATMS04], e.g., in the form of links, connectivity or statistics, while others rely on well defined measures from information theory, e.g., entropy [BLC02]. Also, in [ZP05], the clustering problem is reformulated as a graph problem. A very interesting common feature of these approaches is their capability of clustering categorical data without necessarily relying on explicit distance measures on categorical values themselves. The underlying ideas could be reused to refine or to substitute to the MSCS cost used in our algorithms. However, the most important observation on these techniques is their lack any mechanism for handling the temporal information embedded in the data and for organizing the clusters formed w.r.t. that information. Still, we discuss hereafter these works as they represent the closest related work to our reformulation of time sequence summarization as a clustering problem.

K-MODES [Hua98]

K-MODES is an extension of k-means algorithm for categorical data clustering. K-MODES addresses the issues of traditional k-means methods on categorical data by (i) using a simple matching dissimilarity measure for categorical objects, (ii) replacing the means of clusters by modes and (iii) using a frequency-based method to update modes to

minimize the clustering cost function. The matching dissimilarity measure serves for evaluating the total number of (categorical attribute) mismatches between two objects. Then, equipped with this dissimilarity measure, the approach can replace a clusters' means calculus by its mode calculus. Finally, the k-means approach becomes a K-MODES approach by using a frequency-based method to update the mode of a set of objects.

ROCK [GRS99]

ROCK [GRS99] and CACTUS [GGR99] are representative techniques for clustering transactional databases. In ROCK, Guha et al. propose a novel concept of *links* to measure the similarity/proximity between pairs of objects. The authors overcome the issue of distance by simply not using distances but links to merge clusters. Intuitively, a pair of transactions in a cluster might have few items in common but are linked by a number of transactions in the cluster that have substantial items in common with the two transactions. Hence, the authors define the number of *links* between two objects as the number of *common neighbors* between the two objects. An object's *neighbor* is an object that is considerably similar to it, i.e., $sim(p_i, p_j) \geq \theta$ where θ is a predefined threshold. In fact, the function *sim* could be a well-known distance metric for numerical values, e.g., L_1 or L_2, or non-metric distances, e.g., the Jaccard coefficient [Jac01][3]. The authors redefine the objective function thanks to the concept of links and propose a hierarchical clustering approach.

CACTUS [GGR99]

CACTUS [GGR99] is a combinatorial search-based algorithm where the central idea is that summary information built from the dataset is sufficient for discovering well-defined clusters. In CACTUS, the authors assume that attribute domains are compact, i.e., do not contain more that thousands of values. Clustering is then operated in three phases: (i) summarization, (ii) clustering and (iii) validation. The summarization phase is responsible for computing summary information from the dataset, i.e., counts of inter-attribute and intra-attribute strongly connected pairs. This summary information is then used in the clustering phase to generate a set of candidate clusters. Finally, the validation phase produces the actual set of clusters by verifying the support of candidate clusters against a user-specified threshold.

COOLCAT [BLC02]

COOLCAT [BLC02] is an entropy-based technique for clustering data defined on categorical attributes. The authors define the objective function to optimize thanks to the notion of entropy. Given X a random variable, $S(X)$ the set of values X can take, and $p(x)$ the probability function of X, the entropy $E(X)$ is defined as follows:

$$E(X) = -\sum_{x \in S(X)} p(x) \log(p(x))$$

Hence, the entropy of a multivariate vector $x = \{X_1, \ldots, X_n\}$ is defined as follows:

$$E(x) = -\sum_{x_1 \in S(X_1)} \ldots \sum_{x_n \in S(X_n)} p(x_1, \ldots, x_n) \log(p(x_1, \ldots, x_n))$$

Therefore, for a set D of N points $p_1, \ldots, p_n$, the authors define the objective function, i.e., the total entropy of a clustering $C = \{C_1, \ldots, C_k\}$, as follows:

[3]The Jaccard index of two transactions T_1 and T_2 is defined as: $J(T_1, T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}$

$$E(C) = \sum_{k=1}^{n} \frac{|P(C_k)|}{|D|}(E(P(C_k))) \text{ where } P(C_k) \text{ is the set of points in } C_k \quad (3.2)$$

The idea in COOLCAT is to incrementally build clusters by reducing the entropy of the system (hence, *cooling* down the system). COOLCAT proceeds in two phases: (i) an initialization step followed by (ii) an incremental step. The authors build an initial set of clusters thanks to a bootstrapping algorithm that finds the k most dissimilar data points from a sample set. Then, each remaining data point is incrementally clustered in a suitable cluster thanks to a greedy process that minimizes the total entropy of the system at each iteration. The fact this algorithm is incremental is made possible thanks to Equation 3.2 as it only needs to evaluate the change of entropy when attributing a new data point p_i to cluster C_j. Since this approach highly depends on the order of the data points selected, the authors propose to mitigate this dependency by removing the *worst fitting* points at defined times. These points are then clustered again.

LIMBO [ATMS04]

LIMBO [ATMS04] is a scalable hierarchical categorical clustering algorithm that builds on top of the Information Bottleneck (IB) framework [TPB99] for quantifying the relevant information preserved when clustering. The IB framework is a technique for finding the best trade off between accuracy and complexity when clustering a random variable X, given a joint probability distribution between X and an observed relevant variable Y. Hence, the central idea in LIMBO is that tuples in the dataset and clusters need not be kept in-memory, but just sufficient statistics to describe them are necessary. LIMBO operates in three phases: (i) build statistics summaries, (ii) build clusters from the statistics gathered and (iii) associate tuples to the clusters built.

First, the authors summarize a cluster of tuples in a *Distributional Cluster Feature* (DCF). The DCF of a cluster c, denoted $DCF(c)$ is the couple $(p(c), p(A|c))$ where $p(c)$ is the probability of cluster c and $p(A|c)$ the conditional probability distribution of attribute values given cluster c. Hence, the authors build a DCF tree by considering tuples one by one to grow the DCF tree. Then, LIMBO employs an *Agglomerative Information Bottleneck* algorithm to cluster the DCFs at the leaves level to produce a clustering of k DCFs. These k DCFs serve as representatives of the clusters. LIMBO's final phase associates the tuples from the dataset to the corresponding clusters. Interestingly, the summary/clustering produced by LIMBO partially complies to Definition 1.7 of a time sequence summary. Even though LIMBO does not consider the temporal aspect of data, in its second phase the algorithm produces representatives for the cluster, i.e., the intent of the summary, and in the third phase produces the extent of the summary.

CLICKS [ZP05]

CLICKS [ZP05] is a clustering algorithm based on graph/hyper graph partitioning. CLICKS models the dataset as a graph where the vertices (attribute values) form k disjoint set. Vertices are connected by an edge if the two corresponding attribute values occur in a same instance. The authors match the categorical clustering problem to the problem of enumerating maximal k-partite cliques in the k-partite graph. Hence, the clustering problem is solved by enumerating all maximal k-partite cliques in the graph, then forming the final clusters by verifying the support of the candidate cliques within the original dataset.

7.2 Semantic distances

Semantic distance is a metaphor for measuring the degree of relatedness between concepts. Supposing concepts are the units of knowledge, then semantic distance is a tool for organizing that knowledge. Semantic distances are useful in applications as various as sense disambiguation, text summarization, text annotation, topic tracking, information extraction, information retrieval, indexing or automatic word correction. The problem of formalizing and quantifying the intuitive notion of similarity takes its roots in philosophy, psychology, cognitive science, artificial intelligence, etc.. For instance, we have already discussed in Chapter 2 how humans organize their knowledge in their memory.

Many researchers actually distinguish two kinds of semantic distances: *semantic similarity* and *semantic relatedness*. Semantic similarity evaluates the degree to which two concepts resemble each other, while semantic relatedness encompasses a wider variety of relationships. The difference between semantic similarity and semantic relatedness can be very subtle and is usually better defined through examples. A commonly used example is given by Resnik in [Res95]: For instance, the concepts *car* and *gasoline* appear more related than *car* and *bicycle* since *car* uses *gasoline*; However, *car* and *bicycle* are certainly more similar since they share common features (wheels, brakes, etc.).

Since our work makes use of Background Knowledge acquired in the form of taxonomies, i.e., IS-A relations only, we focus this discussion on semantic similarity techniques. Therefore, we present in this section most prevailing measures proposed in the literature. A more detailed study on this literature can be found in Budanitsky's survey [Bud99] and evaluation work [BH01]. These measures can be separated into two categories: (i) taxonomy-based relatedness and similarity measures and (ii) corpus-based measures of distributional similarity.

7.2.1 Taxonomy-based measures

Rada et al.'s Simple Edge Counting distance [RMBB89]

Rada et al. [RMBB89] propose to evaluate the similarity between two concepts in terms of the path that link the two concepts in a taxonomy. The degree of similarity is determined on the basis of this path and generally corresponds inversely to the length of the path. Approaches based on this path are also known as *edge-counting* approaches.

The authors offer a simple representation of the distance between two concepts c_1 and c_2 in a taxonomy H. They define this distance as the number of edges in H between c_1 and c_2 in H. This measure is very intuitive but its quality highly depends on the quality of the taxonomy employed. The MSCS cost employed in the GRASS algorithm is a direct application of Rada et al.'s Simple Edge Counting distance.

Wu and Palmer [WP94]

Wu and Palmer investigate in [WP94] the semantic representation of verbs in computer systems and its impact on lexical selection problems in Machine Translation (MT). In machine translation research, the inherent difficulty of verb representation lies in the fact verb meanings are involved in several conceptual domains. For instance, the verb *hit* identifies a complex event that involves the domains *force*, *motion* and *contact*. For the purpose of identifying the sense/meaning of a verb V, the authors project V into simpler concepts in which V is involved. Then, they propose an edge-counting based similarity and distance measure, denoted sim_{WP} and $dist_{WP}$, respectively, for concepts in these domains. Given two concepts c_1 and c_2 within one single conceptual domain A organized

into taxonomy H_A, Wu and Palmer's similarity metric sim_{WP} and distance $dist_{WP}$ are defined as follows:

$$sim_{WP}(c_1, c_2) = \frac{2 \times N_3}{N_1 + N_2 + 2 \times N_3} \tag{3.3}$$

$$dist_{WP}(c_1, c_2) = 1 - sim_{WP}(c_1, c_2) = \frac{N_1 + N_2}{N_1 + N_2 + 2 \times N_3} \tag{3.4}$$

where N_3 is the distance (in terms of number of edges) from the root of the hierarchy to the most specific common subsumer of c_1 and c_2, denoted $mscs(c_1, c_2)$ (in our terminology), N_1 the distance between c_1 to $mscs(c_1, c_2)$ and N_2 the distance between c_2 and $mscs(c_1, c_2)$. We can illustrate this semantic measure with our *topic* domain taxonomy given in Figure 3.3. We compute the similarity measure between concepts "Sequential patterns" and "Datamining", and between concepts "Query rewriting" and "Join query". The following results show the clear similarity between the two first concepts:

sim_{WP}("Sequential patterns","Datamining") $= \frac{2 \times N_3}{N_1 + N_2 + 2 \times N_3}$
sim_{WP}("Sequential patterns","Datamining") $= \frac{2 \times 1}{2 + 0 + 2 \times 1}$
sim_{WP}("Sequential patterns","Datamining") $= 0.5$
sim_{WP}("Query rewriting","Join query") $= \frac{2 \times 1}{1 + 2 + 2 \times 1}$
sim_{WP}("Query rewriting","Join query") $= 0.4$

Since verbs could be involved in multiple conceptual domains, the authors extend their similarity measure on one single domain to multiple domains. Given two verbs V_1 and V_2, this similarity measure is defined as a summation of weighted similarities between pairs of simpler concepts in each of the domains the two verbs V_1 and V_2 are projected onto:

$$WordSim(V_1, V_2) = \sum_i W_i \times sim_{WP}(c_{i,1}, c_{i,2}) \tag{3.5}$$

This similarity measure is most interesting since it considers both the *commonality* and the differences between concepts c_1 and c_2; It actually refines Rada et al.'s simple edge counting measure. Also, one should note that verbs are not the only terms that can be involved in multiple conceptual domains. We already mentioned in Section 6.1.2 in Chapter 2 the problem of homonymy and polysemy when preprocessing Reuters's financial news archives into time sequences of financial news events. The generalized similarity measure given in Equation 3.5 presents an alternative for addressing homonymy and polysemy issues.

Leacock and Chodorow [LC98]

Leacock and Chodorow address in [LC98] the problem of polysemous word sense identification using *local context* and WordNet similarity. *Local context* consists of semantic and syntactic cues in the immediate vicinity of a polysemous word. The authors augment local context classifiers with semantic information obtained by finding concepts *similar* to concepts already known by the classifiers. The authors's approach exclusively relies on WordNet to compute the similarity between two concepts c_1 and c_2. First, the length of the shortest path between the different senses of the two concepts in WordNet is determined. Then the length of the path found is scaled as follows:

$$sim_{LC}(c_1, c_2) = \max(-\log(\frac{N_p}{2 \times D})) \tag{3.6}$$

where N_p is the number of nodes in the path p between concepts c_1 and c_2 and D is the maximum depth of the taxonomy. We can illustrate this semantic measure with our

topic domain taxonomy given in Figure 3.3. We compute the similarity measure between concepts "Sequential patterns" and "Datamining", and between concepts "Sequential patterns" and "Join query". The following results show the clear similarity between the two first concepts:

$$sim_{LC}(\text{"Sequential patterns"}, \text{"Datamining"}) = -\log(\frac{path(\text{Sequential patterns, Datamining})}{2*3})$$
$$sim_{LC}(\text{"Sequential patterns"}, \text{"Datamining"}) = -\log(\frac{1}{6})$$
$$sim_{LC}(\text{"Sequential patterns"}, \text{"Datamining"}) = 0.77$$
$$sim_{LC}(\text{"Sequential patterns"}, \text{"Join query"}) = -\log(\frac{path(\text{Sequential patterns, Join query})}{2*3})$$
$$sim_{LC}(\text{"Sequential patterns"}, \text{"Join query"}) = -\log(\frac{5}{6})$$
$$sim_{LC}(\text{"Sequential patterns"}, \text{"Join query"}) = 0.07$$

7.2.2 Corpus-based measures

Philip Resnik [Res95]

Philip Resnik introduced in [Res95] the first similarity measure to combine an edge-counting methodology with corpus statistics. The author's intuition is that *"one criterion of similarity between two concepts is the extent to which they share information in common"*. This information in common is actually the Most Specific Common Subsumer of the two concepts. Hence, Resnik determines the similarity between two concepts as the *information content* of the shared subsumers. The higher the information content, the more the concepts share in common and therefore they are more similar.

First, Resnik defines the probability of encountering an instance of a concept c denoted $P(c)$. For this purpose, the author relies on frequency information from text corpus. He calculates the number of occurrences of concept c and all occurrences of all concepts subsumed by c. The total number frequency of concept c is denoted $freq(c)$ and defined as follows:

$$freq(c) = \sum_{i \in Word(c)} count(i) \tag{3.7}$$

where $Word(c)$ is the set of words that corresponds to all concepts subsumed by c.

Then, given a text corpus of size N, the probability of encountering an instance of concept c is defined as follows:

$$P(c) = \frac{freq(c)}{N} \tag{3.8}$$

Putting the pieces together, Resnik's similarity measure between two concepts c_1 and c_2 is given by:

$$sim_R(c_1, c_2) = max_{c \in S(c_1, c_2)}[-\log(P(c))] \tag{3.9}$$

where $S(c_1, c_2)$ is the set of concepts that subsume both c_1 and c_2.

Jiang and Conrath [JC97]

Similar to Resnik's idea, Jiang and Conrath propose in [JC97] a hybrid approach that relies on information from taxonomies and from text corpus. The authors compensate the unreliability of edge counting by weighting each edge with a probability based on corpus statistics. In contrast with Resnik's approach who uses the information content of the MSCS node, the authors use information theory to weight each edge in the path that links two concepts c_1 and c_2.

The authors show that the semantic distance between a parent concept and its child concept is the difference of their information content. The underlying rationale is that the difference in information content reflects the information required to distinguish a concept from its siblings. Therefore, Jiang and Conrath measure the semantic distance between two concepts by summing the individual semantic distance between nodes in the path that links concepts c_1 and c_2. This distance is defined as follows:

$$\begin{aligned} dist_{JC}(c_1, c_2) &= IC(c-1) + IC(c_2) - 2IC(mscs(c_1, c_2)) \\ dist_{JC}(c_1, c_2) &= 2 \times \log(P(mscs(c_1, c_2))) - (\log(P(c_1)) + \log(P(c_2))) \end{aligned} \tag{3.10}$$

Lin [Lin98]

Lin [Lin98] proposes an information-theoretic approach to define a semantic similarity measure. The author actually proves that *"the similarity between A and B is measured by the ratio between the amount of information needed to state their commonality and the information needed to describe what they are"*. This theorem translates as follows:

$$sim(A, B) = \frac{\log(P(\text{common}(A, B)))}{\log(P(\text{description}(A, B)))} \tag{3.11}$$

This theorem is applicable to numerous domains and in particular for computing the semantic similarity in a taxonomy. The author defines the semantic similarity between two concepts c_1 and c_2 as:

$$sim_L(c_1, c_2) = \frac{2 \times \log(P(mscs(c_1, c_2)))}{\log(P(c_1)) + \log(P(c_2))} \tag{3.12}$$

where the notations $P(c)$ and $mscs(c_1, c_2)$ are similar to Resnik's notations. One should notice that Wu and Palmer's similarity measure [WP94], denoted sim_{WP}, can be understood as a special case of sim_L. In fact, if $P(C|C')$ is the same for all pairs of concepts such that there is an IS-A link from C to C' in the taxonomy, sim_{WP} coincides with sim_L.

LIMBO [ATMS04]

Andritsos et al. propose in [ATMS04] a novel application of LIMBO to quantifying the distance between attribute values of a same attribute. The authors' intuition is that the similarity between two attribute values from a same attribute domain could be evaluated thanks to their *context*. For instance, in a movie database, given relation *Movie(Director, Actor, Genre)*, it is not apparent how to evaluate the similarity of attribute values "Scorses" and "Coppola" from the *Director* attribute domain since each movie only has one director. Intuitively, these values are similar if the *context* in which they appear is similar. The authors define the context as the distribution these values induce on the remaining attributes.

Formally, suppose *A'* is the attribute of interest, e.g., *Director*, and **A'** the domain of *A'*, i.e., the set of all values that can be taken by *A'*. The authors denote by $\tilde{\mathbf{A}} = \mathbf{A} \setminus \mathbf{A'}$ the set of attribute values in the remaining attributes, e.g., if *A'*="Director" then $\tilde{\mathbf{A}}$ contains all attribute values of "Actor" and "Genre". The random variables that range over **A'** and $\tilde{\mathbf{A}}$ are denoted A' and $\tilde{A}$, respectively. Then, $p(\tilde{A}|v)$ denotes the distribution that value $v \in \mathbf{A'}$ induces on the values in $\tilde{\mathbf{A}}$. For some $a \in \tilde{\mathbf{A}}$, $p(a|v)$ is the fraction of the tuples in T that contain v and a. Also, for some $v \in \mathbf{A'}$, $p(v)$ is the fraction of tuples in T that contain the value v. In total, given two attribute values v_1 and v_2 in *A'*, the authors define the semantic distance between v_1 and v_2 as the information loss $\delta I(v_1, v_2)$

incurred about variable $\tilde{A}$ if v_1 and v_2 are merged. This information loss is equivalent to the increase of uncertainty of predicting values in $\tilde{A}$ when v_1 and v_2 are replaced by $v_1 \vee v_2$.

The LIMBO algorithm can be applied to the joint distribution of random variables A' and $\tilde{A}$ for clustering the values of attribute A'. Since v_1 and v_2 never appear together in the database (e.g., these are the directors "Scorses" and "Coppola"), we can produce a new value $v_1 \vee v_2$:

$$p(v_1 \vee v_2) = p(v_1) + p(v_2) \text{ and}$$
$$p(a|v_1 \vee v_2) = \frac{p(v_1)}{p(v_1 \vee v_2)} p(a|v_1) + \frac{p(v_2)}{p(v_1 \vee v_2)} p(a|v_2) \tag{3.13}$$

This approach seems interesting since the authors rely only on the data content and do not require additional domain taxonomies to be available. However, it is difficult to assess the impact of the approach proposed since the authors do not report any experiments on their LIMBO-based measure w.r.t. to other taxonomy-based or corpus-based (or hybrid) measures.

DILKA [IM09]

DILKA [IM09] is another context-based technique for Distance Learning of Categorical Attributes. The authors also back the idea that the semantic similarity between two attribute values v_1 and v_2 can be evaluated through the frequency of attribute values v_j that are employed in association with v_1 and v_2. Formally, let S_1 be the set of examples, i.e., data points, having value v_1 for attribute A_i and let S_2 be the set of examples having value v_2 for the same attribute A_i. The authors rely on the frequency with which values in a certain set of attributes A_j, the *context* of A_i, occur in S_1 and S_2. Hence, the distance between two values v_1 and v_2 of attribute A_i is evaluated as the difference of the frequency with which values in A_j occur in S_1 and S_2.

The key operations in DILKA are:

- Select a relevant subset of attributes A_j as the context for a given attribute A_i.
- Compute the distance between values in A_i using its context A_j

For the purpose of selecting a relevant subset of attributes A_j as the context for a given attribute A_i, the authors leverage ideas from the well known data mining problem of feature selection. In particular, the authors use the Symmetric Uncertainty [YL03] (SU) correlation measure defined to quantify the mutual dependence of two variables X and Y. This measure consists in some mutual information that measures how much knowledge on one of the two variables reduces uncertainty on the other variable. This mutual information is then normalized by the entropy of each variable, i.e., $H(X)$ and $H(Y)$, respectively:

$$SU(X,Y) = 2 \times \frac{IG(X|Y)}{H(X) + H(Y)} \tag{3.14}$$

where IG represents the Information Gain. The authors extend this correlation measure so that the context of an attribute A_i can be expressed as a set of attributes A_j, i.e., a set of variables.

Equipped with the context for a given attribute A_i, the authors then compute the distance between every pair of values in A_i. The conditional probability of the value v of attribute A_i given the value y_k of the context attribute $A_{j,k}$ is denoted $P(v|y_k)$. Thus, the distance between two values v_1 and v_2 is computed by the formula:

$$dist_{IM}(v_1, v_2) = \sqrt{\sum_{Y \in context(A_i)} \sum_{y_k \in Y} (P(v_1|y_k) - P(v_2|y_k))^2} \tag{3.15}$$

Similarly to LIMBO, DILKA is of utmost interest since it relies only on the data content and does not require additional domain taxonomies. The authors show through their experiments that DILKA allows to produce high level quality clusters when coupled to existing hierarchical clustering algorithms. For instance, the Ward hierarchical clustering algorithm [War63], denoted HCL, is used in their experiments. The authors show that the *accuracy*, expressed as a classification error w.r.t. to a reference classification, achieved by DILKA associated to HCL outperforms the accuracy achieved by ROCK or LIMBO associated to HCL. Also, the computational time when operating HCL with DILKA outperforms the computation time when operating HCL with LIMBO by almost 2 orders of magnitude.

From time sequence summarization view point, DILKA is an interesting approach that could be leveraged to compute the similarity between events in a time sequence of events. Taxonomies defined for the descriptive domains from which event descriptors are taken could be used solely for concept formation. However, using DILKA alone is not sufficient and some form of integration with event rearrangement should be considered for handling the temporal dimension of events.

7.3 Time decay models

Research on time decay models has been widely influenced and boosted by research on data streams processing. Indeed, Data Streams Management System need to handle streams of data that arrive at very high rates and that require immediate processing. Much research work has focused on answering queries on these sources of data under some temporal conditions. For instance, sliding windows [CC02, CF02, MWA+03, CJS03, MFHH03, HAF+03, Gol02] have been widely used to address queries. Sliding windows have been very popular due to the fact they capture with very high precision the most recent data in the stream and since in some applications, e.g., networking or finance, it is assumed that the information of interest usually lies in the most recent data.

In [CSSX09], Cormode et al. give a formal definition of a time decay function for a data stream, but this definition broadly applies to any form of sequence data. The authors consider a stream of input items (t_i, v_i) which describes item arrivals. The i^{th} item is associated to a timestamp t_i and a value v_i; In the case of time sequences of events, v_i would correspond to the set of descriptors associated to the i^{th} event. Hence, the authors define a time decay function $w(i,t)$ as a function that takes some information about the i^{th} item and returns a weight for this item that satisfy the following properties:

- $w(i,t) = 1$ when $t_i = t$ and $0 \leq w(i,t) \leq 1$ for all $t \geq t_i$
- w is monotone non-increasing as time increases: $t' \geq t \Rightarrow w(i,t') \leq w(i,t)$

The most well known and commonly used time decay function is the exponential time decay function. This decay function is defined as follows: $w(t) = a_0 \times exp^{-\lambda \times t}$ with a_0 an initial quantity or weight at time $t = 0$ and $\lambda > 0$ the decay constant. This decay function is borrowed from natural science phenomenons. It has been used to describe natural science phenomenons such as the radioactivity decay, atmospheric pressure, electromagnetic radiation etc.. However, in some cases, an exponential function might render the data obsolete too fast. In such situations slower decay functions might be more adapted [CS06]. Polynomial time decay functions are more commonly used in Physics applications such as acoustic energy propagation. A polynomial time decay function's most general form is $w(t) = \sum_{i \in \theta_i} a_i \times t^i$ where θ_i is a set of values. However the most commonly used polynomial functions are monomials, i.e., $w(t) = t^\theta$ with $\theta > 0$.

Cormode et al. distinguish in [CSSX09] two classes of decay functions, namely *backward decay* functions and *forward decay* functions. In the class of backward decay functions,

the weight of an item in the sequence is written as a function of its *age*, i.e., the weight of the item is measured *back* from the current time and it is constantly changing since time is constantly evolving. On the other hand, in the class of forward decay functions, the decay or weight of an items is computed as an amount of the time between the arrival of the item and a fixed point L. L is called a landmark. Therefore, the decay of the item is obtained by looking forward from the landmark L. The authors show that this model captures well known time decay models, such as exponential decay, polynomial decay or landmark windows (equivalent to backward sliding windows model). The authors provide empirical evidence that their novel model is effective and can be executed in streaming environments without the need to extend the underlying system.

An original piece of work is Harrison et al.'s study on progress bars [HAB07]. The authors explore the impact of progress bar behaviors on user perception of process duration. We believe this work is related to ours since it highlights what are tolerable or acceptable evolutions of time from a user point of view. Most time decay functions already presented aim at modeling with fidelity natural phenomenons. However, from summarization view point, we believe the user's perception is as important. In other words, information should be gathered and presented to the user at a temporal granularity that suits his preference and not necessarily only at the granularity that fits the data.

The authors observe the fact that progress bar behavior is often modeled as a linear function to the load of work achieved while human perception of the passage of time is not linear [Hog78, All79, Blo90]. The authors defined nine different progress functions, e.g., linear, slow wavy, fast wavy, power or inverse power, and listed 22 individuals' preferences scores. Interestingly, this study shows that users are ready to tolerate negative behaviors, e.g., stalls and inconsistent progress, at the beginning of a process while they have strong aversion to negative behaviors at the end of a process. When designing time decay functions, we believe these results should be considered to properly refine the decay function such that the function models the natural phenomenon while taking into account the user's perception of time.

8 Chapter summary

Massive collections of time sequences of events appear in a number of domains such as medicine, the WWW, business or finance. A concise representation of these collections is desirable to support chronology-dependent applications. Time sequence summarization is a summarization concept introduced to transform time sequences of events into a more concise but informative form, using the data's content and temporal information. However, the *TSaR* approach proposed in Chapter 2 has numerous non-trivial parameters to tune.

In this chapter, we have presented time sequence summarization under the angle of a novel conceptual clustering task. The problem of building a time sequence summary from a time sequence of events has been introduced as the problem of building an optimal clustering of events in the sequence. The novelty in defining time sequence summarization as a conceptual clustering problem lies in the function used to evaluate how similar two events or clusters on the timeline are. The specificity of this function is its dual consideration of the content of events and their proximity of the timeline. We introduced a simple but effective cost function that consists of two components: (i) a component that evaluates the cost for generalizing events in a cluster into one single and common representation, also called the Most Specific Common Subsumer (MSCS), and (ii) a component that evaluates the cost for virtually rearranging clusters on the timeline so they are considered *close* and eligible for clustering. We rely on this cost function and formalize the novel problem of

building optimal time sequence summaries. Hence, time sequence summarization becomes a parameter-free approach.

We propose three algorithms to solve this problem: (i) a basic exhaustive approach called N-TSS, (ii) a greedy hierarchical ascending algorithm called G-BUSS that generates a local optimal summary and (iii) a greedy parallel random-seeded algorithm called GRASS that computes summaries of quality equivalent to G-BUSS's. Our preliminary experiments on N-TSS show that N-TSS has prohibitive performances and hence, could not be used as a baseline for evaluation.

G-BUSS is a hierarchical ascending clustering algorithm that has the particularity of using our cost function. Events are thus iteratively clustered thanks to their content and their time of occurrence. We showed that G-BUSS has quadratic computational complexity and proposed a pruning technique that exploits the temporal component of our cost function to reduce in practice G-BUSS's computational time. Our experiments show that this improvement allows to reduce G-BUSS's computational time by approximatively one order of magnitude. This approach is therefore chosen as baseline for evaluation.

GRASS is a random-seeded parallel algorithm that builds on top of the pruning technique introduced for G-BUSS and further reduces in practice G-BUSS's computational time. At each iteration GRASS explores the search space at multiple locations on the timeline to improve the quality of clusters produced. Our extensive set of experiments on Reuters's 2003 financial news archives showed that GRASS improves G-BUSS's computational time by two to three orders of magnitude while producing clusters of quality only worse than G-BUSS's by 1% to 3%.

In Section 7.3, we have discussed works related to time decay functions. An interesting direction for future work is to study how to decay the data w.r.t. to the time decay. Indeed, the purpose of a time decay function is to compute a weight for each data point on the timeline. However, the weight attributed to a data point does not state how the content of the data should be degraded or abstracted. Hence, an interesting perspective is to explore how the content of a data point should be abstracted w.r.t. to the weight it is attributed.

Also, so far, the time sequence summarization problem has been defined for summarizing historical archives in an offline mode. However, some applications in networking, e.g., intrusion detection or finance can require real-time analysis of the data. Addressing time sequence summarization in an online mode is an even more challenging task. As presented earlier, addressing streaming data comes with specific limitations: Since data streams are assumed to be infinite and generated at high rates, summaries should be built in constrained memory and in one single pass over the data. Therefore, an interesting research orientation is to study how (locally) optimal time sequence summaries can be built in an incremental way and how the theoretical computational complexity of the algorithm could be reduced. One should note that addressing this challenge also requires to handle the way the content of each data point on the timeline should be degraded or abstracted.

Chapter 4

Mining higher order patterns: Application to Reuters's archives

1 Introduction

Domains such as medicine, the WWW, business or finance generate and store on a daily basis massive amounts of data. This data represents large collections of time sequences of events where events are associated to some *primitive* information and a time of occurrence. For instance, Reuters has a team of several thousand of journalists that produce series of news articles that report events occurring around the world. The information produced and archived is said to be *primitive* or *raw* since it has not been processed for any analysis purpose. This primary data has rich contents and often contains a mix of unstructured information, e.g., free text, and structured information, e.g., descriptors defined on categorical and numerical domains. These archives represent valuable sources of insight for analysts to browse, analyze and discover golden nuggets of knowledge. For instance, biologists could discover disease risk factors by analyzing patient history [WRZ05], traders investigate financial data for understanding global trends or anticipating market moves [ZZ04], web content producers and marketing people are interested in profiling client behaviors [SCDT00].

Recently, Roddick et al. [RSLC08] formally introduced the *Higher Order Mining* (HOM) paradigm. The authors observed that in many environments, e.g., streaming environments, data analysis computation speed hits limits set by hardware and firmware technologies. Also, in some cases, the primary data is not available or available for analysis for a limited time period. Ultimately, data mining methods need to operate upon a derived form of the data. Such derived forms of the data include for instance: Results from previous mining activities, results of aggregates or summaries. Therefore, the HOM paradigm encompasses all data mining methods that operate on such forms of the data.

In this context, *Sequential Pattern Mining* (SPM) is a data mining paradigm introduced by Agrawal and Srikant in [AS95] that is of utmost interest. Indeed, the development of code bar technologies in the 90's and the increasing mass of transaction information generated by these technologies has motivated Agrawal and Srikant to propose a new data mining paradigm to identify recurrent customer patterns. Hence, the original purpose of sequential pattern mining is to analyze customer sales transactions to identify frequent purchase behaviors: The principle of SPM is to identify, in customer sales transactions sets (or subsets) of *items*, also called *itemsets*, that are often purchased together. This data mining paradigm has since been known as the *market basket analysis* paradigm. SPM has attracted

much attention during the past 15 years (these surveys [ZB03, TPBM05, HPY05] give an interesting insight on the domain). The increasing volume of data to address and the increasing size of knowledge sets uncovered has motivated researchers to define more compact patterns, also known as *closed sequential patterns* [YHA03, TYH03, WH04, SBI06, BS07].

Agrawal and Srikant extended and generalized in [SA96] the sequential pattern mining paradigm with the concept of *Generalized Sequential Pattern Mining* (GSPM). The authors augmented the paradigm by introducing semantic and temporal relaxations, and temporal constraints. Hence, GSPM uncovers from collections of customer transactions sequential patterns where itemsets are expressed at different levels of abstraction. GSPM achieves this effect by means of (i) a semantic relaxation mechanism and (ii) a temporal relaxation mechanism. The semantic relaxation mechanism is responsible for expressing items in itemsets at different levels of granularity thanks to the use of taxonomies, i.e., *IS-A* hierarchies, that are assumed to be available to the user. In practice, itemsets are augmented with each item's antecedents taken from the taxonomies. This semantic relaxation allows to discover sequential patterns where items are expressed at different levels of taxonomy.

Temporal relaxation allows SPM algorithms to discover patterns that could not be discovered under the original definition of SPM. Temporal relaxation is expressed as a sliding window w within which a set of transaction can contribute to the support of a candidate pattern p. In other words, in the original definition of SPM, a pattern is supported by a sequence of transactions if each itemset in the pattern is supported by at least exactly one transaction in the sequence. Under the assumption of temporal relaxation, an itemset could be supported by the union of a set of at most w contiguous transactions.

Therefore, semantic relaxation allows GSPM algorithms to discover knowledge at different levels of abstraction otherwise hidden by the specificity of the data while temporal relaxation allows to uncover knowledge that can not be found due to the rigid definition of events in traditional SPM. For these reasons, GSPM can easily be understood as a form of higher order mining.

We presented in Chapter 2 the TSaR summarization approach whose purpose is to build summaries to support chronology-dependent applications such as Sequential Pattern Mining (SPM). TSaR is designed as a preprocessing step to allow process-intensive applications such as SPM to discover knowledge at different levels of representation. Thanks to the use of taxonomies, input event descriptors are expressed at different levels of taxonomy. From this prospect, when SPM is operated on TSaR summaries, patterns extracted are intuitively comparable to those extracted by GSPM techniques. Hence, mining sequential patterns in TSaR summaries can also be understood as a form of higher order mining. In other words, TSaR builds a support structure that enables higher order mining and allows the discovery of *Higher Order Knowledge* (HOK) in a way similar to GSPM. From these observations, we propose in this chapter to study how an off-the-shelf SPM algorithm could exploit in practice summaries produced by TSaR to extract higher order knowledge. The contributions discussed in this chapter are the following.

Contributions

First, we introduce the notion of *Higher Order Pattern* (HOP). We present a higher order pattern as a form of knowledge extracted from time sequence summaries using any conventional SPM technique. We analytically characterize higher order patterns extracted from TSaR summaries w.r.t. sequential patterns discovered from original time sequences.

Second, we propose a methodology called *Exploratory Mining* that uses higher order patterns discovered on TSAR summaries to uncover more specific patterns, i.e., patterns having even lower support. Exploratory mining is a drill-down process that relies on two mechanisms: *Pattern Events Recombination* (PER) and *Pattern Events Specialization* (PES). We detail these two mechanisms and discuss how they contribute to the task for discovering more refined patterns.

Third, we evaluate and validate our contributions through an extensive set of experiments on Reuters's financial news archives. The dataset chosen is a subset of Reuters dataset preprocessed in Chapter 2. The overall dataset represents 4600 time sequences and totals approximatively 100,000 news events. We implemented a conventional but effective SPM algorithm, namely PrefixSpan [PHMA+01], and performed a large series of mining experiments. The preliminary results obtained support our claims that a support structure is necessary for mining applications in domains such as finance. We show that (i) mining computational time explodes at low support and that (ii) the result set output might not be humanly exploitable. Then, we build TSAR summaries at different levels of content and temporal accuracy and mine the summaries built. We show that mining compact summaries can allow the discovery of trends that have very low support ($\approx 0.01\%$) while improving computational time by one order of magnitude. More importantly, we perform exploratory mining on the patterns discovered and achieve the goal of finding even more specific knowledge, i.e., patterns having even lower support.

Finally, we present a comprehensive *S*ervice-based *TE*mpor*A*l *D*ata analysis framework called STEAD for discovering knowledge in textual contents. The STEAD analysis framework is a set of online tools designed to provide analysts with a comprehensive environment to supervise and interactively perform higher order mining. One central component of the system is the summarization service that is built on top of time sequence summarization, e.g., TSAR. We will show how this framework is integrated in an even more ambitious project for efficient management of information resources over **A**d-Hoc **DA**ta **G**rids **E**nvironments, i.e., the ADAGE project.

Organization of the chapter

The remaining of the chapter is organized as follows. We extend in Section 2 the illustrative example used throughout this thesis. We add two more time sequences to the example. Section 3 recalls the basic principles of Sequential Pattern Mining and gives the definitions and notations that will be useful for characterizing knowledge discovered in summaries. Section 4 introduces the concept of *Higher Order Patterns* and characterizes the relationships that exist between higher order patterns discovered in summaries and sequential patterns that can be discovered in original sequences. The mechanisms of *Exploratory mining* are detailed in Section 5 and we discuss our experimental study in Section 6. Section 7 presents the STEAD framework and ADAGE project, and positions time sequence summarization in those environments. Related work is discussed in Section 8 and we conclude this chapter in Section 9.

2 Illustrative example

To illustrate the ideas exposed in this chapter, we extend our example already introduced in previous chapters. Example 4.1 presents three time sequences extracted from conference proceedings. The authors N. Koudas, N. Mamoulis and J. Pei are associated to a time sequence of events where each event is one publication timestamped by its date of

presentation. For simplicity, we make the same assumptions as previously, i.e., the set of descriptors used to describe an event is taken from one single descriptive domain, namely, the paper's *topic*. Without loss of generality, this discussion is valid for any number of descriptive domains. Again, we purposely choose this example unrelated to the application domain of our experiments in Section 6 to illustrate all the concepts introduced and show the genericity of our approach.

Example 4.1
Time sequences of events extracted from conference proceedings. Descriptors are taken from the topic *descriptive domain and represent the different topics covered by conference papers for the authors: N. Koudas, N. Mamoulis and J. Pei. These sequences are given in Table 4.1.*

Author	Event Descriptors	Time
N. Koudas	$x_{1,1}$ ={Datastreams, Aggregation}	JUN05
	$x_{1,2}$ ={Datastreams, Top-k query}	AUG06
	$x_{1,3}$ ={Top-k query}	AUG06
	$x_{1,4}$ ={Top-k query}	SEP06
	$x_{1,5}$ ={Join query, Selection query}	SEP06
	$x_{1,6}$ ={Clustering}	SEP07
N. Mamoulis	$x_{2,1}$ ={Datastreams, Join query}	JUN05
	$x_{2,2}$ ={Summarization}	AUG07
	$x_{2,3}$ ={Top-k query}	SEP07
	$x_{2,4}$ ={Anonymization}	SEP07
	$x_{2,5}$ ={Anonymization, Privacy}	AUG08
J. Pei	$x_{3,1}$ ={Skyline query}	AUG05
	$x_{3,2}$ ={Datastreams}	OCT05
	$x_{3,3}$ ={Mining}	AUG07
	$x_{3,4}$ ={Top-k query}	SEP07
	$x_{3,5}$ ={Privacy}	SEP07
	$x_{3,6}$ ={Skyline query}	SEP07

Table 4.1: Time sequences of events in conference proceedings

3 Sequential Pattern Mining (SPM)

In this section, we recall the basic principles of SPM and introduce the definitions and notations that will be used throughout the rest of this chapter.

3.1 Principles

Sequential Pattern Mining (SPM) is a data mining paradigm first proposed by Agrawal and Srikant [AS95,SA96]. The original purpose of sequential pattern mining is to identify customer behaviors thanks to the analysis of customer sales transactions. It is also known as the *market basket analysis* paradigm. Since its introduction in the data mining landscape, SPM has been very popular in any various domains including web usage mining, pattern discovery in protein sequences or mining XML query access patterns for caching and has attracted much more research efforts. We refer the reader to the following surveys for more indepth details and a larger coverage of the domain [ZB03,TPBM05,HPY05].

Informally, SPM considers a set of customers, or *objects of interest* in general, where each customer is associated to a list of transactions. Each transaction is characterized by the set of items purchased by the customer and a timestamp that indicates when the transaction took place. In this chapter, the list of transactions associated to each customer is called a *sequence* for short. Hence, a transaction database TDB, in the sense of market basket data, is a collection of customer sequences. The number of occurrences a transaction or a series of transactions appear in all customer sequences is called its *support*. Therefore, the Sequential Pattern Mining task consists in identifying all series of transactions that occur for a significant user-defined number of sequences. This user-defined number is called the *minimum support* and series of transactions that achieve the minimum support in the TDB are called *sequential patterns* or *patterns* for short. Note that this minimum support is often expressed as a fraction of sequences in the TDB. For simplicity, we simply express the minimum support as an integer value that represents the number of sequences in the TDB that needs to support a pattern. Two examples of sequential patterns SPM are given in Example 4.2 and Example 4.3.

Example 4.2
Suppose the TDB is the customer transaction history of a book store. We can find frequent sequential patterns of purchases as follows:

"80% of customers who bought the book Database Management typically bought the book Data Warehouse and then bought the book Web Information System within a certain period of time inbetween each transaction.", or:

"80% of customers who bought the book Intro to Visual C++ typically bought the book C++ Primer and then bought the books Perl for Dummies and TCL/TK." Identifying such purchase patterns in a book store would allow to strategically place books: Book on web information systems could be placed in the same section as books on data management and Perl for Dummies could be placed closer to books that relate to C++ programming.

Example 4.3
Suppose the TDB is the history of IT companies' stock prices. An example of sequential pattern would be:

"Every time Microsoft's stock price drops by 5%, IBM's stock price will also drop by at least 4% within three days." The knowledge of such trends would allow analysts to anticipate market movements and consolidate their investments.

An interesting observation one can make through our informal definition of SPM is that the data model of data used in SPM algorithms completely matches our definition of a time sequence of events as given in Definition 1.5 in Chapter 1. This observation allows us to use all the concepts and terminology already introduced and employed for time sequences of events. Therefore, in the rest of this chapter, we will interchangeably and equivalently use terminology from both domains. For instance, a pattern in a collection of time sequences of events is a series of events that is supported by a given number of time sequences. In this case, the notion of support of an event is reduced to the support of the event's content, i.e., its set of descriptors. This specificity aside, all concepts are interchangeably applicable.

3.2 Definitions and notations

In previous section, we have laid out the intuition of the SPM paradigm. Here, we will formally detail the concepts on which SPM relies. These concepts and their formalization will be useful for characterizing knowledge that can be discovered in time sequences, generalized time sequences and time sequence summaries. Thus, Sequential Pattern Mining

relies on the notion of *subsequence* and the notion of *support* of a subsequence. We detail these two concepts in Definition 4.1 and Definition 4.2, respectively.

Definition 4.1 (Subsequence)
Given s a time sequence of events in $\mathcal{S}(\Omega)$, $s = \{e_1, \ldots, e_n\}$ with $e_i = (x_i, t_i)$, a sequence s_{sub}, $s_{sub} = \{e'_1, \ldots, e'_m\}$ with $e'_j = (x'_j, t'_j)$, is a subsequence of s and we note $s_{sub} \sqsubseteq s$ i.f.f. there exist m integers $i_1 < \ldots < i_m$ such that, for all events $e'_1, \ldots, e'_m$ in s_{sub}, exists events $e_{i_1}, \ldots, e_{i_m}$ in s such that $x'_1 \subseteq x_{i_1} \wedge \ldots \wedge x'_m \subseteq x_{i_m}$.

For example, if $s_{sub} = \{(A, t_1), (B, t_2), (C, t_3)\}$ and if we have $s_{sub} \sqsubseteq s$, then s has the form $s = \{*(A, t'_1) * (B, t'_2) * (C, t'_3)*\}$, where $*$ represents any sequence of events, possibly empty. We also equivalently say that "*s contains s_{sub}*" or "*s_{sub}* is contained in *s*".

Definition 4.2 (Support of a subsequence)
Given E a collection of time sequences of events, the support of a subsequence s, denoted $supp_E(s)$, is the number of sequences of E that contain s:

$$supp_E(s) = |\{s' \in E,\ s \sqsubseteq s'\}|$$

For instance, in our illustrative example, the series of events $p = \{(\{$"Top-k query"$\}, t_1), (\{$"Privacy"$\}, t_2)\}$ is a subsequence of N. Mamoulis's time sequence and of J. Pei's time sequence. Indeed:

- In the case of N. Mamoulis's time sequence, itemset {"Top-k query"}$\subseteq x_{2,3}$={"Top-k query"} at $t = SEP07$ and itemset {"Privacy"}$\subseteq x_{2,5}$={"Anonymization", "Privacy"} at $t = AUG08$.
- In the case of J. Pei's time sequence, itemset {"Top-k query"}$\subseteq x_{3,4}$={"Top-k query"} at $t = SEP07$ and itemset {"Privacy"}$\subseteq x_{3,5}$={"Privacy"} at $t = SEP07$.

Since pattern p is a subsequence of 2 time sequences, the support of p is 2 (or equivalently $\frac{2}{3}$ if using the fraction notation in traditional literature on sequential pattern mining). One should note that a pattern p can only be supported once by a sequence s, i.e., if s contains multiple occurrences of pattern p, e.g., 3 occurrences, its support will only be 1 (and not 3). Equipped with these definitions of a subsequence and its support in a collection of time sequences, we can define the notion of *sequential pattern* and the actual task of *sequential pattern mining* in Definition 4.3 and Definition 4.4, respectively.

Definition 4.3 (Sequential pattern)
Given E a collection of time sequences of events and given an integer value γ, a sequential pattern (or pattern *for short) p is a subsequence of at least γ sequences in E, i.e., $supp_E(p) \geq \gamma$.*

Definition 4.4 (Sequential Pattern Mining task)
Given E a collection of time sequences of events and a user-defined minimum support γ, the Sequential Pattern Mining task is the task of extracting the set of all patterns p from E that achieve the minimum support γ. This set of patterns having minimum support γ in E is denoted $P_\gamma(E)$ and defined as follows:

$$P_\gamma(E) = \{p \in \mathcal{S}(\Omega),\ supp_E(p) \geq \gamma\}$$

It is worth to mention that the definition of a subsequence given in Definition 4.1 considers a sequence p to be a subsequence of a sequence s, i.e., $p \sqsubseteq s$, when itemsets in p are contained in larger itemsets in s. In other words, this definition of subsequence does not

consider each itemset in p as a unique item. For instance, suppose candidate pattern p is defined as $p = \{e'\}$, where the itemset associated to e' is {"Top-k query"}. Suppose s is a sequence where $s = \{e\}$ and the itemset associated to e is {"Top-k query", "Datastreams"}. Under Definition 4.1, p is a subsequence of s since {"Top-k query"} is contained in {"Top-k query", "Datastreams"}.

However, in this work, we limit the definition of SPM to mining time sequences where the itemset associated to each event is considered as a *single-item*. Under this assumption, pattern $p = \{e'\}$ is not a subsequence of $s = \{e\}$ since itemset {"Top-k query"}≠{"Top-k query", "Datastreams"}.

The reasons why we consider time sequences of single-items are the following: (i) single-item time sequences represent one of the most important and popular type of sequence, e.g., DNA sequences, strings, web click streams etc. [WH04]; (ii) in financial news domain, we believe that the set of descriptors x that describes the content of a piece of news event e is the atomic unit for understanding event e. In other words, a part of x is not enough to grasp the entire content of the news. Therefore, we revisit the definition of a subsequence as given in Definition 4.5. All other definitions remain unchanged and completely compatible.

Definition 4.5 (Subsequence - revisited)
Given s a time sequence of events in $\mathcal{S}(\Omega)$, $s = \{e_1, \ldots, e_n\}$ with $e_i = (x_i, t_i)$, a sequence $s_{sub} = \{e'_1, \ldots, e'_m\}$ with $e'_j = (x'_j, t'_j)$, is a subsequence of s and we note $s_{sub} \sqsubseteq s$ i.f.f. there exist m integers $i_1 < \ldots < i_m$ such that, for all events $e'_1, \ldots, e'_m$ in s_{sub}, there exist events $e_{i_1}, \ldots, e_{i_m}$ in s such that $x'_1 = x_{i_1} \wedge \ldots \wedge x'_m = x_{i_m}$.

For instance, let us consider again our previous example where p is defined as the series of events ({"Top-k query"}, t_1) followed by ({"Privacy"}, t_2). Under Definition 4.1, p's support equals 2 since p is a subsequence of N. Mamoulis's time sequence and of J. Pei's time sequence. Under definition 4.5, p is not a subsequence of N. Mamoulis's time sequence anymore since itemset {"Privacy"} at t_2 in p is not equal to itemset $x_{2,5}$, i.e., {"Privacy"}≠{"Anonymization", "Privacy"}, at $t = AUG08$. Therefore, the support of pattern p in our illustrative example drops to 1.

Source	**Length=1**	**Length=2**
Raw γ=3	{"Top-k query"}	N/A
Raw γ=2	{"Top-k query"}	N/A

Table 4.2: Example of patterns discovered on raw time sequences under Definition 4.5

Also, we refine the definition of a sequential pattern to characterize patterns extracted by their length. Definition 4.6 restricts the definition of a sequential pattern to patterns of length k, with $k \geq 1$. The restricted definition of a sequential pattern to a given length is necessary for characterizing in details patterns that are extracted from summaries.

Definition 4.6 (Sequential patterns of length k)
Given E a collection of time sequences of events, the set of sequential patterns of length $k \geq 1$ having minimum support $\gamma \geq 1$ that can be extracted from E is denoted $P^k_\gamma(E)$ and is defined as follows:

$$P^k_\gamma(E) = \{p \in P_\gamma(E) \text{ such that } |p| = k\}$$

Note that this definition allows to rewrite the definition of $P_\gamma(E)$ as follows:
$P_\gamma(E) = \bigcup_{k \in [1 \ldots m]} P^k_\gamma(E)$, where m is the maximum length of sequential patterns in $P_\gamma(E)$.

Therefore, under Definition 4.5, applying a traditional SPM algorithm on our illustrative example would give patterns presented in Table 4.2. The sequential patterns extracted under Definition 4.5 limits the number of patterns that could be extracted. For instance, pattern $p = \{(\{\text{“Datastreams”}\}, t_1), (\{\text{“Top-k query”}\}, t_2)\}$ is a sequential pattern having support $\gamma = 3$ under Definition 4.1, but it only has support $\gamma = 1$ under Definition 4.5. In fact, in our example no pattern of length $k \geq 2$ could be extracted under Definition 4.5.

For convenience, we remind in Figure 4.1 the taxonomy for the *topic* descriptive domain already presented in previous chapters. When observing this taxonomy, we can note that concepts “Anonymization” and “Privacy” could be represented by a unique concept which is “Security”. This observation is frustrating when knowing that generalizing descriptors “Anonymization” and “Privacy” into descriptor “Security” would allow a traditional SPM algorithm to capture the pattern $p' = \{(\{\text{“Top-k query”}\}, t_1), (\{\text{“Security”}\}, t_2)\}$, where pattern p' can be understood as a more abstract pattern than p, or a *higher order pattern*. This is the intuition that will guide our work in Section 4.

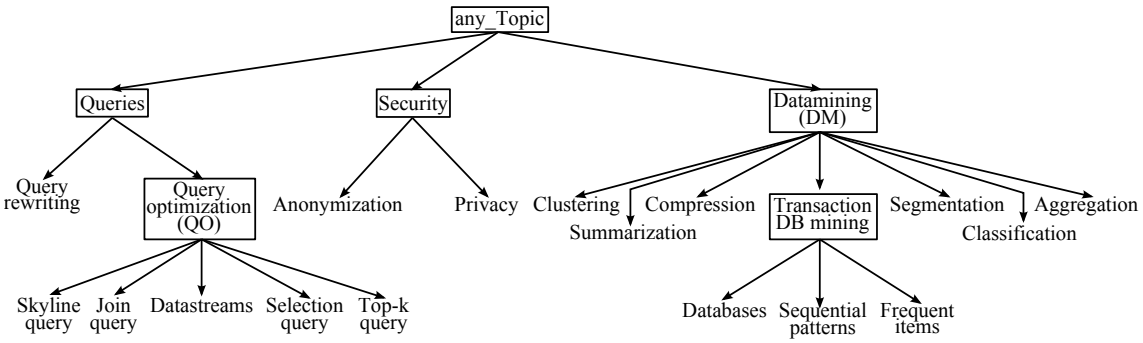


Figure 4.1: Taxonomy for the *topic* descriptive domain

In this section, we have introduced all the definitions necessary for understanding and characterizing the knowledge, i.e., sequential patterns, that can be extracted from a collection of time sequences of events. We have pinpointed through our examples some limitations, in terms of the knowledge that could be discovered, of traditional Sequential Pattern Mining algorithms. Therefore, we propose in the rest of this chapter to support traditional SPM algorithms with time sequence summaries. For this purpose, we need to (i) study how time sequence summaries can contribute in practice to sequential pattern mining and (ii) characterize the knowledge that can be extracted from summaries. This is the subject of Section 4. For convenience, we sum up in the Table 4.3 all notations previously introduced and notations that will be used in the rest of this chapter.

Notation	**Meaning**
E=$\{s \in \mathcal{S}(\Omega)\}$	Collection of raw time sequences of events
$\mathcal{E}$=$\{\varphi_\vartheta(s) \in \mathcal{S}(\Omega), s \in E\}$	Generalization of E by φ_ϑ, also denoted $\mathcal{E} = \varphi_\vartheta(E)$
$\mathbf{E}$=$\{\chi_{\vartheta,w}(s), s \in E\}$	Summary of E by $\chi_{\vartheta,w}$, also denoted $\mathbf{E} = \chi_{\vartheta,w}(E)$
$P_\gamma(E)$=$\{p \in \mathcal{S}(\Omega),\ supp_E(p) \geq \gamma\}$	Patterns having minimum support γ in E
$P^k_\gamma(E)$=$\{p \in P_\gamma(E), \lvert p\rvert = k\}$	Patterns of length k having minimum support γ in E
$P_\gamma(\mathcal{E})$=$\{p \in \mathcal{S}(\Omega),\ supp_\mathcal{E}(p) \geq \gamma\}$	Patterns having minimum support γ in $\mathcal{E}$
$P^k_\gamma(\mathcal{E})$=$\{p \in P_\gamma(\mathcal{E}), \lvert p\rvert = k\}$	Patterns of length k having minimum support γ in $\mathcal{E}$
$P_\gamma(\mathbf{E})$=$\{p \in \mathcal{S}(\Omega), supp_\mathbf{E}(p) \geq \gamma\}$	Patterns having minimum support γ in $\mathbf{E}$
$P^k_\gamma(\mathbf{E})$=$\{p \in P_\gamma(\mathbf{E}), \lvert p\rvert = k\}$	Patterns of length k having minimum support γ in $\mathbf{E}$
$\varphi_\vartheta(P_\gamma(E)))$=$\{\varphi_\vartheta(p) \in \mathcal{S}(\Omega), p \in P_\gamma(E)\}$	Generalization of patterns discovered on E by φ_ϑ

Table 4.3: Notations used for operating SPM on time sequence summaries

4 Higher Order Patterns (HOP): Patterns mined from summaries

TSAR was designed as a support structure for chronology dependent applications, i.e., applications that require the time dimension of the data to be considered for the output to be meaningful, on massive data sources. Sequential Pattern Mining is one such application. Hence, mining sequential patterns on TSAR summaries can be understood as a form of Higher Order Mining. Sequential patterns discovered in this manner are called *Higher Order Patterns* (HOP). We define higher order patterns in Section 4.1 and throughly analyze higher order patterns that can be discovered in TSAR summaries in Section 4.2.

4.1 Definition

Roddick et al. coined the term *Higher Order Mining* in [RSLC08] to refer to data mining tasks operated on non-primitive data:

> "Higher Order Mining (HOM) is the sub-field of knowledge discovery concerned with mining over patterns/models derived from one or more large and/or complex datasets."

The purpose of this chapter is to study how time sequence summaries can benefit the specific task of mining sequential patterns. The idea is to represent large sources of time-varying data at a higher level of abstraction then operate a conventional SPM technique to discover some form of knowledge. Intuitively, TSAR is a good candidate for this task since it uses the data's content and temporal information to produce summaries that reflect a user's understanding of the domains on which the data is defined. Hence, we extend Roddick et al.'s terminology and define patterns extracted from time sequence summaries as *Higher Order Patterns* (HOP) in Definition 4.7.

Definition 4.7 (Higher Order Pattern)
Let us be given E a collection of time sequences of events, i.e., $E = \{s \in \mathcal{S}(\Omega)\}$, χ a time sequence summarization function and $\mathbf{E} = \chi(E) = \{\chi(s), s \in E\}$. For a given user defined minimum support γ, a Higher Order Pattern *(HOP) is a pattern taken from the set $P_\gamma(\mathbf{E})$.*

In the following section, we will thoroughly study the implications of mining sequential patterns in time sequence summaries w.r.t. the knowledge that could be uncovered in the original datasets.

4.2 Characterization

We characterize higher order patterns discovered in TSAR summaries and evaluate how well higher order patterns are linked to knowledge discovered in raw time sequences. We analytically describe as thoroughly as possible the following relationships:

- Patterns discovered in E a collection of raw sequences vs. higher order patterns discovered in $\mathcal{E}$ the collection of generalized time sequences, i.e., time sequences summarized with parameters $\vartheta = \langle k \rangle$, with $k \geq 1$, and $w = 0$.
- Higher order patterns discovered in $\mathcal{E}$ a collection of generalized sequences vs. higher order patterns discovered in $\mathbf{E}$ the collection of summarized time sequences, i.e., generalized time sequences in $\mathcal{E}$ summarized with parameter $\vartheta = \langle 0 \rangle$ and $w \geq 1$.
- Patterns discovered in E a collection of raw sequences vs. higher order patterns discovered in $\mathbf{E}$ the collection of summarized time sequences.

However, due to the use of a generalization vector $\vartheta = \langle k \rangle$ with $k \geq 1$, one should note that in generalized and in summarized time sequences, events are described by descriptors represented at higher levels of taxonomy than descriptors used to describe events in raw time sequences. Hence, higher order patterns can not be directly compared to patterns discovered in raw time sequences. Some transformation needs to be operated beforehand. For this reason, we propose to represent events of patterns in $P_\gamma(E)$ at the same level of taxonomy as events in $\mathcal{E}$ and $\mathbf{E}$. This transformation is legitimate since both sets of patterns are (i) expressed at the same level of taxonomy and (ii) patterns in $\varphi_\vartheta(P_\gamma(E))$ remain unchanged from temporal view point.

We generalize all patterns discovered in $P_\gamma(E)$ with the same generalization vector used to generate $\mathcal{E}$ and $\mathbf{E}$. Consequently, our study and analysis task comes down to characterizing the relationships that link:

1. $\varphi_\vartheta(P_\gamma(E))$ to $P_\gamma(\mathcal{E})$
2. $P_\gamma(\mathcal{E})$ to $P_\gamma(\mathbf{E})$
3. $\varphi_\vartheta(P_\gamma(E))$ to $P_\gamma(\mathbf{E})$

Remind that notations are summed up in Table 4.3.

4.2.1 Patterns in raw sequences vs. HOP in generalized sequences

We study the relationship that links sequential patterns discovered in raw time sequences to higher order patterns discovered in generalized time sequences. We highlighted above the fact that sequential patterns discovered in raw time sequences need to be represented at the same level of taxonomy as sequences in $\mathcal{E}$ and $\mathbf{E}$. This transformation is adamant so that patterns become comparable. We give in Property 4.1 a direct consequence of generalization on the cardinality of $\varphi_\vartheta(P_\gamma(E))$.

Property 4.1 (Cardinality reduction of $\varphi_\vartheta(P_\gamma(E))$)
Given E a collection of time sequences of events, the following inequality occurs:

$$|\varphi_\vartheta(P_\gamma(E))| \leq |P_\gamma(E)| \tag{4.1}$$

This property can be refined and expressed at the granularity of sequential patterns of length $k \geq 1$. In this case, this property is expressed as follows:

$$|\varphi_\vartheta(P^k_\gamma(E))| \leq |P^k_\gamma(E)|, \ m \text{ maximum length of sequential patterns discovered in } E \tag{4.2}$$

Property 4.1 states that the number of patterns obtained in $\varphi_\vartheta(P_\gamma(E))$ after generalization is reduced w.r.t. the number of patterns originally in the set $P_\gamma(E)$. This effect results from the capability of the φ_ϑ operator to reduce the variability of event descriptors. For instance, in our example, suppose $P_\gamma(E) = \{p_1, p_2\}$ with $p_1 = \{(\{\text{“Top-k query”}\}, t_1), (\{\text{“Privacy”}\}, t_2)\}$ and $p_2 = \{(\{\text{“Top-k query”}\}, t_1), (\{\text{“Anonymization”}\}, t_2)\}$. We showed earlier with the taxonomy in Figure 4.1 that concepts “Privacy” and “Anonymization” could be generalized into concept “Security”. Therefore, a possible generalization of $P_\gamma(E)$ could be $\{p'\}$ with $p' = \{(\{\text{“Top-k query”}\}, t_1), (\{\text{“Security”}\}, t_2)\}$, and $|\{p'\}| < |P_\gamma(E)|$.

In order to establish the relationship that links sequential patterns discovered in raw time sequences to higher order patterns discovered in generalized time sequences, we need to study their respective support. We give compare in Theorem 4.1 the support of a pattern in a collection of raw time sequences to its support in a collection of generalized time sequences.

Theorem 4.1 (Support of patterns in raw vs. generalized sequences)
Given E a collection of time sequences of events and $\mathcal{E}$ the generalization of E by φ_ϑ, the following inequality occurs:

$$\forall p \in \mathcal{S}(\Omega),\ supp_E(p) \leq supp_{\mathcal{E}}(\varphi_\vartheta(p)) \tag{4.3}$$

Proof 7 (Proof of Theorem 4.1)
The proof is straightforward and relies on the monotonicity of the generalization function φ_ϑ: $\forall s \in E, \forall p \in E,\ p \sqsubseteq s \Rightarrow \varphi_\vartheta(p) \sqsubseteq \varphi_\vartheta(s)$. Hence, if s supports the candidate pattern p in E, then $\varphi_\vartheta(s)$ supports the candidate pattern $\varphi_\vartheta(p)$ in $\mathcal{E}$. From Definition 4.2, we obtain $supp_E(p) \leq supp_{\mathcal{E}}(\varphi_\vartheta(s))$. The situation where p is not contained in any sequence s of E (i.e., $supp_E(p) = 0$) trivially states that $supp_{\mathcal{E}}(\varphi_\vartheta(p)) \geq 0$. □

Theorem 4.1 states that any pattern p in E has lower support than its corresponding generalized candidate pattern $\varphi_\vartheta(p)$ in $\mathcal{E}$; this effect is a consequence of generalization. As a direct rule, for a given minimum support γ, every sequential pattern p in raw time sequences in E has an image by φ_ϑ, i.e., $\varphi_\vartheta(p)$, that is also a higher order pattern in $\mathcal{E}$. This property is given in Corollary 4.1.

Corollary 4.1 (Patterns in raw vs. HOP in generalized sequences)
Given E a collection of time sequences of events and $\mathcal{E} = \varphi_\vartheta(E)$ the generalization of E by φ_ϑ, the following containment relation occurs:

$$\varphi_\vartheta(P_\gamma(E)) \subseteq P_\gamma(\mathcal{E}) \tag{4.4}$$

Proof 8 (Proof of Corollary 4.1)
$\forall p' \in \varphi_\vartheta(P_\gamma(E)),\ \exists\ p \in P_\gamma(E),\ p' = \varphi_\vartheta(p) \wedge supp_E(p) \geq \gamma$ [Def. 4.3]. *From Theorem 4.1, one can say that $supp_E(p) \leq supp_{\varphi_\vartheta(P_\gamma(E))}(p')$. Hence, $supp_{\varphi_\vartheta(P_\gamma(E))}(p') \geq \gamma$; that is the condition for $p \in P_\gamma(\mathcal{E})$* [Def. 4.3]. □

A direct restriction of Corollary 4.1 can be given using the restriction of sequential patterns to patterns of a length k. Corollary 4.2 restricts Corollary 4.1 and states that every sequential pattern of length k in a collection of raw time sequences E has an image by φ_ϑ that is also a higher order pattern of length k in the set of generalized time sequences $\mathcal{E}$.

Corollary 4.2 (Patterns in raw vs. HOP in generalized sequences/restricted)
Given E a collection of time sequences of events and $\mathcal{E} = \varphi_\vartheta(E)$ the generalization of E by φ_ϑ, the following containment relation occurs:

$$\varphi_\vartheta(P^k_\gamma(E)) \subseteq P^k_\gamma(\mathcal{E}) \tag{4.5}$$

Proof 9 (Proof of Corollary 4.2)
Proof is identical to Proof of Corollary 4.1 using the $P^k_\gamma(E)$ notation instead. □

The relationship that links sequential patterns discovered in raw time sequences to higher order patterns discovered in generalized time sequences is intuitive and straightforward. It simply relies on the fact that generalizing a time sequence reduces the variability of events descriptors and increases the support of events. Since no grouping is operated, there is no loss of temporal accuracy and hence no loss of patterns at all. However, note that since generalization is responsible of the increase of events' support, for a given minimum support γ, mining sequential patterns in generalized sequences will output many more patterns. Thus, the mining process could require longer computational time. In the following section, we focus on the relationship that links higher order patterns discovered in generalized time sequences to higher order patterns discovered in summarized time sequences.

4.2.2 HOP in generalized sequences vs. HOP in summarized sequences

In this section, we study the relationship that links higher order patterns discovered in generalized time sequences to higher order patterns discovered in summarized time sequences. Before we can present this relationship, we need to introduce in Property 4.2 an important containment property on the $\pi \circ \psi_w$ operations of the TSaR summarization function.

Property 4.2 (Subsequence containment in a summary intent)
Given s a time sequence of events in $\mathcal{S}(\Omega)$, s' its generalization by φ_ϑ and $s^\star$ its summary intent, i.e., $s^\star = \pi \circ \psi_w(s')$, the following implication occurs:

$$\forall p \in \mathcal{S}(\Omega),\ p \sqsubseteq s^\star \ \Rightarrow\ p \sqsubseteq s' \tag{4.6}$$

Proof 10 (Proof of Property 4.2)
Since $p \sqsubseteq s^\star$, for all $e = (x,t) \in p$, $\exists\ e^\star = (x^\star, t^\star) \in s^\star$ such that $x = x^\star$. The semantic preservation given in Property 2.1 in Chapter 2 allows to write that there exists a set of events $Y = \{e'_1, \ldots, e'_r,\} \sqsubseteq s'$ with $e'_k = (x'_k, t'_k)$ such that $x = x'_1 = \ldots = x'_r$ and $t^\star_k = t'_1$.

Therefore, there exist m integers values $i_1 < \ldots < i_m$ such that, for all events $e_1, \ldots, e_m$ in p, exist sets $Y_1, \ldots, Y_m$ such that $x_1 = x'_{1,1} = \ldots = x'_{1,r},\ \ldots,\ x_m = x'_{m,1} = \ldots = x'_{m,r}$. This is the condition for $p \sqsubseteq s'$ as given in Definition 4.5. We proved Property 4.2. □

This property is necessary for characterizing the support of patterns in generalized sequences vs. patterns in summarized sequences. We state in Property 4.2 that given a time sequence s and a grouping $\psi_w(s)$ of s, any subsequence p in $\pi \circ \psi_w(s)$ is actually a subsequence of s. In other words, this property states that if a pattern is supported by a grouped-and-projected sequence, it is also supported by the non-grouped-and-projected counterpart.

Now, equipped with this property, we can establish the relationship that links higher order patterns discovered in generalized time sequences to higher order patterns discovered in summarized time sequences. For this purpose, we study their respective support. We compare in Theorem 4.2 the support of a pattern in a collection of generalized time sequences to its support in a collection of summarized time sequences.

Theorem 4.2 (Support of HOP in generalized. vs. in summarized sequences)
Given E a collection of time sequences of events, $\mathcal{E}$ the generalization of E by φ_ϑ and $\mathbf{E}$ the summary of E by $\chi_{\vartheta,w}$, the following inequality occurs:

$$\forall p \in \mathcal{S}(\Omega),\ supp_{\mathbf{E}}(p) \leq supp_{\mathcal{E}}(p) \tag{4.7}$$

Proof 11 (Proof of Theorem 4.2)
Suppose $supp_{\mathcal{E}}(p) = k$, with $k \geq 1$. Let us denote by $S_k = \{s_1, \ldots, s_k\}$ the set of generalized sequences in $\mathcal{E}$ that support pattern p of length k, i.e., $S_k \subseteq \mathcal{E}$ and $\forall s \in S_k, p \sqsubseteq s$. We denote $S^\star = \{s^\star = \pi \circ \psi_w(s), s \in S_k\}$, with $w \geq 0$, the summarization of time sequences in S_k. Since there are k sequences in S_k there are also k summarized sequences in $S^\star$, i.e., $|S^\star| = |S_k|$. Also, we denote by $S^\dagger$ the set of summarized time sequences in $\mathbf{E}$ that support pattern p and thus, $S^\dagger \subseteq S^\star$ and $|S^\dagger| \leq |S^\star|$. Proving Theorem 4.2 is equivalent to proving the following inequality: $|S^\dagger| \leq |S_k|$.

When $w = 0$, there is no event rearrangement and no loss of temporal accuracy. Hence, this inequality is trivial and the sets S_k, $S^\star$ and $S^\dagger$ are identical and thus $|S^\dagger| = |S_k|$.

When $w \geq 1$, summarizing sequences in E is equivalent to operating ψ_w on all sequences in $\mathcal{E}$ since $\mathcal{E} = \varphi_\vartheta(E)$. Having $w \geq 1$, groupings of distant events on the timeline might

be operated by the ψ_w operator on sequences in S_k. In the case such operations occur, the number of events in sequences in S_k could be reduced, formally:
$\forall s \in S_k$, let $s^\star = \pi \circ \psi_w(s)$ then $s^\star \in S^\star$ and $s^\star \subseteq s$ (Property 4.2)
if $\exists\ s' \in S^\star$, such that $p \not\sqsubseteq s'$,
then necessarily, $s' \notin S^\dagger$ and consequently $|S^\dagger| < |S^\star|$, i.e., $|S^\dagger| < |S_k|$
otherwise, $|S^\dagger| = |S^\star| = |S_k|$

In any case, we proved the inequality $|S^\dagger| \leq |S_k|$ and Theorem 4.2. □

Theorem 4.2 states that any candidate pattern p in $\mathbf{E}$ has lower support in $\mathbf{E}$ than in $\mathcal{E}$, i.e., there exist fewer sequences s in $\mathbf{E}$ for which $p \sqsubseteq s$. Consequently, we give in Corollary 4.3 the actual containment relationship that links higher order patterns discovered in generalized sequences to higher order patterns discovered in summarized sequences.

Corollary 4.3 (HOP in generalized seq. vs. HOP in summarized sequences)
Given E a collection of time sequences of events, $\mathcal{E}$ the generalization of E by φ_ϑ and $\mathbf{E}$ the summary of E by $\chi_{\vartheta,w}$, the following containment relation occurs:

$$P_\gamma(\mathbf{E}) \subseteq P_\gamma(\mathcal{E}) \tag{4.8}$$

Proof 12 (Proof of Corollary 4.3)
In Table 4.3, we defined $P_\gamma(\mathbf{E})$ as follows: $P_\gamma(\mathbf{E}) = \{p \in \mathcal{S}(\Omega),\ supp_{\mathbf{E}}(p) \geq \gamma\}$. Therefore, we can write the following equations:
$\forall p \in P_\gamma(\mathbf{E})$, $\exists\ S^\star = \{s_1^\star, \ldots, s_m^\star\}$, with $m \geq \gamma$,
where $\forall s \in \mathbf{E}$, $p \sqsubseteq s$ with $s = \pi \circ \psi_w(s')$, and $s' \in \mathcal{E}$.
Therefore, $p \subseteq s'$ and $s' \in \mathcal{E}$ (as a direct consequence of Property 4.2)
Let us denote by $S = \{s_1, \ldots, s_m\}$ the set of sequences in $\mathcal{E}$ that support pattern p.
Hence, $\forall s \in S$, $\exists\ s^\star \in S^\star$, $s^\star = \pi \circ \psi_w(s)$ and $p \sqsubseteq s$.
Since we proved in Theorem 4.2 that $\forall p \in \mathcal{S}(\Omega)$, $supp_{\mathbf{E}}(p) \leq supp_{\mathcal{E}}(p)$
then S is only a subset of $P_\gamma(\mathcal{E})$, i.e., $S \subseteq P_\gamma(\mathcal{E})$.
Putting the pieces together, we proved that for any pattern p in $P_\gamma(\mathbf{E})$ there exists a set S' of sequences in $\mathcal{E}$ that supports p, and this set S' is a subset of $P_\gamma(\mathcal{E})$. We proved that $P_\gamma(\mathbf{E}) \subseteq P_\gamma(\mathcal{E})$. □

The general rule is that mining sequential patterns in generalized sequences allows to capture all patterns that could be discovered in summarized sequences. This relationship can be refined and detailed by considering the length k of the patterns discovered. This refinement is given in Corollary 4.4.

Corollary 4.4 (HOP in generalized vs. HOP in summarized sequences/k=1)
Given E a collection of time sequences of events, $\mathcal{E}$ the generalization of E by φ_ϑ and $\mathbf{E}$ the summary of E by $\chi_{\vartheta,w}$, the following equality occurs:

$$P_\gamma^1(\mathcal{E}) = P_\gamma^1(\mathbf{E}) \tag{4.9}$$

Proof 13 (of Corollary 4.4)
Theorem 4.4 is proved thanks to the semantic preservation of Property 2.1 given in Chapter 2. Property 2.1 allows us to write: $\forall s \in \mathcal{E}$, $\forall e = (x,t) \in s$, $\exists\ e^\star = (x^\star, t^\star) \in \pi \circ \psi_w(s)$, such that $x = x^\star$. Therefore, if s supports pattern $\{e\}$ then sequence $\pi \circ \psi_w(s)$ supports pattern $\{e\}$. □

Corollary 4.4 refines this containment relationship to patterns containing one single event, i.e., $k = 1$. In fact, this refined relationship shows that patterns of length 1 captured

in generalized or summarized patterns are identical. On the other hand, Corollary 4.5 is a simple rewriting of the containment relationship in Corollary 4.3 to account for the length of patterns with $k \geq 2$. This case is more general and patterns discovered deeply depend on the distribution of the data and the summarization parameters, in particular w. For this reason we believe it is difficult if not impossible to analytically characterize any more precisely this containment relationship.

Corollary 4.5 (HOP in generalized vs. HOP in summarized sequences/$k \geq 2$) *Given E a collection of time sequences of events, $\mathcal{E}$ the generalization of E by φ_ϑ and $\mathbf{E}$ the summary of E by $\chi_{\vartheta,w}$, the following equality occurs:*

$$P_\gamma^k(\mathbf{E}) \subseteq P_\gamma^k(\mathcal{E}), k \geq 2 \tag{4.10}$$

Let us give an example to support our claim. We show in the following example that even for a pattern of length $k = 2$, it is not possible to give a definitive containment relationship. Suppose p is a higher order pattern in $\mathcal{E}$ with $p = \{(x_1, t_1), (x_2, t_2)\}$. Suppose pattern p is supported in $\mathcal{E}$ by only 2 sequences, namely s_1 and s_2. s_1 is defined as follows: $s_1 = \{\ldots, (x_1, t_1), (x_2, t_2)\}$ and $\forall e = (x, t) \in s_1$ with $t < t_1,\ x \neq x_1\ \wedge x \neq x_2$. In other words, there is one unique subsequence in s_1 that supports pattern p. Similarly, s_2 is defined as follows:
$s_2 = \{\ldots, (x_2, t_1), (x_1, t_2), (x_2, t_3)\}$ and $\forall e = (x, t) \in s_2$ with $t < t_1,\ x \neq x_1\ \wedge x \neq x_2$. Therefore, the summarization of s_1 and s_2 with temporal locality parameter $w = 2$ gives:
$s_1^\star = \pi \circ \psi_{w=2}(s_1) = \{\ldots (x_1, t_1), (x_2, t_2)\}$ and $s_2^\star = \pi \circ \psi_{w=2}(s_2) = \{\ldots (x_2, t_1'), (x_1, t_2)\}$. Consequently, pattern p of length $k = 2$ is supported by s_1 and s_2, and by $s_1^\star$ but not by $s_2^\star$. This is a counter-example illustrating the fact it is not possible to analytically characterize further Corollary 4.5.

4.2.3 Patterns in raw time sequences vs. HOP in summarized sequences

Most interesting is the relationship that links sequential patterns discovered in raw time sequences to higher order patterns discovered in summarized time sequences. However, this relationship is not trivial and deeply depends on the distribution of the data and the summarization parameters, in particular w. We present in the following section the few characteristics that are actually provable.

We showed in Property 4.3 that higher order patterns of length 1 mined in TSAR summaries capture patterns of length 1 that can be discovered in raw time sequences. As simple as this relationship might appear, generalizing this relationship to $k \geq 2$ is a very difficult task, if not impossible. Intuitively, the grouping function in TSAR reduces the numerosity of events in a time sequence. Therefore, a pattern $p \in \varphi_\vartheta(P_\gamma(E))$ that was supported by a sequence s in $\mathcal{E}$ could eventually not be supported by sequence $s^\star = \pi \circ \psi_w(s)$ in $\mathbf{E}$. In other words, the support of pattern p in $\mathbf{E}$ is reduced due to the shortening of sequences in $\mathbf{E}$. Similarly to Section 4.2.2, this *loss of support* deeply depends on the summarization parameters, the SPM algorithm parameters and the data themselves.

Property 4.3 (Patterns in raw vs. HOP in summarized sequences/k=1) *Given E a collection of time sequences of events, $\mathcal{E}$ the generalization of E by φ_ϑ and $\mathbf{E}$ the summary of E by $\chi_{\vartheta,w}$, the following containment relation occurs:*

$$\varphi_\vartheta(P_\gamma^1(E)) \subseteq P_\gamma^1(\mathbf{E}) \tag{4.11}$$

Proof 14 (of Property 4.3)
The proof of this theorem is straightforward. Corollary 4.4 states that $P_\gamma^1(\mathcal{E}) = P_\gamma^1(\mathbf{E})$

and Corollary 4.2 states that $\varphi_\vartheta(P^k_\gamma(E)) \subseteq P^k_\gamma(\mathcal{E})$. *Consequently, the containment relation* $\varphi_\vartheta(P^1_\gamma(E)) \subseteq P^1_\gamma(\mathbf{E})$ *is true.* □

Property 4.4 gives a direct consequence of generalization on the support of patterns in **E**, i.e., the generalization operator φ_ϑ is responsible for reducing the variability of event descriptors. By the same mechanism, the support of generalized events is augmented w.r.t. the support of events they subsume. Therefore, events that did not have minimum support γ in E could eventually have minimum support in $\mathcal{E}$ and **E**. In other words, events that have support η in E with $\eta < \gamma$ could be captured in $\mathcal{E}$ and **E**. Property 4.4 describes this phenomenon.

Property 4.4 (Existence of lower support patterns)
Given E *a collection of time sequences of events and* **E** *the summary of* E *by* $\chi_{\vartheta,w}$, *the following relation occurs:*

$$\begin{array}{l} \textit{if } \exists\ p \in P_\gamma(\mathbf{E}),\ \textit{such that } p \notin \varphi_\vartheta(P_\gamma(E)) \\ \textit{then } \exists\ \eta \geq 1 \textit{ with } \eta < \gamma,\ \textit{such that } \exists\ q \in P_\eta(E), \varphi_\vartheta(q) = p,\ \gamma > 1 \end{array}$$

Proof 15 (Proof of Property 4.4)
Trivially, the set of patterns $P_\eta(E)$ *with* $\eta = 1$ *contains all subsequences of sequences in* E. *If* $p \in P_\gamma(\mathbf{E})$ *and* $p \notin \varphi_\vartheta(P_\gamma(E))$ *then at least* $p \in \varphi_\vartheta(P_\eta(E))$. □

This is an interesting property since it could be used for discovering very low support sequential patterns in E that otherwise would have not been captured. We will give a more detailed usage scenario in our experiments reported in Section 6. However, one should note that it is not possible to analytically determine the exact support increase of each generalized event.

Despite the difficulty of establishing the exact relationship that links patterns discovered in raw time sequences to higher order patterns discovered in summarized time sequences, the structural information of summaries can still be useful for analyzing patterns. Indeed, an interesting question to answer is whether a given pattern p in $\varphi_\vartheta(P_\gamma(E))$ that is not supported by a summary $s^\star$ is actually supported by the underlying time sequence s, where $s^\star = \pi \circ \chi_{\vartheta,w}(s)$. In the case the underlying time sequence s did not support p, the summary is not responsible for any loss of patterns.

Therefore, given only the knowledge of a time sequence summary $s^\star$ and p a pattern of interest in $\varphi_\vartheta(P_\gamma(E))$, we show it is possible to decide from $s^\star$ for any raw sequence s:

1. if s could *possibly* have supported p or
2. if s *definitely does not* support p

We provide in Property 4.5 the minimal structural conditions on $s^\star$ to determine if s could have supported pattern p.

Property 4.5 (Conditions for possibly supporting a pattern)
Let $s^\star$ *be a summary intent, i.e.,* $s^\star = \pi \circ \chi_{\vartheta,w}(s)$ *and* $s^\star = \{e^\star_1, \ldots, e^\star_m\}$ *where* $e^\star_i = (x^\star_i, t^\star_i)$. *Let* p *be a sequential pattern of length* $r \geq 2$ *in* $\varphi_\vartheta(P_\gamma(E))$ *and* $p = \{e_1, \ldots, e_r\}$ *where* $e_j = (x_j, t_j)$. *The following conditions occur:*

1.1. *If* $\exists\ e \in p,\ \forall e^\star \in s^\star,\ x \neq x^\star$ *then sequence* s *does not support pattern* p.

1.2. *Otherwise,* $s^\star$ *supports individually each generalized event in* p. *The order in which these events are supported needs to be determined. Let us denote by* P *the collection of couples of contiguous events (as defined in Definition 3.9) in pattern* p, *i.e.,* $P = \{(e_j, e_{j+1}),\ 1 \leq j \leq r-1,\ \{e_j, e_{j+1}\} \sqsubseteq p\}$. *Then, one of the following conditions apply:*

2.1. *If* $\forall(e_j, e_{j+1}) \in P$, $\exists e^\star_u = (x^\star_u, t^\star_u) \in s^\star$ *and* $e^\star_v = (x^\star_v, t^\star_v) \in s^\star$ *with* $t^\star_u < t^\star_v$ *such that* $x^\star_u = x_j$ *and* $x^\star_v = x_{j+1}$, *then* s *possibly supports* p.

2.2. *Otherwise, it means that exists at least one couple* $(e_j, e_{j+1} \in P$ *does not satisfy condition 2.1.. In this case, since condition 1.2 is verified, i.e.,* $s^\star$ *support individually each event in* p, $\exists e^\star_p = (x^\star_p, t^\star_p) \in s^\star$ *and* $e^\star_q = (x^\star_q, t^\star_q) \in s^\star$ *such that* $x^\star_p = x_{j+1}$ *and* $x^\star_q = x_j$ *with* $t^\star_p < t^\star_q$. *In other words, events* e_j *and* e_{j+1} *are supported in summarized sequence* $s^\star$ *but in reverse order. There are two possible cases:*

2.2.1 *If* $d_T(t^\star_p, t^\star_q) \leq w$, *then it is possible that* s *supports pattern* p.

2.2.2 *Otherwise, i.e.,* $d_T(t^\star_p, t^\star_q) > w$, s *does not support pattern* p.

Let us explicit the conditions given in Property 4.5. Conditions **1.x** are straightforward and state that each individual event e in p needs to be supported by sequence $s^\star$ for s to possibly support pattern p. When condition **1.2** is met, condition **2.1** states that if all events e in pattern p appear in their exact order in sequence $s^\star$ then sequence s possibly supports pattern p. However, if some events in pattern p do not appear in their exact order in sequence $s^\star$, it does not necessarily mean that sequence s does not support pattern p. Indeed, due to the groupings of distant events on the timeline performed by operator ψ_w, there are conditions on the temporal locality parameter w to be taken into account before it can be decided whether sequence s supports pattern p or not.

Hence, Condition **2.2.** takes into account the structure of the summary $s^\star$. This condition supposes that there exist at least two contiguous events e_j and e_{j+1} in p that do not appear in their exact order in $s^\star$, i.e., "e_j followed by e_{j+1}". In other words, e_j and e_{j+1} are permutated in $s^\star$. In this case, it is still possible to determine whether sequence s possibly supports pattern p.

Under Condition **2.2.1**, we state that as long as e_j is within a temporal locality w from e_{j+1} in $s^\star$, i.e., $d_T(t^\star_p, t^\star_q) \leq w$, then it is still possible for sequence s to support pattern p. Indeed, when Condition **2.2.1** occurs, sequence $s^\star$ can be described as follows: $s^\star = \{\ldots, \underbrace{e^\star_p, \ldots, e^\star_q}_{d_T(t^\star_p, t^\star_q) = w' \leq w}, \ldots\}$. We denote by W the subsequence of contiguous events in $s^\star$ that are located within a distance w' of event $e^\star_p$, i.e., $W = \{e^\star_p, e^\star_{p+1}, \ldots, e^\star_q\}$. W is generated by a subsequence s_{sub} contained in the set $\{\{e_p, (\ldots e \ldots,)^* e_q\}\}$ with $e^\star_p = \pi \circ \psi_w(e_p)$ and $e^\star_q = \pi \circ \psi_w(e_q)$ (Definition 4.5), i.e., the collection of all combinations of generalized events that produce s_{sub} after operating ψ_w. Therefore, under condition **2.2.1**, W is possibly generated by a subsequence s_{sub} in which event e_{j+1} could occur after event e_j. For this reason, s possibly supports pattern p. Condition **2.2.2** states that s certainly does not support pattern p.

We established in this section some relationships that link sequential patterns discovered in raw time sequences to higher order patterns discovered in time sequence summaries. Not all relationships could be established since these relationships deeply depend on the summarization parameters, the mining parameters and the data themselves. All of these parameters greatly complexify the task, in particular establishing the relationships that link patterns discovered in raw time sequences to higher order patterns discovered in summarized sequences when the length k of patterns considered is greater than 2.

5 Exploratory Mining: A methodology to uncover specific knowledge from HOP

Previously, we characterized the relationships that link patterns discovered in raw time sequences to higher order patterns discovered in generalized or summarized time sequences, and those that link higher order patterns discovered in generalized sequences to higher

order patterns discovered in summarized sequences. In particular, we gave in Property 4.4 the intuition that mining higher order patterns having support γ in TSAR summaries could allow to capture patterns in raw sequences having support η lower than γ. This observation is interesting since it might not be possible to directly mine the original sequences with minimum support η. We rely on this property and present in this section a methodology that uses all characteristics already demonstrated to derive more specific patterns from higher order patterns discovered on TSAR summaries. This methodology is called *Exploratory mining*. Exploratory mining is achieved thanks to two complementary operations, namely, (i) *Pattern Events Specialization* and (ii) *Pattern Events Recombination*.

The ideas that guide these two complementary operations rely on observations made from the usage of TSAR's φ_ϑ and ψ_w operators. On the one hand, operator φ_ϑ rewrites event descriptors at a higher level of taxonomy. By doing so, the variability of event descriptors decreases while the support of each generalized event is increased. Mechanically, events that have very low support in a raw time sequence, e.g., due to the use of specific vocabulary, might have high support in its summarized counterpart, e.g., due to the association of specific vocabulary to more commonly used concepts. The idea of *Pattern Events Specialization* (PES) is then to identify in summarized sequences a pattern p of interest that have high support and try to find lower support patterns by *specializing* some event descriptors of events in pattern p using the same taxonomies input to TSAR.

On the other hand, operator ψ_w groups similar events that are within a certain temporal locality w on the timeline. As a direct consequence, summarized time sequences are shorter than their non-grouped counterpart and induce a loss of temporal accuracy. We showed in Section 4.2.3 that this operation induces a loss of *recall* capability, i.e., higher order patterns discovered on time sequence summaries can not capture all patterns discovered in raw sequences. Hence, the idea of *Pattern Events Recombination* (PER) is to mitigate the consequences of the ψ_w operator and produce candidate patterns from higher order patterns already discovered in the summaries. These candidate patterns are generated by recombining events that compose the higher order pattern. We detail these two operations in the following sections.

5.1 Pattern Events Specialization (PES)

The first mechanism that supports exploratory mining is called *Pattern Events Specialization* (PES). This operation is responsible for generating a set of candidate patterns from a higher order pattern mined in a collection of summarized time sequences. The idea of pattern event specialization is to select a pattern of interest, denoted $p = \{e_1, \ldots, e_r\}$ with $r \geq 1$, and a descriptive domain of interest denoted A. The domain of A is assumed to be organized into taxonomy H_A.

PES operates by parsing events $e = (x, t)$ in p and identifying descriptors d in x that are contained in taxonomy H_A. Each such descriptor is associated to a collection of descriptors denoted D. Descriptors in D are the direct specializations of descriptors d taken from taxonomy H_A. If a descriptor d does not appear in taxonomy H_A, it is not specialized and simply added to D. For instance, in our *topic* domain taxonomy, if descriptor d="Security" then D={"Anonymization", "Privacy"}. Therefore, each event $e = (x, t)$ in p generates a set of candidate events denoted C_e. C_e is obtained by producing all possible combinations of descriptors taken from sets of descriptors D, i.e., $C_e = \{(x', t),\ x' \in \{\{d_1\} \cup \ldots \cup \{d_m\}, d_1 \in D_1, \ldots, d_m \in D_m\}\}$ where D_k is the set

of specializations of descriptor d_k in x. The total set of candidate patterns generated by PES is denoted C_p and is the set of all possible combinations of events taken from the set of candidate events C_e for each event e in pattern p, i.e., $C_p = \{\{e_1, \ldots, e_r\},\ e_i \in C_e\}$.

Let us give a tangible example with a pattern having one single event. Suppose our pattern of interest is $p = \{e\}$ where $e = (x, t)$ with x={"Market data", "Security"} and we want to specialize this pattern on the *topic* descriptive domain. "Market data" does not belong to the *topic* descriptive domain but "Security" does. Therefore, set D_1 is associated to descriptor d_1="Security" of event e, where D_1={"Anonymization", "Privacy"}. The set of candidate events generated by PES is $C_e = \{e_1, e_2\}$ where $e_1 = (x_1, t)$ and $e_2 = (x_2, t)$, and x_1={"Market data", "Anonymization"} and x_2={"Market data", "Security"}. Therefore, the set of candidate patterns generated by PES on our pattern p is $C_p = \{p_1 = \{e_1\}, p_2 = \{e_2\}\}$.

The pseudo-code of the PES mechanism is given in Algorithm 7. Descriptors that appear in taxonomy H_A are identified and specialized between Line 5 and Line 11. The *specialize* procedure in Line 9 is responsible for retrieving from taxonomy H_A the set of concepts that specialize a given descriptor d. Candidate events are generated in Line 12 and candidate patterns are generated from these candidate events between Line 13 and Line 17.

Algorithm 7 Pattern event specialization pseudo-code

1. INPUTS:
   H_A: Taxonomy of attribute of interest
   $p = \{e_1, \ldots, e_r\}$: pattern
2. OUTPUT: C_p: Collection of candidate patterns
3. LOCALS:
   D: Collection of specialized descriptors for descriptor d in event e
   C_e: Collection of candidate events for event e
   C_p: Collection of candidate patterns

4. **for all** event $e = (x, t)$ in p **do**
5. **for all** descriptor d in x **do**
6. **if** $(d \notin H_A)$ **then**
7. $D \leftarrow \{d\}$
8. **else**
9. $D \leftarrow specialize(d)$
10. **end if**
11. **end for**
12. $C_e \leftarrow \{(x', t)\}$, where $x' \in \{\{d_1\} \cup \ldots \cup \{d_m\}, d_1 \in D_1, \ldots, d_m \in D_m\}\}$
13. **if** $(C_p = \emptyset)$ **then**
14. $C_p \leftarrow C_e$
15. **else**
16. $C_p \leftarrow p \cup \{e\},\ p \in C_p$ and $e \in C_e$
17. **end if**
18. **end for**
19. **return** C_p

The benefit of pattern event specialization for sequential pattern mining is to generate candidate patterns that are more precise from semantic view point. Therefore, these candidate patterns allow the discovery of more refined higher order patterns such that: (i) specialized patterns have lower support and (ii) descriptors used to describe events in

specialized patterns are more precise and specific. These two properties of pattern events specialization are most interesting for the following reasons: (i) since specialized patterns have lower support, it could have not been possible to discover such patterns using conventional algorithms (due to the explosion of computational time); (ii) if an analyst is not satisfied by the level of abstraction of given pattern, he can obtain other patterns by specialization without the need to mine the entire dataset again.

5.2 Pattern Events Recombination (PER)

The second mechanism that supports exploratory mining is called *Pattern Event Recombination* (PER). Properties demonstrated in Section 4.2 show that event rearrangements during summarization could be responsible for the loss of sequential patterns. These effects should somehow be mitigated. Hence, the idea of pattern events recombination is inspired from the application of edit distances to approximate string matching in spell checking applications. The assumption is that a higher order pattern p discovered in a summary could actually represent a larger number of patterns if a certain number of *edits* on pattern p are permitted. For instance, suppose there are two patterns p_1 and p_2 where $p_1 = \{e_1, e_2, e_3\}$ and $p_2 = \{e_1, e_3, e_2\}$. Patterns p_1 and p_2 are visually *similar* and in fact, p_2 is only different from pattern p_1 due to the permutation of events e_2 and e_3.

Therefore, PER operates by selecting a higher order pattern of interest, denoted $p = \{e_1, \ldots, e_r\}$ with $r \geq 2$. The events e that compose this pattern are then used to generate a set of candidate patterns C_p. Candidate patterns are obtained by recombining all events e involved in pattern p. This event recombination process could (i) generate all possible combinations of events in p or (i) be controlled by a maximum number of *edits*, e.g., 1, 2 or 3 permutations of events. In this case, the number of edits could be computed using well known edit distances such as Hamming's distance [Ham50] or Levenshtein's distance [Lev65].

The benefit of pattern events recombination for sequential pattern mining is to generate candidate patterns that might have been lost due to the shortening of time sequences by the ψ_w operator. Also, it allows to widen the exploratory space for searching for higher order patterns.

Exploratory mining is then the process of reiterating the two PES and PER mechanisms to continuously refine in a drill-down fashion higher order patterns discovered in TSAR summaries.

6 Experiments

In this section, we experiment Sequential Pattern Mining on time sequence summaries produced by TSAR. We have shown in our analysis work that there exists a fragile connection between patterns discovered in raw time sequences and higher order patterns discovered in summaries. This connection deeply depends on 3 dimensions: (i) The summarization parameters, (ii) the mining parameters and (iii) the distribution of the data. It is undeniable that mining summaries induces loss of patterns, possibly important losses. If one only focuses on the *recall* of patterns that should be discovered in raw time sequences. However, the purpose of Higher Order Mining is not necessarily to produce exactly the same knowledge that should be discovered in raw time sequences. For this reason, in the application of mining sequential patterns on time sequence summaries, we believe that it does not make sense to use conventional and widely accepted measures from Information

Retrieval, namely *precision* and *recall*, to evaluate the *quality* of patterns discovered. We would rather show in this experimental work that mining summaries allows to discover different knowledge that would otherwise have remained hidden.

From this perspective, it seems to us most interesting to evaluate how useful TSaR summaries are for the task of discovering very precise and specific knowledge. We propose to experiment SPM on TSaR summaries in two phases: (i) evaluate TSaR summaries as a support structure for SPM algorithms, i.e., their capability of aiding in reducing computational time, and (ii) operate exploratory mining on summaries produced by TSaR to see what kind of knowledge can be uncovered and how precise and specific it is. For this purpose, we reuse again the dataset extracted from Reuters's 2003 financial news archives as described in Section 6.1.2 in Chapter 2. However, since SPM algorithms are process-intensive tasks, we limit the dataset to one month of news. The dataset used still represents approximatively 100,000 financial news events distributed over 4600 time sequences where each sequence is associated to a company.

This section is organized as follows. We study in Section 6.1 how well TSaR summaries support SPM algorithms in the knowledge discovery task. We evaluate this support by observing the computational time of a traditional SPM algorithm, namely PrefixSpan [PHMA+01]. Then, in Section 6.2, we explore two scenarios for using TSaR summaries. We propose a first scenario for finding *trends* of events thanks to higher order patterns discovered. Then, we put into practice the exploratory mining methodology for refining and detailing the higher order patterns discovered with the objective of finding informative patterns having even lower support.

6.1 Mining financial time sequence summaries

We report here our experiments on mining TSaR summaries with a conventional SPM algorithm. We chose to implement the PrefixSpan [PHMA+01] algorithm to avoid the flaws of Apriori-based algorithms, i.e., the generation of too many candidate patterns. Also, we chose PrefixSpan for its high performances. Note that even though PrefixSpan is relatively old, its high performances still make it a premium choice as a base technique for some state of the art approaches such as BIDE [WH04].

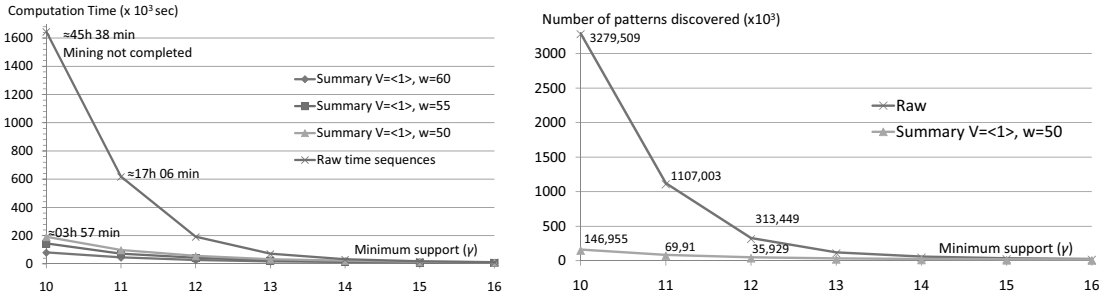
We generate TSaR summaries using the following parameters: (i) $\vartheta = \langle 1 \rangle$ and (ii) $w \in \{1, 2, 3, 4, 5, \ldots, 50, 55, 60\}$. The overhead induced by summarizing the input time sequences of news data with these parameters is almost constant and takes approximatively 35 seconds. In comparison to the mining computational time presented in Figure 4.2(a), this overhead is negligible. This observation confirms our first intuition, i.e., summarization is a costless preprocessing step in comparison to the actual SPM task.

Figure 4.2(a) gives the computational time of PrefixSpan at different minimum support levels: $\gamma \in [10 \ldots 20]$ which represents 0.01% to 0.02% of input time sequences. These minimum support values are in fact very low values. For more readability, we only report in Figure 4.2(a) the computational time obtained when mining summaries built with $w \in \{50, 55, 60\}$. As the minimum support γ decreases, we can observe in Figure 4.2(a) the exponential increase of the computational time when mining raw time sequences. Paradoxically, mining sequential patterns on financial news requires the use of very low support values γ in order to obtain non trivial results, i.e., patterns with a least a minimum of 2 to 4 events.

For instance, at the lowest value of minimum support, i.e., when $\gamma = 10$ (i.e. $\gamma = 0.01\%$), the mining task could not complete even after approximately 45 hours of runtime. These performances are very similar to those obtained by Masseglia et al.'s [MTT04] and

Raju et al.'s [RKS07] who operated SPM on Web Usage logs. Masseglia et al. and Raju et al. also showed that conventional SPM algorithms hit a performance wall with minimum support levels as low as $\approx 0.06\%$ to $\approx 0.02\%$. These figures empirically show that performing SPM on raw time sequences of financial news events is a very costly task. This conforts our choices (i) to mine TSAR summaries to discover knowledge and (ii) to limit the size of our dataset to one month of news.

When mining TSAR summaries, the combination of (i) few event rearrangements, i.e., small temporal locality windows w, (ii) the increased support of each event due to generalization and (iii) using low minimum support, i.e., $w \leq 5$, $\vartheta = \langle 1 \rangle$ and $\gamma < 15$, respectively, makes the computational time explode exponentially as well. Note that this phenomenon occurs slower than on raw time sequences. However, interestingly, when TSAR summaries are compact enough, e.g., built with temporal locality window $w \geq 25$ or $w \geq 50$, we observe that mining higher order patterns having minimum support as low as $\gamma = 10$ is possible and can be achieved faster than on raw time sequences, e.g., at least one order of magnitude faster.



(a) SPM processing time

(b) Number of sequential patterns discovered

Figure 4.2: SPM performances

6.2 TSAR summary usage scenarios

Our preliminary results show that extracting knowledge from time sequences of financial news could use summaries as a support structure, at least to reduce computational time. However, these experiments do not explicitly inform the analyst on the nature, quality or content of the knowledge extracted. For this reason, we present in this section two scenarios to illustrate how to use TSAR summaries for sequential pattern mining applications.

6.2.1 Scenario 1 (Sc.1): Discover higher order knowledge

The purpose of the first scenario is to discover higher order patterns in the sequences of financial news events. These patterns are *trends* that inform the analyst on the general topic of most frequent patterns. In other words, higher order patterns discovered give the *profile* of the most recurring series of events. Indeed, we have shown in Section 4.2 that, due to the loss of temporal information when rearranging events, it is very difficult to establish the exact relationship that links sequential patterns discovered on raw time sequences to higher order patterns discovered on time sequence summaries.

Therefore, the purpose of the trends discovered is to inform the analyst on the main topics of frequently occurring events. This is why sequential patterns discovered on very compact summaries, i.e., built with strong generalization and large temporal locality windows w, still give valuable information on the underlying data. If the analyst requires more

fine grain trends, more refined TSAR summaries, i.e., built with lower generalization levels and smaller temporal locality windows, could be used by the SPM algorithm.

6.2.2 Scenario 2 (Sc.2): Discover specific patterns from high order patterns

The second scenario is complementary to scenario 1. In the case the analyst has found a (set of) higher order pattern(s) of interest, he should be given the opportunity to refine this knowledge without the need to operate SPM again. Hence, this scenario allows the analyst to use knowledge already discovered to uncover more specific patterns in a drill-down fashion using the exploratory mining methodology. We select a higher order pattern of interest then specialize its event descriptors and/or recombine events to generate a set of candidate patterns. We compute the support of these candidate patterns and show that it is possible to find specific patterns that have very low support, otherwise difficult to discover.

6.3 Usage scenarios in practice

We build TSAR summaries for the input dataset with summarization parameters $\vartheta = \langle 1 \rangle$ and $w = 50$. The output summary $\chi_{\vartheta=\langle 1 \rangle, w=50}(E)$ is denoted $\mathbf{E}$. We mine in $\mathbf{E}$ higher order patterns having minimum support $\gamma = 14$ and limit the maximum length of patterns to 7. This constraint is added to conveniently reduce the mining computational time and the total number of patterns output. The set of sequential patterns discovered is denoted $P_\gamma(\mathbf{E})$. We give in Table 4.4 some sequential patterns of interest discovered in $\mathbf{E}$ and their respective support. The set of descriptors used to describe each event in Table 4.4 is given in Table 4.5.

Pattern	$supp_{\mathbf{E}}$
p_1=$\{e_1\}$	132
p_2=$\{e_2\}$	121
p_3=$\{e_3\}$	273
p_4=$\{e_4\}$	83
$p_{1,2}$=$\{e_1, e_2\}$	33
$p_{1,3,4}$=$\{e_1, e_3, e_4\}$	24
$p_{3,4,1}$=$\{e_3, e_4, e_1\}$	21

Table 4.4: Patterns discovered and support

Event	Event descriptors
e_1	{any_operation, financial institution, west}
e_2	{any_operation, sales, west}
e_3	{any_operation, west}
e_4	{any_operation, business, financial institution, west}
$e_{\downarrow 1,1}$	{any_operation, islf, west}
$e_{\downarrow 1,2}$	{any_operation, ins, west}
$e_{\downarrow 1,3}$	{any_operation, fin, west}
$e_{\downarrow 1,4}$	{any_operation, bnk, west}
$e_{\downarrow 2,1}$	{any_operation, bus, west}
$e_{\downarrow 2,2}$	{any_operation, ret, west}
$e_{\downarrow 2,3}$	{any_operation, who, west}

Table 4.5: Description of events

6.3.1 Sc.1: High order trends

In usage scenario Sc.1, we highlight the possibility of discovering general knowledge in the form of trends from compact TSAR summaries. For instance, let us consider pattern p_3 in Table 4.4. From the event's description given in Table 4.5, news that involve *"some kind of operation related to western countries"* seem to be very frequent ($supp_{\mathbf{E}}(p_3) = 273$). More specifically, pattern $p_{1,3,4}$ has support 24 in $\mathbf{E}$ and indicates that series of news that involve *"some financial institution"* or *"business"* are also frequent. These two patterns

by themselves provide comprehensive and informative enough information on the trend of some frequent series of news. Here, the trend of the news is *"business/financial operation in western countries"*. If the analyst requires more insight into the patterns and news involved, this higher order pattern discovered in the summaries could be refined thanks to the exploratory mining methodology. This is the purpose of scenario 2. We propose to use and explore pattern $p_{1,3,4}$ in more details.

6.3.2 Sc.2: Exploratory mining

Sc 2.1: Pattern events recombination

In scenario Sc.2, we try to discover more refined and more detailed knowledge from trends discovered in Sc.1 by means of exploratory mining. We start by operating pattern events recombination on higher order patterns discovered earlier. Let us consider higher order pattern $p_{1,3,4}$ in Table 4.4. The support of $p_{1,3,4}$ in $\mathbf{E}$ is 24. If fact, when mining sequential patterns in $\mathbf{E}$ with minimum support $\gamma = 24$, $p_{1,3,4}$ is the only higher order pattern that involves all three events e_1, e_3 and e_4. In other words, patterns $\{e_1, e_4, e_3\}$, $\{e_3, e_1, e_4\}$, $\{e_3, e_4, e_1\}$, $\{e_4, e_1, e_3\}$ and $\{e_4, e_3, e_1\}$ should have support lower than 24.

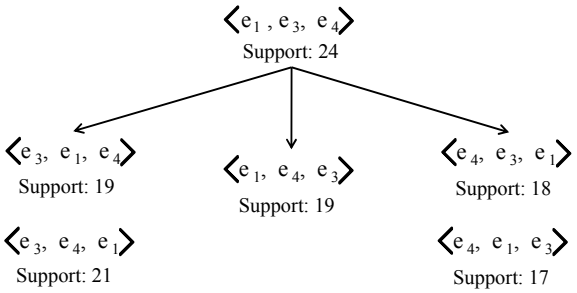


Figure 4.3: Sequences of combined events

Therefore, we start by recombining events in pattern $p_{1,3,4}$ and we generate all possible combinations of events in $p_{1,3,4}$. The resulting set of candidate recombined sequences and their respective support in $\mathbf{E}$ is given in Figure 4.3. Note that the support of all candidate recombined sequences is lower than the support of $p_{1,3,4}$. In average, each combined sequence has a support lower than that of pattern $p_{1,3,4}$ by approximately 21%, while the candidate pattern that has the highest support is $p_{3,4,1}$ with support equal to 21, i.e., 12.5% lower than $p_{1,3,4}$. This observation shows that we were capable of discovering other patterns that (i) contain all three events e_1, e_2 and e_3, and that (ii) have lower support than $p_{1,3,4}$ without the need to completely reoperate SPM on the entire dataset.

Also, let us consider pattern $p_{3,4,1}$ that has support 21 in $\mathbf{E}$, built with temporal locality window $w = 50$. When we compute the support of $p_{3,4,1}$ in all summaries built with a temporal locality window w that ranges from $w = 1$ to $w = 50$, the support of $p_{3,4,1}$ goes below 24 as soon as $w \geq 25$. This observation empirically confirms our statement that event rearrangements during summarization can be responsible of the loss of sequential patterns.

These observations allows us to confirm that pattern event recombination is an interesting tool to discover other higher order patterns. Eventually, these other higher order patterns have lower support. In our experiments, suppose the minimum support γ used for mining is low enough, e.g., $\gamma \approx 10$, then pattern event recombination could allow the discovery of patterns with support lower than 10. These patterns would have remained

undiscovered due to (i) the very long mining computation time when performing SPM with $\gamma < 10$ or due to (ii) the information overload inherent to the exponentially large result sets.

Sc 2.2: Pattern events specialization

The second mechanism involved in exploratory mining is pattern events specialization. We illustrate this second mechanism using a pattern selected from the knowledge extracted earlier. Suppose we are interested in patterns that contain the descriptors *"financial sector"* and *"sales"* taken from the *"Industrial sector"* domain. Table 4.6(a) and Table 4.6(b) give in details the specialization of descriptors *"financial sector"* and *"sales"*, respectively.

In practice, without any prior knowledge, higher order pattern $p_{1,2} = \{e_1, e_2\}$ is a series of two news events that can be understood as follows: *"some kind of operation in the* financial sector *occured in a western country"* followed by *"some kind of operation in the* sales *industry occured in a western country"*. These two news events carry some very general information on the location and the nature of the industry concerned.

(a) Specialization of descriptor "financial sector"

Item	Meaning	Concerns
islf	Islamic finance	News relating to Islamic finance and banking; Stories about banks run entirely according to Koranic precepts; Stories about the development of Islamic bond markets as well as the individual issuance of Islamic bonds; Development of Islamic banking regulations and other guidelines similarly.
ins	Insurance	Life and health insurance; property and casualty insurance; reinsurance; regulation or regulatory bodies associated with the insurance industry; Medicare and Medicaid and insurance brokers.
fin	Financial and Business Services	Agents or brokers of financing and financial services; investment trusts; offshore funds; investment banks; brokers; merchant banks; travel-related money services; personal, consumer and educational credit; lease financing and regulation or regulatory bodies associated with the financial services industry.
bnk	Banking	Private banking; medium-term financing; all depository and credit institutions e.g. commercial banks, clearing banks, savings and loans, chartered banks, universal banks; futures and options exchanges and regulation or regulatory bodies associated with the banking industry.

(b) Specialization of the "sales" descriptor

Item	Meaning	Concerns
bus	Business, Public services	Services to business and consumers including office supplies; advertising/marketing; software development, data vendors & data processing; security; transporters, custom agents, package and mail delivery; agencies; water distribution; airport, port, tunnel, highway management; port-harbour transport and warehousing; waste management, cleaning, water filtration
ret	Retail	All retailing including retail sales and consumer price indexes
who	Wholesale	Wholesaling and distribution, including wholesale price indexes

Table 4.6: Descriptor specializations and meanings

This knowledge can be refined thanks to the specialization of descriptors *"sales"* and *"financial sector"*. Therefore, pattern $p_{1,2}$ is specialized into 12 candidate specialized patterns. The support of each candidate specialized pattern in $\chi_{\vartheta',w}(E)$ is computed and

results are gathered in Table 4.7. In this example, the support of $p_{1,2}$ in $\mathbf{E}$ is 33 and all candidate specialized patterns obtained from $p_{1,2}$ have lower support in $\chi_{\vartheta',w}(E)$.

Note that ϑ'=$(A_1, \ldots, A_{12})$ and $\vartheta[A_i] = 1$ if $A_i \neq$"Industrial sector", $\vartheta[A_i] = 0$ otherwise. In other words, $\chi_{\vartheta',w}(E)$ is the summary obtained when descriptors from descriptive domain "Industrial sector" are generalized one time less; Since $\vartheta = \langle 1 \rangle$, i.e. all descriptors are generalized exactly once, ϑ' is obtained when descriptors from descriptive domain "Industrial sector" are not generalized at all. Thus, the pattern events specialization mechanism requires the building of new summaries. However, we already showed that this task has negligible computational time.

Now, let us consider candidate pattern $p_{(1,1;2,3)} = \{e_{\downarrow 1,1}, e_{\downarrow 2,3}\}$. In practice, this candidate pattern represents a series of two news that can be understood as follows: *"some kind of operation in* Islamic finance and banking *occurred in a western country"* followed by *"some kind of* wholesale *operation occurred in a western country"*.

Specialized patterns	$supp_{\chi_{\vartheta',w}(E)}$
$p_{(1,1;2,1)}$=$\{e_{\downarrow 1,1}, e_{\downarrow 2,1}\}$	3
$p_{(1,1;2,2)}$=$\{e_{\downarrow 1,1}, e_{\downarrow 2,2}\}$	1
$p_{(1,1;2,3)}$=$\{e_{\downarrow 1,1}, e_{\downarrow 2,3}\}$	4
$p_{(1,2;2,1)}$=$\{e_{\downarrow 1,2}, e_{\downarrow 2,1}\}$	8
$p_{(1,2;2,2)}$=$\{e_{\downarrow 1,2}, e_{\downarrow 2,2}\}$	5
$p_{(1,2;2,3)}$=$\{e_{\downarrow 1,2}, e_{\downarrow 2,3}\}$	15
$p_{(1,3;2,1)}$=$\{e_{\downarrow 1,3}, e_{\downarrow 2,1}\}$	3
$p_{(1,3;2,2)}$=$\{e_{\downarrow 1,3}, e_{\downarrow 2,2}\}$	0
$p_{(1,3;2,3)}$=$\{e_{\downarrow 1,3}, e_{\downarrow 2,3}\}$	9
$p_{(1,4;2,1)}$=$\{e_{\downarrow 1,4}, e_{\downarrow 2,1}\}$	0
$p_{(1,4;2,2)}$=$\{e_{\downarrow 1,4}, e_{\downarrow 2,2}\}$	0
$p_{(1,4;2,3)}$=$\{e_{\downarrow 1,4}, e_{\downarrow 2,3}\}$	0

Table 4.7: $p_{1,2}$ specialized sequences & support

Interestingly, the support of $p_{(1,1;2,3)}$ in $\chi_{\vartheta',w}(E)$ is only 4. When considering in Figure 4.2(a) that mining computational time explodes exponentially with the decrease of the minimum support γ, we can safely claim that pattern $p_{(1,1;2,3)}$ is impossible to discover in normal conditions. Furthermore, Figure 4.2(b) shows that the number of sequential patterns discovered also explodes exponentially with the decrease of the minimum support γ. Hence, even if we suppose mining sequential patterns with $\gamma = 4$ was possible in an acceptable computational time, pattern $p_{(1,1;2,3)}$ would still have been lost in the mass of sequential patterns discovered.

Now, let us consider pattern $p_{(1,2;2,3)} = \{e_{\downarrow 1,2}, e_{\downarrow 2,3}\}$. Table 4.7 shows that this pattern has support $supp_{\chi_{\vartheta',w}(E)}(p_{(1,2;2,3)}) = 15$. This sequential pattern can be understood as follows: *"some kind of operation occurred in the insurance domain in a western country"* followed by *"some kind of* wholesale *operation occurred in a western country"*. This series of events has occurred 15 times in the entire dataset. If the granularity of descriptors that describe events in this higher order pattern does not satisfy the analyst, he can reiterate the exploratory mining process and detail the pattern by specializing other descriptors, e.g., *"west"*, or by recombining events in $p_{(1,2;2,3)}$.

We have provided in this section a thorough experimental study on mining TSAR summaries with a conventional SPM algorithm. We have shown that for certain applications,

such as financial applications, mining the raw data by means of a conventional SPM algorithm is not sufficient. Interesting patterns, i.e., patterns that involve at least 2 to 4 events, are only extracted for very low support levels, i.e., $\gamma \leq 0.06\%$. Unfortunately, at such low levels of support, the mining computational time simply explodes and the user is overwhelmed with the mass of patterns extracted. On the other hand, TSAR summaries have appeared as interesting candidates for supporting SPM algorithms. Experiments show that it is possible to mine summaries at very low support levels, i.e., $\gamma \approx 0.01\%$, but this operation requires the summaries to be compact, i.e., built with large temporal locality windows w. However, we provided the two usage scenarios and demonstrated that applying the exploratory mining methodology on higher order patterns discovered in very compact summaries could allow the discovery of very precise knowledge.

7 TSAR as a support service in the ADAGE project

Since the late 90's, Reuters has been generating and storing a massive amount of financial information, e.g., market quotes and financial news articles. A very natural application and need that has risen is to correlate both sources of data. For instance, suppose a trader receives a news flash on his information system. A most desirable application is to estimate if this piece of news will have an impact on market shares and when that might occur. This additional information that is most strategic for analysts to be able to anticipate market movements to consolidate their customer portfolios.

Time series analysis to estimate stock prices' very short term variations is a well studied area [Ham94,Hei99,SZ04,Bri08]. However, evaluating the impact of a piece of news on the evolution of stock prices is a different and (maybe more) difficult problem. Since financial news contains both unstructured, e.g., free text, and structured data, e.g., attribute-value pairs, conventional techniques used for analyzing time series can not be leveraged. In fact, a whole Knowledge Discovery –from Databases– (KDD) process should be implemented to uncover patterns from news data and then correlate patterns discovered to stock data. This scenario and the specificities of the input data considered has motivated our work to build a framework dedicated to analyzing time sequences of events where events have complex structure. Therefore, we present in Section 7.1 the STEAD framework that proposes a set of tools for discovering knowledge from textual contents. We then present in Section 7.2 the more general project called ADAGE in which STEAD is implemented.

7.1 The STEAD framework

We present hereafter the *S*ervice-based *TE*mpor*A*l *D*ata (STEAD) analysis framework for discovering knowledge from textual contents. The STEAD framework was designed to provide analysts with a comprehensive tool that allows them to supervise and interactively refine sequential pattern mining on complex data. This tool was demonstrated in [PSPB+08]. An overview of the STEAD framework is illustrated in Figure 4.4. The framework is organized into five services for processing textual datasets and an additional service intended for analysts to define and express their understanding of domains of interest in the form of taxonomies. The five base services are (i) Data Selection, (ii) Data preprocessing, (iii) Data transformation, (iv) Data mining and (v) Results evaluation. These services correspond to the five classical steps identified for Knowledge Discovery from Databases (KDD) proposed by Fayyad et al. in [FPSS96].

Through a User Interface (UI), STEAD allows analysts to specify parameters for setting the different services and configuring the analysis task. The *Domain Knowledge Acquisition* service assists analysts in the definition of taxonomies that allow them to express their interests and understanding of specific domains regarding the data. Taxonomies are

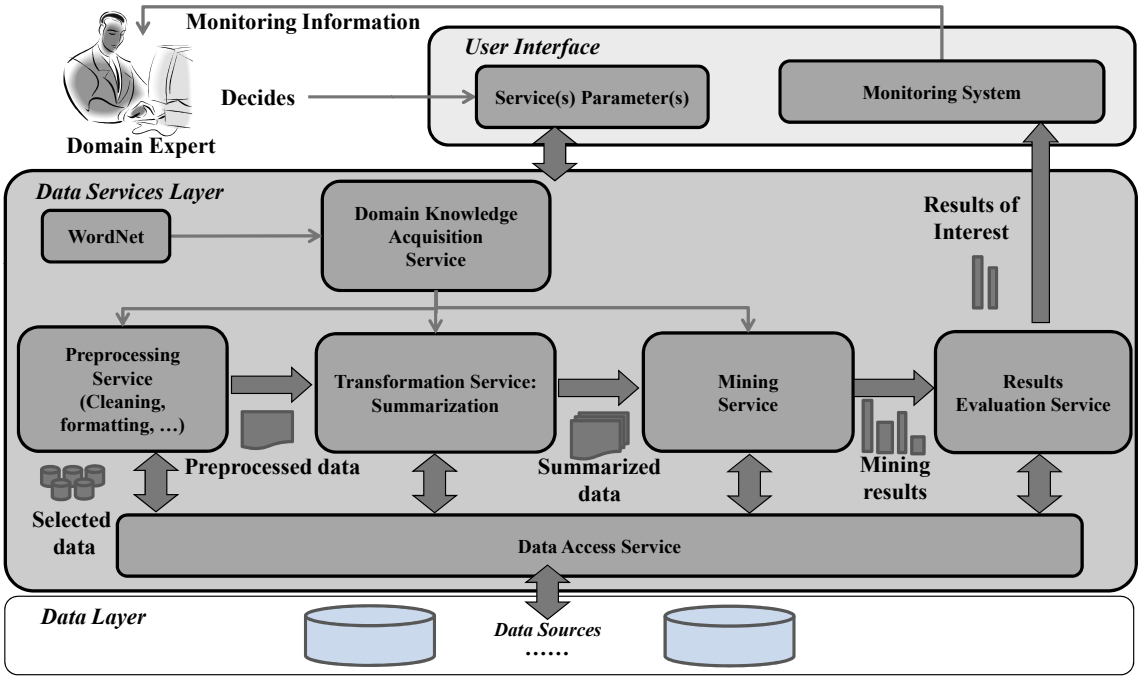


Figure 4.4: STEAD analysis framework

then used for controlling other steps of the services, e.g., data cleaning, preprocessing and TSAR summarization. Knowledge extracted from the input data is then presented to the analyst through the UI. Analysts may refine the taxonomies and/or summarization parameters and/or mining parameters to further investigate certain aspects of the data or certain aspects of the higher order patterns discovered. We detail the role of each service in the following paragraphs.

7.1.1 Domain knowledge acquisition service

The analyst expresses the way raw data should be preprocessed and the way TSAR should abstract data descriptors through sets of taxonomies over the features he is interested in. For instance, in previous chapters, we chose to preprocess Reuters's financial news archives and extract descriptors from 15 domains that include: Location, Commodities, Industrial sector, etc.. For this purpose, we generate background knowledge for these domains in the form of taxonomies. This background knowledge (i) can already exist, e.g., in the form of domain specific ontologies such as WordNet, or (ii) needs to be defined. When undefined, we propose to manually generate this knowledge thanks to tools such as Protégé [fBIRatSUSoM] or to automatically generate it using mechanisms as presented in Section 6.1.2 in Chapter 2. This understanding of the data then serves for all subsequent preprocessing and transformation tasks.

7.1.2 Data access service

The analyst selects through this service the data source on which he wishes to operate his analysis. This services acts as a *Data Access Service* from which the analyst can retrieve as well the raw input data or any other processed form of the data. This flexibility in the retrieval of the data is the key to an interactive approach for operating KDD. It allows analysts to examine data in any form, i.e., in a primitive form, as summaries or as a data mining result set.

7.1.3 Data preprocessing service: cleaning and filtering

Conventional sequential pattern mining algorithms require the input data to be in the format of a customer transactions database as described in Chapter 1. In order to analyze textual content, the preprocessing service takes as input (i) a collection of raw structured or unstructured data, (ii) a collection of taxonomies for the descriptive domains of interest, (iii) parameters that identify how to organize the data into time sequences and that determine the nature of the timestamps, i.e., actual dates or integer values, for timestamping events on the timeline. This service outputs a collection of time sequences. However, this data preprocessing task is not trivial. When we consider xfinancial news data generated by Reuters, it contains both structured (i.e. single- or multi-valued categorical descriptors) and unstructured information (i.e., free text):

- Structured information: For instance, in the context of financial news, descriptors are provided as *Topic_ codes* or *Company_ ID* by Reuters' journalists at the time a news is written. For example: {Timestamp: 01 Jan 2004}, {Company_ID: AAH.AS ABN.N MER.N}, {Topic_codes: EUROPE USA Bank INS NL}

- Free text: In financial news, free text appears both in the news header and in its body. For example "Dutch bank ABN AMRO said on Wednesday it had reached a preliminary agreement to sell its U.S.-based Professional Brokerage business to Merrill Lynch & Co...".

Structured information is well handled by SPM algorithms. However, free text needs to be processed and a set of descriptors extracted to precisely describe the content of the news article. As discussed earlier in Chapter 2, advanced techniques from Natural Language Processing can be leverage, e.g., OpenCalais [Reu], or more simple, terms in the free text that match concepts defined in input taxonomies could be extracted to describe the content of the news. Whatever the technique chosen, an interesting work perspective is to assess the quality of descriptors w.r.t. the quality of summaries produced and the quality of the knowledge extracted.

7.1.4 Data transformation service

The purpose of this service is to provide an additional preprocessing step responsible for building support structures, if needed, for the data mining task the analyst wants to carry out. For the purpose of mining sequential patterns, we propose to implement a time sequence summarization algorithm. We chose TSaR for its capability (i) of reducing the variability of descriptive domains, (ii) of reducing the numerosity events in time sequences and (iii) summaries produced are time sequences of events that can be directly exploited by the underlying data mining algorithms. Also, the analyst has total control over the knowledge discovered since we have analytically identified and determined the characteristics of higher order patterns that could be extracted from TSaR summaries. Hence, this service implements the TSaR summarization algorithm as it is presented in Chapter 2.

7.1.5 Data mining service

The data mining service is the core of the framework for knowledge discovery. The framework was designed to allow any kind of mining but most particularly we focused on sequential pattern mining for discovering knowledge in Reuters's financial news archives in the form of higher order patterns. This service takes as input a collection of customer transactions and data mining parameters, e.g., the minimum support and the maximum length

of frequent patterns to discover. The service outputs a collection of frequent sequential patterns.

Initially, the AprioriAll [AS95] algorithm was implemented for mining sequential patterns but subsequent experiments failed to operate on Reuters's financial news dataset. There are two reasons for this failure: (i) mining financial news requires the minimum support to be very low, i.e., $\gamma \leq 0.06\%$, for patterns to be of interest, and (ii) the candidate generation phase of Apriori-based algorithms induce a combinatory explosion at very low levels of minimum support. Therefore, we chose to mitigate these issues by implementing the more efficient prefix-projected pattern growth algorithm called PrefixSpan [PHMA+01]. This service can account for more advanced approaches in the literature, especially all the work done in Web Usage Mining [MTT04,RKS07] where neural network based clustering methods are used to partition the input dataset or techniques that only mine closed sequential patterns such as BIDE [WH04].

7.1.6 Knowledge evaluation and filtering service

When performing intensive mining, an issue is to assess the mass of knowledge discovered and present to the analyst the information that might be of interest. Indeed, most existing data mining techniques focus on the efficient computation of result sets and give little interest to its *quality* or *usability*. The purpose of this service is to provide a workspace and mechanisms for taking into account analysts' needs, experience and feedback to post-process and filter the result sets. This filtering activity can be done using some metrics [GH06], text filtering, post-mining [Spi99b], etc.. Previous work such as Xin et al.'s [XSMH06] that focuses on the users feedback for discovering interesting patterns is a very good candidate for extracting golden nuggets from large result sets. In our implementation of the service, we adopted an approach that allows the analyst to filter and select higher order patterns of interest using keyword-based searches. Higher order patterns selected this way can then be used to discover more refined patterns using the exploratory mining methodology introduced in Section 5.

The STEAD analysis framework presented in this section is an independent KDD system for analysts to discover knowledge from Reuters's financial news archives. As it is presented here, STEAD does not specify how Reuters's data sources are accessed, how data from Reuters is modeled and transformed, etc.. The reason is that different components of the STEAD framework such as the *Data access service* are actually integrated in a more generic system called the **A**d-Hoc **DA**ta **G**rids **E**nvironments (ADAGE) project. We give an overview of this project in the following section.

7.2 STEAD as part of the ADAGE project

7.2.1 Overview

The Efficient Management of Information Resources Over **A**d-Hoc **DA**ta **G**rids **E**nvironments (ADAGE) project [Srg,RGY09] is a project funded by the DEST-ISL Competitive Grants Program (Round 10). The project will also be closely lined to the newly launched European Union funded project (called SORMA) aimed at developing methods, tools and software for efficiently managing grid computing resources.

The ADAGE project is primarily concerned with *data grids*. A "data grid" is a new term used to refer to huge repositories of data which are distributed across several heterogeneous platforms and systems. In particular, the ADAGE project deals with ad-hoc data grids which commonly refer to unstructured data, i.e., data that is not formatted according

to any standards, or *dirty* structured data, i.e., data that still needs cleaning due to errors introduced during data generation. Ad-hoc data is becoming increasingly important in all walks of life. It does not only include data generated by humans but also includes data generated by machines. For instance, many sensors generate incomplete data, some market data feeds also generate data with missing information or gaps and documents written by human operators might contain misspelled terms.

The purpose of the ADAGE project is to investigate how to efficiently gather, store, retrieve and process ad-hoc data grids from both a manager and an end-user perspective. These techniques can benefit various applications in e-science and e-research including: Flexible sharing of data among researchers, analysis of information for security purposes, analysis of sensed data, etc.. Supporting these applications involve researching and applying a wide spectrum of Information and Communication Technologies (ICT) in areas as diverse as databases, query-processing, data mining, visualization and grid computing. For instance, one target application is to understand existing correlations between several data grids, i.e., market data generated and announcements and financial news data grids from Reuters.

7.2.2 STEAD's place in ADAGE

The ADAGE project is a user-driven Service-Oriented Architecture (SOA) enabling a separation of concerns regarding large news data provision, processing and visualization. In each of these three categories of services, re-usable and interoperable software components are defined as Web Services to manipulate entities of an underlying event-based data model. ADAGE provides a uniformized framework for representing data from heterogenous sources and relies on the Common Base Event (CBE) Model [OLS+02] promoted by IBM. This data model was proposed to facilitate the analysis of message-like information introduced by networked computer systems and captured in log files.

Therefore, the CBE data model implemented in ADAGE allows to manage data generated by diverse data grids, e.g., numerical values such as stock values or textual data such as financial news articles, under a same data representation. Hence, all data objects generated by the data grids are produced as sequences of events. In this conceptual model, financial electronic markets (e-markets) [Her03] can be understood as distributed event systems producing different types of events such as market events or news events. Market events capture data attributes such as bid/ask, types of products being traded, volume/number of products etc.. News events capture data related to particular news stories being published by news organizations such as Reuters.

From this perspective, the STEAD analysis framework naturally appears as a subsystem of the more general ADAGE architecture. The STEAD analysis framework provides the tools and services necessary for selecting, summarizing, mining and/or visualizing large amounts of primitive news events, events summarized into higher level entities or high order patterns extracted from these entities. Generating and using such higher level entities reduces the number of news entities to manipulate and this makes a range of data mining techniques feasible on larger datasets. Visualization becomes also feasible on very large amount of data as a group of news is represented as one single generalized news item.

8 Related work

8.1 Mining patterns at different levels of representation

The applicative study presented in this chapter relates most to research focused on discovering generalized sequential patterns as introduced by Srikant and Agrawal with GSP [SA96]. Indeed, we mentioned earlier in this chapter that Agrawal and Srikant's extension for sequential pattern mining could be understood as a form of higher order mining: (i) taxonomies generalize the level of representation of the data, (ii) sliding windows relax the notion of support of a pattern and (iii) time constraints allow to limit the scope of the mining operation. These add-ons address various limitations and the rigidity of the original definition of SPM and allow to discover more general knowledge or knowledge at different levels of abstraction.

This extension to the SPM paradigm has attracted much interest and new methods have been proposed to improve GSP's Apriori-based performances [LLSYW02,RCYBY04, MPT04,HY06]. In particular, some methods focus on the temporal aspect of GSPM, i.e., the sliding window relation, the minimum and maximum gap parameters:

- Sliding window w: This notion relaxes the definition of when a time sequence s contributes to the support of a candidate pattern p. This relaxation allows a collection of events within a range w, i.e., $e_i, \ldots, e_{i+w}$, to support each event e_i in p. In other words, suppose event $e = (x, t)$ in pattern p is not supported by any event in s. However, there exists two events e_i and e_{i+1} within a neighborhood w of each other, e.g., $t_{i+1} - t_i \leq w$, such that $x \subseteq x_i \cup x_{i+1}$. In this case, the sliding window relaxation considers that event e is supported in s, by the combined events e_i and e_{i+1}.

- Minimum gap: min_{gap} represents the minimum time gap in a time sequence between two events that could contribute to the support of a candidate pattern p. This temporal constraint limits the number of sequences that could support a pattern and, ultimately, reduces the overall number of patterns that can be discovered.

- Maximum gap: max_{gap} represents the maximum time gap in a time sequence between two events that could contribute to the support of a candidate pattern p. This temporal constraint limits the number of sequences that could support a pattern and, ultimately, reduces the overall number of patterns that can be discovered.

In a nutshell, these additional temporal parameters allow GSPM algorithms to discover knowledge at different levels of temporal granularity and even patterns that otherwise would have remained hidden. The temporal relaxation provided by sliding windows allow to uncover knowledge that can not be found due to the rigid definition of events in traditional SPM. Methods as DELISP [LLSYW02], EPSpan [RCYBY04] and Hirate et al.'s method [HY06] are PrefixSpan-based [PHMA+01] techniques for efficiently mining generalized sequential patterns. These approaches mainly focus on (i) the temporal relaxation and (ii) improving the performances of Apriori-based approaches. In [MPT04], Masseglia et al. propose GTC that takes up the principles of GSP and PSP [MC98]. GTC contrasts with GSP by handling the temporal constraints in the earlier stage of the algorithm for more efficiency.

However, these approaches contrast with our study by the fact they are specifically developed to efficiently address the temporal aspects of GSPM. In comparison, our study gives more interest to the semantic aspect of GSPM and proposes to apply off-the-shelf SPM algorithms, e.g., PrefixSpan, to discover higher order knowledge from TSaR summaries.

When one considers the semantic relaxation introduced by GSPM, one can notice that this relaxation has an important impact on mining computation performances. Indeed, semantic relaxation is achieved by extending an itemset by means of adding each item's antecedents taken from input taxonomies into the itemset. Under the traditional definition of a subsequence (Definition 4.1), the computational time of conventional GSPM algorithms, in particular Apriori-based algorithms, will explode. This effect directly results from adding more abstract terms into itemsets. Mechanically, the variability of itemsets is reduced and their support augmented: The number of candidate generated by Apriori-based algorithms at each iteration will increase exponentially. Assuming the mining task could be achieved in acceptable times, the benefit of semantic relaxation is it capability of capturing all patterns that could be discovered on TSaR summaries.

Also, in this work we assume that event descriptors in input sequences form a *single-item* itemset, i.e., an itemset is understood as an atomic unit. Thus, we limit the definition of Sequential Pattern Mining to mining sequences of single-item itemsets. As a reminder, the reasons why we consider such sequences are as follows: (i) Sequences of single-item itemsets represent one of the most important and popular type of sequence, e.g., DNA sequences, strings, web click streams etc. [WH04]; (ii) In financial news domain, we believe that the set of descriptors that describes a piece of news is the atomic unit for understanding its content. A subset of these descriptors is not enough to grasp the entire content of the news.

In fact, under this assumption, conventional GSPM algorithms can not be leveraged to discover higher order patterns for the following reasons:

- In GSPM algorithms, semantic relaxation augments items with each item's antecedents taken from the available taxonomies. However, when mining sequences of single-item itemsets, this relaxation has no benefit. If a single-item itemset x does not support a itemset x', its extended single-item itemset $x^{\dagger}$ will not support x since $\exists d \in x^{\dagger}$ such that $d \notin x$. Therefore, patterns at higher levels of representation can not be discovered.

- Sliding window w: The notion of sliding window is less relevant when mining sequences of single-items. Indeed, given a candidate pattern p and events e_j in p, events $x = \cup x_i$ within a sliding window w from a sequence s must be strictly equal to events x_j for sequence s to contribute to the support of p. In applications such as financial news, x_i is a set of descriptors that very precisely describes the semantic content a piece of news thanks to specific vocabulary. It is very unlikely the set of descriptors x of a random event e equals x_j.

For these reasons, GSPM algorithms degenerate into conventional SPM algorithms under the assumption of sequences of single-item itemsets and do not allow the discovery of any form of higher order knowledge. This observation comforts our study to transform input sequences into a more abstract representation beforehand then leverage conventional SPM techniques to extract higher order patterns.

8.2 General KDD frameworks

We proposed in STEAD a framework that implements all the necessary steps to extract knowledge from rich textual contents. Fayyad et al. [FPSS96] identified five steps in general *Knowledge Discovery from Databases* (KDD) processes: (i) Data selection, (ii) Data preprocessing, (iii) Data transformation, (iv) Data mining and (v) Results interpretation/evaluation. These steps are represented in Figure 4.5.

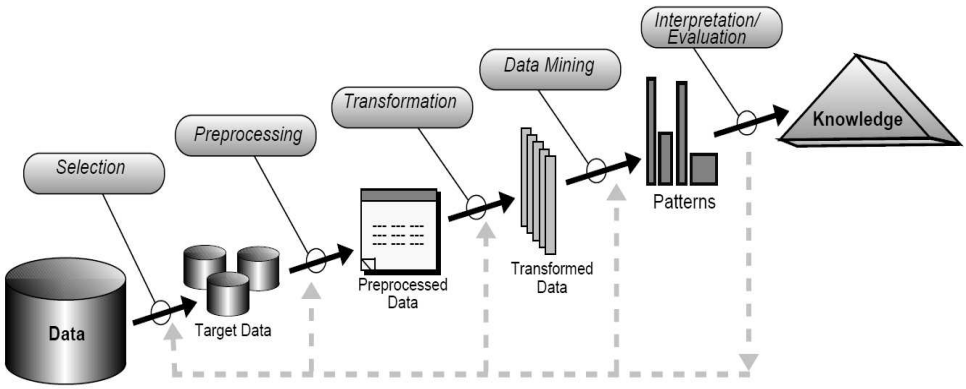


Figure 4.5: Overview of the steps in a KDD process (source: [FPSS96])

The data mining component in KDD processes has by far received the most attention in literature. However, we agree with Fayyad et al. [FPSS96] that other steps such as data preprocessing and data transformation for a particular type of mining are as important. This view is shared in Ding et al.'s work [DEWY06] where the authors propose a framework specifically designed for mining regional association rules in spacial datasets in 2 phases: (i) Discovery and identification of subregions of interest through clustering and (ii) spatial association rule mining.

More general frameworks were proposed by Tsai et al. [TT05] and Pan et al. [DS05] where the idea of using web services was introduced. Indeed, Tsai et al. [TT05] proposed a Service-Oriented Architecture (SOA) to support Dynamic Data Mining Processes. The authors throughly studied the service integration aspects (databases integration, mining services composition, etc.) of their architecture. However, even though analysts can search, design and execute mining tasks, their domain knowledge is a valuable source of insight that was not exploited. We believe in the importance of this knowledge for generating high quality preprocessed data which is accepted to significantly affect mining results. For this purpose, Pan et al. [DS05] proposed an architecture based on an ontology service that (i) shares and reuses previous knowledge, (ii) constitutes a knowledge discovery platform extending support for logical and physical data independence. Our contribution can be understood as a best pick between these frameworks where each service can benefit from (i) all the work done on services integration and (ii) exploit the expressiveness power of domain knowledge for addressing large volumes of textual contents.

9 Chapter summary

In this chapter, we have focused on the applicative aspect of time sequence summarization. TSAR was originally designed to build a support structure for chronology dependent applications on massive data sources, i.e., applications that require the time dimension of the data to be considered for the output to be meaningful. Sequential Pattern Mining is one such application. However, there can be a large gap between the purpose of an object and its actual utility. For this reason, we proposed to study how well TSAR summary could support Sequential Pattern Mining in a specific application, i.e., mine sequential patterns from Reuters's financial news archives. The purpose of this application is to discover sequential patterns from news data, then use the knowledge extracted to help analysts anticipate markets.

Roddick et al. introduced and formalized in [RSLC08] the concept of Higher Order Min-

ing, i.e., any form of data mining on non-primitive data. We showed that our motivation to operate conventional SPM methods on time sequences summaries could also be understood as a form of higher order mining. Hence, we build on top of the notions introduced by Roddick et al., and define and formalize knowledge that is extracted from collections of time sequence summaries as *Higher Order Patterns*. Therefore, we recalled the principles of the sequential pattern mining and thoroughly analyzed higher order patterns that could be extracted from TSaR summaries. We analytically established the relationships that link sequential patterns discovered in raw sequences to higher order patterns discovered in TSaR summaries. We showed that there are situations where this relationship could be completely characterized, but due to the summarization choices and to the variability of the data, the more general cases could not be characterized.

Equipped with this theoretical study, we proposed a new methodology to find very precise knowledge from higher order patterns extracted from TSaR summaries. This methodology is called *Exploratory mining* and relies on two mechanisms: (i) *Pattern Event Specialization* and (ii) *Pattern Event Recombination*. These two mechanisms directly address the processes in TSaR that reduce the accuracy of the data, i.e., (i) generalization and (ii) grouping.

We provided a thorough experimental study to evaluate how well TSaR summaries could support conventional SPM algorithms in the task of finding higher order patterns. Once again, we used Reuters's financial news archives for this study. Our results show that using TSaR summaries has made it possible to extract patterns having very low support, i.e., $\gamma \approx 0.01\%$, where conventional SPM algorithms simply could not complete. However, this operation has required the use of very compact summaries and such limitations could be considered as a shortcoming since compact summaries are obtained at the price of important loss of content and/or temporal accuracy. We address these observations by providing two usage scenarios. We demonstrated in these scenarios and on the dataset used that applying the exploratory mining methodology on higher order patterns discovered from very compact summaries still allows discovery of very precise and specific knowledge.

We mentioned that the purpose of mining Reuters's financial news archives is to use from historical data to help analysts anticipate markets. In fact, mining recurrent patterns in news archives only represents half of the task. Indeed, one orientation of future work is to evaluate the impact of patterns, that have been identified as frequent, on the market data. This problem can also be understood as evaluating the cross-correlation between patterns mined in news data and market data.

Conclusion

In this thesis work, we have investigated a particular form of data summarization called "Time sequence summarization". Time sequence summarization is a summarization activity specifically designed to represent time-varying data, e.g., time sequence of events, in a more concise, yet comprehensive and informative, representation. The originality of this representation resides in the dual and simultaneous consideration of the content and the temporal information associated to events. The motivation to design such summary structure originates from the need in domains such as medicine, the WWW, business or finance, for data structures capable of supporting process intensive and chronology dependent applications on very large data sources, i.e., applications that rely on the chronology of the data to produce meaningful and useful information.

In this context, we made the following contributions.

We started by introducing and formalizing the concept of a "Time sequence summary". We proposed several solutions to build a time sequence summary under this definition. The first solution is a user-oriented time sequence summarization approach called TSaR. TSaR is designed to take into account the user's preferences and understanding of specific domains to represent time sequences at different levels of abstraction. TSaR is a three phase process that builds on top of the *generalize and merge* paradigm introduced for Attribute Oriented Induction. The three phases are (i) generalization, (ii) grouping and (iii) concept formation. Generalization is responsible for representing events at a higher level of abstraction while grouping is responsible for gathering events *similar* at a given level of abstraction that are *close* on the timeline. Concept formation is responsible for representing groups formed this way by a single representative event. The semantic accuracy and the temporal accuracy of the summary are controlled by the user thanks to two input parameters for the generalization and grouping phases, respectively. We showed through an extensive set of experiments on real world data extracted from Reuters's financial news archives that TSaR has linear complexity and low memory footprint. Our technique produces summaries that can achieve high level of compression while preserving the quality of the data, from semantic and temporal view point.

Then, we proposed to improve our definition of a "Time sequence summary" by introducing the parameter-free property. Doing so, we reformulated the time sequence summarization problem into a novel conceptual clustering problem. The originality of this clustering problem definition resides in the objective function to optimize. Indeed, we introduced an objective function that takes simultaneously into account (i) the semantic content of events and (ii) their time of occurrence in the sequence. Hence, the objective function to optimize is a cost function to globally minimize. This cost considers the effort required to produce a common representation for a group of events and the effort required to gather these events together on the timeline. We proposed three solutions to solve this problem: (i) N-TSS a basic solution based on Exhaustive Enumeration, (ii) G-BUSS a greedy hierarchical ascending technique used as baseline and (iii) GRASS a random-

seeded parallel technique. G-BUSS is a conventional hierarchical ascending approach that uses our cost function. Since G-BUSS has quadratic computational complexity, we proposed a pruning technique based on the temporal component of the cost function. Our experiments showed that this pruning technique allows to reduce in practice G-BUSS's computational time by one order of magnitude. GRASS is a parallel algorithm based on the pruning technique introduced for G-BUSS that explores the search space at multiple locations simultaneously. Even though GRASS has in theory quadratic complexity, GRASS improves in practice G-BUSS's computational time by two orders of magnitude while producing summaries of quality similar to G-BUSS's.

Last, we studied how well a time sequence summarization technique such as TSaR could support in practice one specific chronology dependent application, namely, Sequential Pattern Mining. In fact, mining sequential patterns from summaries produced by TSaR can be understood as a form of *Higher Order Mining*. Therefore, we introduced the notion of *Higher Order Patterns*, i.e., sequential patterns that are discovered from time sequence summaries. We provided a complete analysis and characterization of the knowledge that can be extracted from time sequence summaries w.r.t. knowledge extracted from non summarized sequences. Thanks to the properties and characterizations presented, we proposed a new methodology for discovering more precise knowledge from higher order patterns discovered in summaries. This methodology is called *Exploratory mining* and relies on two mechanisms: (i) *Pattern Event Rearrangement* and (ii) *Pattern Event Specialization*. We proposed an extensive set of experiments and showed on some tangible examples taken from Reuters's financial news archives that exploratory mining allows to identify very precise and specific knowledge thanks to the use of higher order patterns discovered on the summaries.

Throughout this thesis work, we have identified at each stage of the summarization process several directions for future work. These perspectives are listed as follows:

- The datasets used for our experiments were generated form Reuters's 2003 financial news archives. The operations performed to preprocess the data are simple techniques that we limited by the need to extract pertinent descriptors from the free text and by the need to organize the descriptors extracted into taxonomies. Hence, one direction for future work is to leverage more advanced Natural Language Processing techniques to extract descriptors, e.g., OpenCalais [Reu], and organize descriptors. It would then be very interesting to study the summaries produced w.r.t. (i) the compression ratio that can be achieved and (ii) their quality.

- In this thesis work, we have focused on processing historical data. Therefore, time sequence summarization is operated in an offline mode and is not limited by the constraints of data streaming environments. One of the most interesting and challenging work orientation would be operating time sequence summarization in an online mode. Addressing summarization on data streams comes with several requirements: (i) improve the computational complexity of summaries produced, i.e., make the algorithms linear, (ii) maintain the summary in constrained memory space and (iii) handle the way the semantic content of events in the summary should be decayed w.r.t. passing time.

- A last, but not least, future work orientation concerns discovering knowledge from time sequence summaries. Indeed, we have only studied how sequential patterns could be extracted from textual contents, e.g., Reuters's financial news. In the introduction of this thesis we have presented the scenario of Google taking over Youtube. We gave examples of questions an investor might be asking himself when Bloomberg

announces that Google intends to buy DoubleClick. Mining sequential patterns from the news information thanks to the use of summaries is not enough to answer these questions. We also need to study how to cross-correlate patterns mined over the news with market data. This is another interesting and most challenging problem to tackle.

Bibliography

[ABB+04] A. Arasu, B. Babcock, S. Babu, J. McAlister, and J. Widom, *Characterizing memory requirements for queries over continuous data streams*, ACM Transactions on Database Systems **29** (2004), no. 1, 162–194.

[Abr06] T. Abraham, *Event sequence mining to develop profiles for computer forensic investigation purposes*, Proc. of the 2006 Australasian workshops on Grid computing and e-research, 2006, pp. 145–153.

[ABW06] A. Arasu, S. Babu, and J. Widom, *The cql continuous query language: semantic foundations and query execution*, The VLDB Journal **15** (2006), no. 2, 121–142.

[ADLS09] E. Agirre, O. L. De Lacalle, and A. Soroa, *Knowledge-based wsd on specific domains: performing better than generic supervised wsd*, Proc. of the 21st International Joint Conference on Artificial Intelligence, 2009, pp. 1501–1506.

[Agg06a] C. C. Aggarwal, *A framework for clustering massive text and categorical data streams*, Proc. of the 6th SIAM International Conference on Data Mining, 2006, pp. 477–481.

[Agg06b] ———, *On biased reservoir sampling in the presence of stream evolution*, Proc. of the 32nd International Conference on Very Large Databases, 2006.

[AHWY03] C. C. Aggrawal, J. Han, J. Wong, and P. S. Yu, *A framework for clustering evolving data streams*, Proc. of the 29th International Conference on Very Large Data Bases, 2003.

[AHWY07] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, *On clustering massive data streams: A summarization paradigm*, Advances in Database Systems, vol. 31, pp. 9–38, Springer US, 2007.

[AIS93] R. Agrawal, T. Imielenski, and A. Swami, *Mining association rules between sets of items in large databases*, Proc. of the 12th ACM SIGMOD International Conference on Management of Data (Washington D.C.), May 1993, pp. 207–216.

[Aki06] Tetsu Akiyama, *The continued growth of text information : From an analysis of information flow censuses taken during the past twenty years*, Keio communication review **25** (2006), 77–91.

[All79] L. G. Allan, *The perception of time*, Perception and Psychophysics **26** (1979), 340–354.

[AM04] A. Arasu and G. S. Manku, *Approximate counts and quantiles over sliding windows*, Proc. of the 24th ACM Symposium on Principles of Database Systems, 2004.

[AS94] R. Agrawal and R. Srikant, *Fast algorithms for mining association rules in large databases*, Proc. of the 20th International Conference on Very Large Databases (Santiago, Chile), September 1994, pp. 487–499.

[AS95] ———, *Mining sequential patterns*, Proc. of the 11th International Conference on Data Engineering, 1995.

[ASU86] A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers: principles, techniques, and tools*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.

[ATMS04] P. Andritsos, P. Tsaparas, R. Miller, and K. C. Sevick, *Limbo: Scalable clustering of categorical data*, Proc. of the 9th International Conference on Extending Database Technology, 2004, pp. 123–146.

[BADW82] A. Bolour, T. L. Anderson, L. J. Dekeyser, and H. K. T. Wong, *The role of time in information processing: a survey*, ACM SIGMOD Record **12** (1982), no. 3, 27–50.

[BBD+02] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, *Models and issues in data stream systems*, Proc. of the 22nd ACM Symposium on Principles of Database Systems, 2002.

[BC52] W. A. Bousfield and B. H. Cohen, *The effects of reinforcement on the occurrence of clustering in the recall of randomly arranged associates*, Tech. report, Connecticut Univ Storrs, 1952.

[BDMO02] B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan, *Sliding window computations over data streams*, Tech. Report 538, Stanford InfoLab, 2002.

[BDMO03] ———, *Maintaining variance and k-medians over data stream windows*, Proc. of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, 2003, pp. 234–243.

[Bec09] M. Bechchi, *Clustering-based approximate answering of query result in large and distributed databases*, Ph.D. thesis, University of Nantes, France, 2009.

[Ber02] P. Berkhin, *Survey of clustering data mining techniques*, Tech. report, Accrue Software, 2002.

[Bet01] B. Bettini, *Semantic compression of temporal data*, Proc. of the 2nd International Conference on Advances in Web-Age Information Management, 2001, pp. 267–278.

[BGA65] J. S. Bruner, J. J. Goodnow, and G. A. Austin, *A study of thinking*, New York: Wiley, 1965.

[BGR01] S. Babu, M. Garofalakis, and R. Rastogi, *Spartan: A model-based semantic compression system for massive data tables*, Proc. of the 20th ACM SIGMOD International Conference on Management of Data, 2001.

[BH01] A. Budanitsky and G. Hirst, *Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures*, 2001.

[Bih] Bihrmann, *The history of taxinomy*, http://www.bihrmann.com/caudiciforms/DIV/.

[BKKS01] M. M. Breuig, H.-P. Kriegel, P. Kroger, and J. Sander, *Data bubbles: Quality preserving performance boosting for hierarchical clustering*, Proc. of the 20th ACM SIGMOD International Conference on Management of Data, 2001.

[BLC02] D. Barbara, Y. Li, and J. Couto, *Coolcat: an entropy-based algorithm for categorical clustering*, Proc of the 11th International Conference on Information and Knowledge Management, 2002, pp. 582–589.

[BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila, *The semantic web*, Scientific American Magazine (2001), N/A.

[Blo90] R. A. Block, *Cognitive models of psychological time*, Lawrence Erlbaum Associates, 1990.

[BLV02] J. A. Barnden, M. G. Lee, and M. Viezzer, *Semantic networks*, The handbook of brain theory and neural networks (2nd edition) (M. A. Arbib, ed.), Bradford Books, MIT Press, Nov 2002, pp. 1010–1013.

[BM03] M. Babcock, B. Datar and R. Motwani, *Load shedding techniques for data stream systems*, Proc. of the 2003 Workshop on Management and Processing of Data Streams, 2003.

[BMS07] S. Budhaditya, L. Mihai, and V. Svetha, *Infrequent item mining in multiple data streams*, Proc. of the 7th IEEE International Conference on Data Mining Workshops, Oct 2007, pp. 569–574.

[Bri08] D. R. Brillinger, *Time series data analysis and theory (classics in applied mathematics - 36)*, SIAM: Society for Industrial and Applied Mathematics, 2008.

[BS05] A. Bulut and A. K. Singh, *A unified framework for monitoring data streams in real time*, Proc. of the 21st International Conference on Data Engineering, 2005.

[BS07] V. Boonjing and P. Songram, *Efficient algorithms for mining closed multidimensional sequential patterns*, Proc. of the 4th International Conference on Fuzzy Systems and Knowledge Discovery, 2007.

[Bud99] A. Budanitsky, *Lexical semantic relatedness and its application in natural language processing*, Tech. Report CSGR-390, Computer Systems Research Group, University of Toronto, Canada, 1999.

[Cai91] Y. Cai, *Attribute-oriented induction in relational databases*, Knowledge Discovery in Databases (1991), 213–228.

[CBKC07] V. Ch, A. Banerjee, V. Kumar, and V. Chandola, *Outlier detection: A survey*, Tech. report, Department of CSE, University of Minnesota, 2007.

[CC00] L. Cherkasova and G. Ciardo, *Characterizing temporal locality and its impact on web server performance*, Proc. of the International Conference on Computer Communication Networks, 2000.

[CC02] D. Carney and U. et al. Centintemel, *Monitoring streams - a new class of data management applications*, Proc. of the 28th International Conference on Very Large Databases, 2002, pp. 215–226.

[CC03] S. Chandrasekharan and O. et al Cooper, *Telegraphcq: Continuous dataflow processing for an uncertain world*, Proc. of the 1st Conference on Innovative Data Systems Research, 2003, pp. 269–280.

[CCC08] H.-L. Chen, K.-T. Chuang, and M.-S. Chen, *On data labeling for clustering categorical data*, IEEE Transactions on Knowledge and Data Engineering **20** (2008), no. 11, 1458–1472.

[CF02] S. Chandrasekaran and M. J. Franklin, *Streaming queries over streaming data*, Proc. of the 28th international conference on Very Large Data Bases, 2002, pp. 203–214.

[CGRS00] K. Chakrabarti, M. Garofalakis, R. Rastogi, and K. Shim, *Approximate query processing using wavelets*, Proc. of the 26th International Conference on Very Large Databases, 2000.

[CIK+08] K. Chandramouli, E. Izquierdo, T. Kliegr, J. Nemrava, and V. Svatek, *Wikipedia as the premiere source for targeted hypernym discovery*, Proc. of the WBBT workshop at ECML/PKDD, 2008.

[CJS03] C. Cranor, T. Johnson, and O. Spataschek, *Gigascope: a stream database for network applications*, Proc. of the 22nd ACM SIGMOD International Conference on Management of Data, 2003, pp. 647–651.

[CK05] V. Chandola and V. Kumar, *Summarization - compressing data into an informative representation*, Proc. of the 5th International Conference on Data Mining, 2005.

[CL05] K. Chen and L. Liu, *Detecting the change of clustering structure in categorical data streams*, Tech. Report GIT-CC-05-11, Georgia Institute of Technology, 2005.

[CMN99] S. Chaudhuri, Rajeev Motwani, and V. Narasayya, *On random sampling over joins*, Proc. of 18th ACM SIGMOD International Conference on Management of Data, 1999.

[CMR05] G. Cormode, S. Muthukrishnan, and I. Rozenbaum, *Summarizing and mining inverse distributions on data streams via dynamic inverse sampling*, Proc. of the 31st International Conference on Very Large Databases, 2005.

[Cod69] E. F. Codd, *Derivability, redundancy, and consistency of relations stored in large data banks*, Tech. report, IBM Thomas J. Watson Research Center, 1969.

[Cod70] ———, *A relational model of data for large shared data banks*, Communications of the ACM **13** (1970), no. 6, 377–387.

[CR04] C. Carpineto and R. Romano, *Exploiting the potential of concept lattices for information retrieval with credo*, Journal of Universal Computer Science **10** (2004), no. 8, 985–1013.

[CS06] E. Cohen and M. J. Strauss, *Maintaining time-decaying stream aggregates*, Journal of Algorithms **59** (2006), no. 1, 19–36.

[CSSX09] G. Cormode, V. Shkapenyuk, D. Srivastava, and B. Xu, *Forward decay: A practical time decay model for streaming systems*, Proc. of the 25th International Conference on Data Engineering, 2009, pp. 138–149.

[Cuk10] K. N. Cukier, *Managing information. information has become superabundant*, The Economist: Special reports (2010), N/A.

[CYL+08] L. Chao, Z. Yanchang, C. Longbing, O. Yuming, and L. Li, *Outlier mining on multiple time series data in stock market*, Proc. of the 10th Pacific Rim International Conference on Artificial Intelligence, 2008, pp. 1010–1015.

[Dat04] C. J. Date, *An introduction to database systems, 8th edition*, Addison-Wesley, 2004.

[DDL02] C.J. Date, H. Darwen, and N. A. Lorentzos, *Temporal data and the relational model*, Morgan Kaufmann Publishers Inc., 2002.

[Den05] P. J. Denning, *The locality principle*, Communications of the ACM **48** (2005), no. 7, 19–24.

[DEWY06] W. Ding, C. F. Eick, J. Wang, and X. Yuan, *A framework for regional association rule mining in spatial datasets*, Proc. the 6th International Conference on Data Mining, 2006.

[DF02a] Y. Ding and S. Foo, *Ontology research and development. part 1 - a review of ontology generation*, Journal of Information Science **28** (2002), no. 2, 123–136.

[DF02b] ———, *Ontology research and development. part 2 - a review of ontology mapping and evolving*, Journal of Information Science **28** (2002), no. 5, 375–388.

[DGGR02] A. Dobra, M. Garofalakis, J. Gehrke, and R. Rastogi, *Processing complex aggregate queries over data streams*, Proc. of the 21st ACM SIGMOD International Conference on Management of Data, 2002.

[DH73] R. O. Duda and P. E. Hart, *Pattern classification and scene analysis*, John Wiley and Sons Inc., Jan 1973.

[DH01] P. Domingos and G. Hulten, *A general method for scaling up machine learning algorithms and its application to clustering*, Proc. of the 18th International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., 2001, pp. 106–113.

[Dob05] A. Dobra, *Histograms revisited: When are histograms the best approximation method for aggregates over joins?*, Proc. of the 25th ACM Symposium on Principles of Database Systems, 2005.

[DS05] P. Ding and J.-Y. Shen, *Ontology service-based architecture for continuous knowledge discovery*, Proc. of the International Conference on Machine Learning and Cybernetics, 2005.

[DSS93] R. Davis, H. Shrobe, and P. Szolovits, *What is a knowledge representation?*, AI Magazine **14** (1993), no. 1, 17–33.

[EN06] C. Estan and J. F. Naughton, *End-biased samples for join cardinality estimation*, Proc. of the 22nd International Conference on Data Engineering, 2006.

[ENK06] F. Eichinger, D. D. Nauck, and F. Klawonn, *Sequence mining for customer behaviour predictions in telecommunications*, Proc. of the Workshop on Practical Data Mining: Experiences and Challenges at PKDD, 2006, pp. 3–10.

[fBIRatSUSoM] Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine, *Protégé, the ontology editor and knowledge-base framework*, http://protege.stanford.edu/.

[Fis86] D. H. Fisher, *A proposed method of conceptual clustering for structured and decomposable objects*, Machine learning: a guide to current research (1986), 67–70.

[Fis87] ———, *Knowledge acquisition via incremental conceptual clustering*, Machine Learning **2** (1987), no. 2, 139–172.

[FPSS96] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, *Knowledge discovery and data mining: Towards a unifying framework*, Proc. of the 2nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1996.

[Fre60] E. Fredkin, *Trie memory*, Communications of the ACM **3** (1960), no. 9, 490–499.

[Gar07] Gartner, *Gartner says worldwide relational database market increased 14 percent in 2006*, http://www.gartner.com/it/page.jsp?id=507466, 2007.

[GCM+08] J. F Gantz, C. Chute, A. Manfrediz, S. Minton, D. Reinsel, W. Schlichting, and A. Toncheva, *The diverse and exploding digital universe*, IDC white paper (2008) **2** (2008), 2–15.

[Gen89] J. H. Gennari, *Focused concept formation*, Proc. of the 6th International Workshop on Machine learning, 1989, pp. 379–382.

[GGI+02] A. C. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss, *Fast, small-space algorithms for approximate histogram maintenance*, Proc. of the 34th Annual ACM Symposium on Theory of Computing, 2002.

[GGR99] V. Ganti, J. Gehrke, and R. Ramakrishnan, *Cactus: Clustering categorical data using summaries*, Proc. of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data mining, 1999, pp. 73–83.

[GH06] L. Geng and H. J. Hamilton, *Interestingness measures for data mining: A survey*, ACM Computer Surveys **38** (2006), no. 3, N/A.

[GJ79] M. R. Garey and D. S. Johnson, *Computers and intractability: A guide to the theory of np-completeness*, W. H. Freeman & Co., New York, NY, USA, 1979.

[GK01] M. Greenwald and S. Khanna, *Space-efficient online computation of quantile summaries*, Proc. of the 20th ACM SIGMOD International Conference on Management of Data, 2001.

[GKMS01] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss, *Surfing wavelets on streams: One-pass summaries for approximate aggregate queries*, Proc. of the 27th International Conference on Very Large Databases, 2001.

[GKS01] S. Guha, N. Koudas, and K. Shim, *Data-streams and histograms*, Proc. of the 33rd Annual ACM Symposium on Theory of Computing, 2001.

[GKS04] S. Guha, C. Kim, and K. Shim, *Xwave: Optimal and approximate extended wavelets for streaming data*, Proc. of the 30th International Conference on Very Large Databases, 2004.

[GLF89] J. H. Gennari, P. Langley, and D. H. Fisher, *Models of incremental concept formation*, Artificial Intelligence **40** (1989), no. 1-3, 11–61.

[GLSS06] J. Ghosh, D. Lambert, D. B. Skillicorn, and J. Srivastava, *Detecting the change of clustering structure in categorical data streams*, Proc. of the 6th SIAM International Conference on Data Mining, 2006.

[GM99] A. Gupta and I. S. Mumick, *Maintenance of materialized views: problems, techniques, and applications*, MIT Press, Cambridge, MA, USA, 1999.

[GMM+03] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan, *Clustering data streams: Theory and practice*, IEEE Transactions on Knowledge and Data Engineering **15** (2003), no. 3, 515–528.

[GMMO00] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, *Clustering data streams*, Proc. of the 41st Annual Symposium on Foundations of Computer Science, 2000, pp. 359–366.

[Gol02] L. Golab, *Sliding window query processing over data streams*, Tech. Report CS-2006-27, University of Waterloo, 2002.

[Goo] Google, *Google finance*, http://finance.google.com.

[GRM05] N.K. Govindaraju, N. Raghuvanshi, and D. Manocha, *Fast and approximate stream mining of quantiles and frequencies using graphics processors*, Proc. of the 24th ACM SIGMOD International Conference on Management of Data, 2005.

[GRS99] S. Guha, R. Rastogi, and K. Shim, *Rock: A robust clustering algorithm for categorical attributes*, Proc. of the 15th International Conference on Data Engineering, 1999, p. 512.

[Gru93] T. R. Gruber, *A translation approach to portable ontology specifications*, Knowledge Acquisition **6** (1993), no. 2, 199–221.

[GSW04] S. Guha, K. Shim, and J. Woo, *Rehist: Relative error histogram construction algorithms*, Proc. of the 30th International Conference on Very Large Databases, 2004.

[HA04] V. Hodge and J. Austin, *A survey of outlier detection methodologies*, Artificial Intelligence Review **22** (2004), no. 2, 85–126.

[HAB07] C. Harrison, S. Amento, B. Kuznetsov, and R. Bell, *Rethinking the progress bar*, Proc. of the 20th Annual ACM Symposium on User interface Software and Technology, 2007, pp. 115–118.

[HAF+03] M. A. Hammad, W. G. Aref, M. J. Franklin, M. F. Mokbel, and A. K. Elmagarmid, *Efficient execution of sliding-window queries over data streams*, Tech. Report TR03-035, Department of Computer Sciences, Purdue University, 2003.

[Ham50] R. W. Hamming, *Error detecting and error correcting codes*, The Bell System Technical Journal **XXVI** (1950), no. 2, 147–160.

[Ham94] J. D. Hamilton, *Time series analysis*, Princeton University Press, 1994.

[HBMB05] G. S. Halford, R. Baker, J. E. McCredden, and J. D. Bain, *How many variables can humans process?*, Psychological Science - Cambridge **16** (2005), no. 1, 70–76.

[HC08] K.-Y. Huang and C.-H. Chang, *Efficient mining of frequent episodes from complex sequences*, Inf. Syst. **33** (2008), no. 1, 96–114.

[HCCC92] J. Han, Y. Cai, O. Cai, and N. Cercone, *Knowledge discovery in databases: An attribute-oriented approach*, Proc. of 18th International Conference on Very Large Data Bases, 1992, pp. 547–559.

[Hea92] M. Hearst, *Automatic acquisition of hyponyms from large text corpora*, Proc. of the 14th International Conference on Computational Linguistics, 1992, pp. 539–545.

[Hei99] S. Heiler, *A survey on nonparametric time series analysis*, Tech. Report 9904005, EconWPA, Apr 1999.

[Her03] T. Herdershott, *Electronic trading in financial markets*, IT Pro (2003), 10–14.

[HF96] J. Han and Y. Fu, *Exploration of the power of attribute-oriented induction in data mining*, Advances in Knowledge Discovery and Data Mining (1996), 399–421.

[HFH+94] J. Han, Y. Fu, Y. Huang, Y. Cai, and N. Cercone, *Dblearn: a system prototype for knowledge discovery in relational databases*, Proc. of the 13th ACM SIGMOD International Conference on Management of Data, 1994, p. 516.

[HFW+96] J. Han, Y. Fu, W. Wang, J. Chiang, O. R. Zaïane, and K. Koperski, *Dbminer: interactive mining of multiple-level knowledge in relational databases*, SIGMOD Record **25** (1996), no. 2, 550.

[HK04] P. J. Haas and C. König, *A bi-level bernouilli scheme for database sampling*, Proc. of the 23rd ACM SIGMOD International Conference on Management of Data (Paris, France), Jun 2004.

[Hoe63] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, Journal of the American Statistical Association **58** (1963), no. 301, 13–30.

[Hog78] W. H. Hogan, *A theoretical reconciliation of competing views of time perception*, The American Journal of Psychology **91** (1978), no. 3, 417–428.

[HPY05] J. Han, J. Pei, and X. Yan, *Sequential pattern mining by pattern-growth: principles and extensions*, Studies In Fuzziness And Soft Computing, vol. 180, pp. 183–220, Springer Berlin, 2005.

[HS02] A. Hotho and G. Stumme, *Conceptual clustering of text clusters*, Proc. of FGML Workshop, 2002.

[HT85] S. R. Hiltz and M. Turoff, *Structuring computer-mediated communication systems to avoid information overload*, Communications of the ACM **28** (1985), no. 7, 680–689.

[Hua98] Z. Huang, *Extensions to the k-means algorithm for clustering large data sets with categorical values*, Data Mining and Knowledge Discovery **2** (1998), no. 3, 283–304.

[HXDH03] Z. He, X. Xu, S. Deng, and J. Z. Huang, *Clustering categorical data streams*, Proc. of the International Conference in Computational Methods in Science and Engineering, 2003.

[HY00] J. Han and Y. Yin, *Mining frequent patterns without candidate generation*, Proc. of the 19th ACM SIGMOD International Conference on Management of Data, 2000.

[HY06] Y. Hirate and H. Yamana, *Generalized sequential pattern mining with time interval*, Proc. of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2006.

[IM09] D. Ienco and R. Meo, *Distance based clustering for categorical data*, Proc. of the 17th Italian Symposium on Advanced Database Systems, 2009.

[iPo] iPolicyNetworks, http://www.ipolicynetworks.com.

[Jac01] P. Jaccard, *étude comparative de la distribution florale dans une portion des alpes et des jura*, Bulletin de la Société Vaudoise des Sciences Naturelles **37** (1901), 547Ű–579.

[JC97] J. J. Jiang and D. W. Conrath, *Semantic similarity based on corpus statistics and lexical taxonomy*, Proc. of International Conference Research on Computational Linguistics, 1997.

[JHC00] I. Jonyer, L. B. Holder, and D. J. Cook, *Graph-based hierarchical conceptual clustering*, International Journal on Artificial Intelligence Tools **2** (2000), 107–135.

[JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn, *Data clustering: a review*, ACM Computer Survey **31** (1999), no. 3, 264–323.

[JMN99] H. V. Jagadish, J. Madar, and R. T. Ng, *Semantic compression and pattern extraction with fascicles*, Proc. of 25th International Conference on Very Large Data Bases, 1999.

[JMR05] T. Johnson, S. Muthukrishnan, and I. Rozenbaum, *Sampling algorithms in a stream operator*, Proc. of the 24th ACM SIGMOD International Conference on Management of Data, 2005.

[JMS95] H. V. Jagadish, I. S. Mumick, and A. Silberschatz, *View maintenance issues for the chronicle data model (extended abstract)*, Proc. of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 1995, pp. 113–124.

[JNOT04] H. Jagadish, R. Ng, B. Ooi, and A. Tung, *Itcompress: an iterative semantic compression algorithm*, Proc. of ICDE, 2004.

[JPA04] C. Jermaine, A. Pol, and S. Arumugam, *Online maintenance of very large random samples*, Proc. of the 23rd ACM SIGMOD International Conference on Management of Data, Jun 2004.

[KCN+08] T. Kliegr, K. Chandramouli, J. Nemrava, V. Svatek, and E. Izquierdo, *Combining image captions and visual analysis for image concept classification*, Proc. of the 9th International Workshop on Multimedia Data Mining, 2008, pp. 8–17.

[KJF97] F. Korn, H. V. Jagadish, and C. Faloutsos, *Efficiently supporting ad hoc queries in large datasets of time sequences*, Proc. of the 16th ACM SIGMOD International Conference on Management of Data, 1997.

[Kli93] N. Kline, *An update of the temporal database bibliography*, ACM SIGMOD Record **22** (1993), 66–80.

[KM05] P. Karras and N. Mamoulis, *One-pass wavelet synopses for maximum-error metrics*, Proc. of the 31st International Conference on Very Large Databases, 2005.

[KNRC05] R. Kaushik, J. F. Naughton, R. Ramakrishnan, and V. T. Chakravarthy, *Synopses for query optimization: A space-complexity perspective*, ACM Transactions on Database Systems **30** (2005), no. 4, 1102–1127.

[Kro97] R. Krovetz, *Homonymy and polysemy in information retrieval*, Proc. of the 8th conference on European chapter of the Association for Computational Linguistics, Association for Computational Linguistics, 1997, pp. 72–79.

[KS02] G. Karakostas and D.N. Serpanos, *Exploitation of different types of locality for web caches*, Proc. of the 7th IEEE Symposium on Computers and Communications, 2002.

[KT08] J. Kiernan and E. Terzi, *Constructing comprehensive summaries of large event sequences*, Proc. of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008.

[KT09] ———, *Constructing comprehensive summaries of large event sequences*, ACM Transactions on Knowledge Discovery from Data **3** (2009), no. 4, 1–31.

[Lab] Princeton University Cognitive Science Lab, *Wordnet*, http://wordnet.princeton.edu/.

[LC98] C. Leacock and M. Chodorow, *Combining local context and wordnet similarity for word sense identification*, pp. 265Ű–283, The MIT Press, 1998.

[Les86] M. Lesk, *Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone*, Proc. of the 5th annual International Conference on Systems Documentation, 1986, pp. 24–26.

[Lev65] I. Levenshtein, *Error detecting and error correcting codes*, Doklady Akademii Nauk SSSR **163** (1965), no. 4, 845–848.

[Lin98] D. Lin, *An information-theoretic definition of similarity*, Proc. of the 15th International Conference on Machine Learning, Morgan Kaufmann, San Francisco, CA, 1998, pp. 296–304.

[LKLC03] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, *A symbolic representation of time series, with implications for streaming algorithms*, Proc. of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, 2003.

[LLD+07] M. Li, M. Liu, L. Ding, E. A. Rundensteiner, and M. Mani, *Event stream processing with out-of-order data arrival*, Proc. of the 27th International Conference on Distributed Computing Systems Workshops (Washington, DC, USA), IEEE Computer Society, 2007, p. 67.

[LLSYW02] M.-Y. Lin, Lee, S.-Y., and S.-S. Wang, *Delisp: Efficient discovery of generalized sequential patterns by delimited pattern-growth technology*, Proc. of the 6th Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2002.

[LR98] L. Lin and Risch, *Querying continuous time sequences*, Proc. of the 24th International Conference on Very Large Databases, 1998.

[LRSB96] L. Lin, T. Risch, M. Skold, and D. Badal, *Indexing values of time sequences*, Proc. of the 5th ACM Conference on Information and Knowledge Management, November 1996.

[LXLY04] X. Lin, J. Xu, H. Lu, and J.X. Yu, *Continuously maintaining quantile summaries of the most recent n elements over a data stream*, Proc. of the 20th International Conference on Data Engineering, Apr 2004.

[Man67] G. Mandler, *Organization and memory*, The psychology of learning and motivation **1** (1967), 327–372.

[MC98] F. Masseglia and F. Cathala, *The psp approach for mining sequential patterns*, Proc. of the 2nd European Conference on Principles of Data Mining and Knowledge Discovery, 1998.

[McG81] William C. McGee, *Data base technology*, IBM Journal of Research and Development (1981), 505–519.

[MFHH03] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, *The design of an acquisitional query processor for sensor networks*, Proc. of the 22nd ACM SIGMOD International Conference on Management of Data, 2003, pp. 491–502.

[Mic80] R. S Michalski, *Knowledge acquisition through conceptual clustering: A theoretical framework and algorithm for partitioning data into conjunctive concepts*, International Journal of Policy Analysis and Information Systems **4** (1980), 219–243.

[Mic93] R. S. Michalski, *A theory and methodology of inductive learning*, Readings in knowledge acquisition and learning: automating the construction and improvement of expert systems (1993), 323–348.

[Mil62] G. A. Miller, *Information input overload*, Proc. of the Conference on Self-Organizing Systems, 1962.

[Min02] J.-K. Min, *Beast : A buffer replacement algorithm using spatial and temporal locality*, Proc. of the International Conference on Computational Science and its Applications, 2002.

[MM02] R. Motwani and G. S. Manku, *Approximate frequency counts over data streams*, Proc. of the 28th International Conference on Very Large Databases, 2002.

[MPT04] F. Masseglia, P. Poncelet, and M. Teisseire, *Pre-processing time constraints for efficiently mining generalized sequential patterns*, Proc. of the 11th International Symposium on Temporal Representation and Reasoning, 2004.

[MR04] C. H. Mooney and J. F. Roddick, *Mining relationships between interacting episodes*, Proc. of the 4th SIAM International Conference on Data Mining, 2004, pp. 21–8.

[MS80] R. S Michalski and R. Stepp, *Learning from observation: conceptual clustering*, Morgan Kauffmann, 1980.

[MTAI97] H. Mannila, H. Toivonen, and V. A. Inkeri, *Discovery of frequent episodes in event sequences*, Data Mining and Knowledge Discovery **1** (1997), no. 3, 259–289.

[MTT04] F. Masseglia, D. Tanasa, and B. Trousse, *Web usage mining: Sequential pattern extraction with a very low support*, Proc. of the 6th Asia Pacific Web Conference, 2004.

[MTV94] H. Mannila, H. Toivonen, and A. I. Verkamo, *Efficient algorithms for discovering association rules*, Proc. of the 1994 Workshop on Knowledge Discovery in Databases (Seattle, Washington), 1994, pp. 181–192.

[MTV97] ———, *Discovery of frequent episodes in event sequences*, Data Mining and Knowledge Discovery **1** (1997), no. 3, 259–289.

[Mut05] S. Muthukrishnan, *Data streams: Algorithms and applications*, Now Publishers Inc, 2005.

[MWA+03] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. S. Manku, C. Olston, J. Rosenstein, and R. Varma, *Query processing, approximation, and resource management in a data stream management system*, Proc. of the 1st Biennial Conference on Innovative Data Systems Research, 2003.

[Noh04] S.-Y. Noh, *Literature review on temporal, spatial, and spatio-tempoal data models and temporal query languages*, Tech. Report 04-12, Computer Science, Iowa State University, 2004.

[Nvi] Nvidia, *Fermi*, http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf.

[OKUA09] H. Ohtani, T. Kida, T. Uno, and H. Arimura, *Efficient serial episode mining with minimal occurrences*, Proc. of the 3rd International Conference on Ubiquitous Information Management and Communication, 2009, pp. 457–464.

[Olk93] F. Olken, *Random sampling from databases*, Ph.D. thesis, University of California at Berkeley, 1993.

[OLS+02] D. Ogle, E. Labadie, J. Schoech, H. Kreger, M. Chessell, M. Wamboldt, A. Salashour, B. Horn, J. Cornpropst, and J. Gerken, *Canonical situation data format: The common base event v1.0.1*, Tech. report, International Business Machines Corporation, 2002.

[Ont] OntologyOnline, http://ontologyonline.org/visualisation/c/Directory/Domain_ontology.

[OR90] F. Olken and D. Rotem, *Random sampling from database files: a survey*, Proc. of the 5th International Conference on Statistical and Scientific Database Management, Springer-Verlag New York, Inc., 1990, pp. 92–111.

[OWNL04] K.-L. Ongl, Li Wenyuan, W.-K. Ng, and E.-P. Lim, *Sclope: An algorithm for clustering data streams of categorical attributes*, Proc. of the International conference on data warehousing and knowledge discovery, Sep 2004.

[PBTL99] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, *Discovering frequent closed itemsets for association rules*, Proc. of the 7th International Conference on Database Theory, 1999, pp. 398–416.

[Pea78] J. Pearl, *Causality Ũ models, reasoning, and inference*, Cambridge University Press, 1978.

[PHMA+01] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, *Prefixspan: Mining sequential patterns efficiently by prefix projected pattern growth*, Proc. of the 17th International Conference on Data Engineering, 2001.

[PIHS96] V. Poosala, Y. E. Ioannisdis, P. J. Haas, and E. J. Shekita, *Improved histograms for selectivity estimation of range predicates*, Proc. of the 22nd International Conference on Very Large Databases, 1996.

[PMR06] Q.-K. Pham, N. Mouaddib, and G. Raschia, *Data stream synopsis using saintetiq*, Proc. of the International Conference on Flexible Query Answering Systems, 2006.

[PRSP+09] Q.-K. Pham, G. Raschia, R. Saint-Paul, B. Benetallah, and N. Mouaddib, *Time sequence summarization to scale up chronology-dependent applications*, Proc. of the 18th ACM Conference on Information and Knowledge Management, November 2009.

[PS07] S. P. Ponzetto and M. Strube, *Deriving a large scale taxonomy from wikipedia*, Proc. of AAAI Conference on Artificial Intelligence, 2007.

[PSPB+08] Q.-K. Pham, R. Saint-Paul, B. Benetallah, G. Raschia, and N. Mouaddib, *Mine your own business! mine others' news!*, Proc. of the 11th Conference on Extending Database Technology, 2008.

[PWP08] Christopher D Pilcher, Joseph K Wong, and Satish K Pillai, *Inferring hiv transmission dynamics from phylogenetic sequence relationships*, PLoS Med **5** (2008), no. 3, 69–71.

[Rad07] M. Rada, *Using wikipedia for automatic word sense disambiguation*, Proc. of the North American Chapter of the Association for Computational Linguistics, 2007.

[RCYBY04] J.-D. Ren, Cheng, Y.-B., and L.-L. Yang, *An algorithm for mining generalized sequential patterns*, Proc. of the International Conference on Machine Learning and Cybernetics, 2004.

[Res95] P. Resnik, *Using information content to evaluate semantic similarity in a taxonomy*, Proc. of the 1995 International Joint Conference on AI, 1995, pp. 448–453.

[Reu] Thomson Reuters, *Opencalais*, http://www.opencalais.com.

[RGY09] F. A. Rabhi, A. Guabtni, and L. Yao, *A data model for processing financial market and news data*, International Journal of Electronic Finance **3** (2009), no. 4, 387–403.

[RKS07] G. T. Raju, Kunal, and P. S. Satyanarayana, *Knowledge discovery from web usage data: Extraction of sequential patterns through art1 neural network based clustering algorithm*, Proc. of the International Conference on Computational Intelligence and Multimedia Applications, 2007.

[RM02] G. Raschia and N. Mouaddib, *Saintetiq: a fuzzy set-based approach to database summarization*, Fuzzy Sets Systems **129** (2002), no. 2, 137–162.

[RMBB89] R. Rada, H. Mili, E. Bicknell, and M. Blettner, *Development and application of a metric on semantic nets*, IEEE Transactions on Systems, Man and Cybernetics **19** (1989), no. 1, 17–30.

[RSLC08] J. F. Roddick, M. Spiliopoulou, D. Lister, and A. Ceglar, *Higher order mining*, SIGKDD Explorations Newsletter **10** (2008), no. 1, 5–17.

[SA96] R. Srikant and R. Agrawal, *Mining sequential patterns: Generalizations and performance improvements*, Proc. of the 5th International Conference on Extending Database Technology, 1996.

[SBC06] A. Srinivasan, D. Bhatia, and S. Chakravarthy, *Discovery of interesting episodes in sequence data*, Proc. of the 2006 ACM Symposium on Applied Computing, 2006, pp. 598–602.

[SBI06] P. Songram, V. Boonjing, and S. Intakosum, *Closed multidimensional sequential pattern mining*, Proc. of the 3rd International Conference on Information Technology: New Generations, 2006.

[SCDT00] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan, *Web usage mining: Discover and applications of usage patterns from web data*, SIGKDD Explorations Newsletter **1** (2000), no. 2, 12–23.

[SheBC] Emperor Shen, *Shen nung pen ts'ao ching*, N/A, $\sim$ 2650B.C.

[SJN06] R. Snow, D. Jurafsky, and A. Y. Ng, *Semantic taxonomy induction from heterogenous evidence*, Proc. of the 21st International Conference on Computational Linguistics, 2006, pp. 801–808.

[SKSC06] B. Smith, W. Kusnierczyk, D. Schober, and W. Ceusters, *Towards a reference terminology for ontology research and development in the biomedical domain*, Proc. of the 2nd International Workshop on Formal Biomedical Knowledge Representation, 2006.

[SMG02] W. Shi, M. H. MacGregor, and P. Gburzynski, *On temporal locality in ip address sequences*, IEICE Transactions **E85-B** (2002), no. 1, 3352–3354.

[Smy78] M. Smyth, *Power domains*, Journal of Computer and System Sciences **16** (1978), no. 1, 23–36.

[SOL03] X. Sun, M. E. Orlowska, and X. Li, *Introducing uncertainty into pattern discovery in temporal event sequences*, Proc. of the 3rd International Conference on Data Mining, 2003.

[Soo91] M. D. Soo, *Bibliography on temporal databases*, ACM SIGMOD Record **20** (1991), no. 1, 14–23.

[Spi99a] M. Spiliopoulou, *Managing interesting rules in sequence mining*, Proc. of the 3rd European Conference on Principles of Data Mining and Knowledge Discovery, 1999, pp. 554–560.

[Spi99b] ———, *Managing interesting rules in sequence mining*, Proc. of the 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases, 1999.

[SPRM05] R. Saint-Paul, G. Raschia, and N. Mouaddib, *General purpose database summarization*, Proc. of 31st International Conference on Very Large Data Bases, 2005.

[Srg] CSE at the University of New South Wales SOC research group, *Adage project*, http://adage.unsw.edu.au/.

[SS98] R. Stam and R. Snodgrass, *A bibliography on temporal databases*, Database Engineering **4** (1998), 231–239.

[Sul96] Mark Sullivan, *Tribeca: A stream database manager for network traffic analysis*, Proc. of the 22nd International Conference on Very Large Databases, 1996.

[SZ04] D. Shasha and Y. Zhu, *High performance discovery in time series*, Springer, 2004.

[TB01] L. Talavera and J. Bejar, *Generality-based conceptual clustering with probabilistic concepts*, IEEE Transactions on Pattern Analysis and Machine Intelligence **23** (2001), no. 2, 196–206.

[TCZ+03] N. Tatbul, U. Cetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker, *Load shedding in a data stream manager*, Proc. of the 29th International Conference on Very Large Data Bases, VLDB Endowment, 2003, pp. 309–320.

[TGIK03] N. Thaper, S. Guha, P. Indyk, and N. Koudas, *Dynamic multidimensional histograms*, Proc. of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, Jun 2003.

[TGNO92] D. Terry, D. Goldberg, D. Nichols, and B. Oki, *Continuous queries over append-only databases*, Proc. of the 11th ACM SIGMOD International Conference on Management of Data, 1992, pp. 321–330.

[TID] DARPA TIDES, http://projects.ldc.upenn.edu/TIDES/index.html.

[TL91] K. Thompson and P. Langley, *Concept formation in structured domains*, pp. 127–161, Morgan Kaufmann Publishers Inc., 1991.

[TLHY03] Y. Tang, L. Liang, R. Huang, and Y. Yu, *Bitemporal extensions to non-temporal rdbms in distributed environment*, Proc. of the 8th International Conference on Computer Supported Cooperative Work in Design, 2003, pp. 370–374.

[TPB99] N. Tishby, F. Pereira, and W. Bialek, *The information bottleneck method*, Proc. of the 37th Allerton Conference on Communication, Control and Computation, 1999, pp. 368Ű–377.

[TPBM05] M. Teisseire, P. Poncelet, G. Besse, and F. Masseglia, *Sequential pattern mining: A survey on issues and approaches*, pp. 3–29, Oxford University Press, 2005.

[Tra] Traderbots, http://www.traderbots.com.

[TT05] C.-Y. Tsai and M.-H. Tsai, *A dynamic web service based data mining process system*, Proc. of the 5th International Conference on Computer and Information Technology, 2005.

[Tul93] E. Tulving, *What is episodic memory?*, Current Directions in Psychological Science **2** (1993), no. 3, 67–70.

[Tul02] ———, *Episodic memory: From mind to brain*, Annual Review of Psychology **52** (2002), 1–25.

[TYH03] P. Tzvetkov, X. Yan, and J. Han, *Tsp: mining top-k closed sequential patterns*, Proc. the 3rd International Conference on Data Mining, 2003.

[Van] E. Vanderlinden, *The finance ontology website*, http://www.fadyart.com/index.html.

[Vit85] J. S. Vitter, *Random sampling with a reservoir*, ACM Transactions on Mathematical Software (New York, NY, USA), ACM Press, 1985, pp. 37–57.

[VRUM04] A. W. Voglozin, G. Raschia, L. Ughetto, and N. Mouaddib, *Querying the saintetiq summaries: A first attempt*, Proc. of the International Conference on Flexible Query Answering Systems, 2004.

[VW99] J. S. Vitter and M. Wang, *Approximate computation of multidimensional aggregates of sparse data using wavelets*, Proc. of the 18th ACM SIGMOD International Conference on Management of Data, 1999.

[WA05] Q. Wan and A. An, *Compact transaction database for efficient frequent pattern mining*, Proc. of the 2005 IEEE International Conference on Granular Computing, 2005.

[War63] J. H. Jr. Ward, *Hierarchical grouping to optimize an objective function*, Journal of the American Statistical Association **58** (1963), no. 301, 236–244.

[WFZ+08] S. Wang, Y. Fan, C. Zhang, H. Xu, X. Hao, and Y. Hu, *Entropy based clustering of data streams with mixed numeric and categorical values*, Proc. of the 7th IEEE/ACIS International Conference on Computer and Information Science, IEEE Computer Society, 2008, pp. 140–145.

[WH04] J. Wang and J. Han, *Bide: Efficeint mining of frequent closed sequences*, Proc. of the 20th International Conference on Data Engineering, 2004.

[Win] Wintercorp, http://www.wintercorp.com/Newsletter/Scaling the Data Warehouse IW 17Oct08.pdf.

[WJW98] Y. Wu, S. Jajodia, and X. S. Wang, *Temporal database bibliography update*, vol. 1399/1998, Springer, 1998.

[WK06] J. Wang and Karypis, *On efficiently summarizing categorical databases*, Knowledge and Information Systems **9** (2006), no. 1, 19–37.

[WP94] Z. Wu and M. Palmer, *Verb semantics and lexical selection*, Proc. of the 32nd. Annual Meeting of the Association for Computational Linguistics (New Mexico State University, Las Cruces, New Mexico), 1994, pp. 133–138.

[WRZ05] A. Wright, T. N. Ricciardi, and M. Zwickc, *Application of information-theoretic data mining techniques in a national ambulatory practice outcomes research network*, Proc. of AMIA Annual Symposium, 2005.

[WS03] H. Wang and K. C. Sevcik, *A multi-dimensional histogram for selectivity estimation and fast approximate query answering*, Proc. of the 2003 Conference of the Centre for Advanced Studies on Collaborative research, 2003, pp. 328–342.

[WZL03] H. Wang, C. Zaniolo, and C. R. Luo, *Atlas: a small but complete sql extension for data mining and data streams*, Proc. of the 29th International Conference on Very Large Data Bases, VLDB Endowment, 2003, pp. 1113–1116.

[WZZ08] F. Wang, C. Zaniolo, and X. Zhou, *Archis: an xml-based approach to transaction-time temporal database systems*, The VLDB Journal **17** (2008), no. 6, 1445–1463.

[XCY06] C. Xie, Z. Chen, and X. Yu, *Sequence outlier detection based on chaos theory and its application on stock market*, Proc. of the 3rd International Conference on Fuzzy Systems and Knowledge Discovery, 2006, pp. 1221–1228.

[XJFD08] Y. Xiang, R. Jin, D. Fuhry, and F. F. Dragan, *Succint summarization of transactional databases: An overlapped hyperrectangle scheme*, Proc. of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008.

[XSMH06] D. Xin, X. Shen, Q. Mei, and J. Han, *Discovering interesting patterns through user's interactive feedback*, Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006.

[YGY02] Y. Yang, X. Guan, and J. You, *Clope: A fast and effective clustering algorithm for transactional data*, Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data mining, 2002, pp. 682–687.

[YHA03] X. Yan, J. Han, and R. Afshar, *Clospan: Mining closed sequential patterns in large datasets*, Proc. of the SIAM International Conference on Data Mining, 2003.

[YHC09] J. Yao, y. Huang, and B. Cui, *Constructing evolutionary taxonomy of collaborative tagging systems*, Proc. of the 18th ACM Conference on Information and Knowledge Management, 2009, pp. 2085–2086.

[YL03] L. Yu and H. Liu, *Feature selection for high-dimensional data: A fast correlation-based filter solution*, Proc. of the 20th International Conference on Machine Leaning, 2003, pp. 856–863.

[YSLS07] Y. Yoon, C.-N. Seon, S. Lee, and J. Seo, *Unsupervised word sense disambiguation for korean through the acyclic weighted digraph using corpus and dictionary*, Information Processing and Management: an International Journal **43** (2007), no. 3, 836–847.

[Zad65] L. A. Zadeh, *Fuzzy sets*, Information and Control **8** (1965), no. 3, 338–353.

[ZB03] Q. Zhao and S. S. Bhowmick, *Sequential pattern mining: A survey*, Tech. Report No. 2003118, CAIS, Nanyang Technological University, Singapore, 2003.

[ZL77] J. Ziv and A. Lempel, *A universal algorithm for sequential data compression*, IEEE Transactions on Information Theory **23** (1977), 337Ű–343.

[ZLX+06] Y. Zhang, X. Lin, J. Xu, F. Korn, and W. Wang, *Space-efficient relative error order sketch over data streams*, Proc. of the 22nd International Conference on Data Engineering, 2006.

[ZP05] M. J. Zaki and M. Peters, *Clicks: Mining subspace clusters in categorical data via k-partite maximal cliques*, Proc. of the 21st International Conference on Data Engineering, 2005, pp. 355–356.

[ZRL96] T. Zhang, R. Ramakrishnan, and M Limy, *Birch: An efficient data clustering method for very large databases*, Proc. of the 15th ACM SIGMOD International Conference on Management of Data, 1996.

[ZZ04] D. Zhang and K. Zhou, *Discovering golden nuggets: data mining in financial application*, IEEE TSMC, Part C: Applications and Reviews **34** (2004), 513–522.

Appendix A

Background Knowledge

This section of the appendix synthesizes in Table A.1 different sources where Background Knowledge can be acquired as taxonomies or ontologies.

Table A.1: Example of published domain specific ontologies and taxonomies

Name	Description
Basic Formal Ontology	A formal upper ontology designed to support scientific research
BioPAX	An ontology for the exchange and interoperability of biological pathway (cellular processes) data.
BMO	An e-Business Model Ontology based on a review of enterprise ontologies and business model literature.
CCO (Cell-Cycle Ontology)	An application ontology that represents the cell cycle.
CContology	An e-business ontology to support online customer complaint management.
CIDOC Conceptual Reference Model	An ontology for cultural heritage.
COSMO	A Foundation Ontology (current version in OWL) that is designed to contain representations of all of the primitive concepts needed to logically specify the meanings of any domain entity. It is intended to serve as a basic ontology that can be used to translate among the representations in other ontologies or databases. It started as a merger of the basic elements of the OpenCyc and SUMO ontologies, and has been supplemented with other ontology elements (types, relations) so as to include representations of all of the words in the Longman dictionary defining vocabulary.

Continued on next page

Table A.1: continued from previous page

Name	Description
Cyc	A large Foundation Ontology for formal representation of the universe of discourse.
Disease Ontology	Designed to facilitate the mapping of diseases and associated conditions to particular medical codes.
DOLCE	A Descriptive Ontology for Linguistic and Cognitive Engineering.
Dublin Core	A simple ontology for documents and publishing.
Foundational, Core and Linguistic Ontologies.	
Foundational Model of Anatomy	An ontology for human anatomy.
Gene Ontology for genomics.	
GUM (Generalized Upper Model)	A linguistically-motivated ontology for mediating between clients systems and natural language technology.
Gellish English dictionary	An ontology that includes a dictionary and taxonomy that includes an upper ontology and a lower ontology that focusses on industrial and business applications in engineering, technology and procurement. See also Gellish as Open Source project on SourceForge.
GOLD	General Ontology for Linguistic Description
IDEAS Group	A formal ontology for enterprise architecture being developed by the Australian, Canadian, UK and U.S. Defence Depts.
Linkbase	A formal representation of the biomedical domain, founded upon Basic Formal Ontology.
LPL	Lawson Pattern Language
NIFSTD	Ontologies from the Neuroscience Information Framework: a modular set of ontologies for the neuroscience domain.
OBO Foundry	A suite of interoperable reference ontologies in biomedicine.
Ontology for Biomedical Investigations	an open access, integrated ontology for the description of biological and clinical investigations.
OMNIBUS Ontology	An ontology of learning, instruction, and instructional design.
Plant Ontology	For plant structures and growth/development stages, etc..
POPE	Purdue Ontology for Pharmaceutical Engineering.

Continued on next page

Table A.1: concluded from previous page

Name	Description
PRO	The Protein Ontology of the Protein Information Resource, Georgetown University.
Program abstraction taxonomy.	
Protein Ontology for proteomics.	
Systems Biology Ontology (SBO)	for computational models in biology.
Suggested Upper Merged Ontology	a formal upper ontology.
SWEET	Semantic Web for Earth and Environmental Terminology.
ThoughtTreasure ontology.	
TIME-ITEM	Topics for Indexing Medical Education.
YATO	Yet Another Top-level Ontology.
WordNet	A lexical reference system.

Appendix B

Algorithms

This section of the appendix gathers utility pseudo-code for the time sequence summarization algorithms proposed throughout this dissertation.

Algorithm 8 PopulateHashFromSubtree

```
1. INPUTS:
   T: XML tree node
   L: List of leaves
   Hash: Hashtable
2. OUTPUT: None
3. LOCAL: N: Son node

4. for all (Son node N of T) do
5.    Hash.add(N,T)
6.    if (N is a leaf) then
7.       L ← L ∪ N
8.    else
9.       PopulateHashFromSubtree(N,L,Hash)
10.   end if
11. end for
```

Algorithm 9 Populate hashtable from XML tree pseudo-code

1. INPUTS:
   g: Generalization level for the descriptive domain D_A (we assume $g \geq 1$)
   T: Taxonomy H_A for descriptive domain D_A in XML tree
2. OUTPUT: $Hash$: Hashtable with precomputed generalizations
3. LOCALS:
   L: Current list of leaves in H_A
   $NewL$: New list of leaves

4. Initialize L
5. Initialize $Hash$ with $Hash.add(T, T)$
6. Call PopulateHashFromSubtree(T,L,$Hash$)
7. $g \leftarrow g - 1$ {// Update number of generalizations already performed}
8. **while** $(L \neq \{T\})$ **do**
9. Initialize $NewL$
10. **for** (int $i = 0; i \leq L.count; i++$) **do**
11. **if** $(NewL.contains(Hast[L[i]]))$ **then**
12. $NewL \leftarrow NewL \cup Hast[L[i]]$
13. **end if**
14. **for** (int $j = 0; j \leq g - 1; j++$) **do**
15. $Hash[L[i]] \leftarrow Hash[Hash[L[i]]]$
16. **end for**
17. **end for**
18. $L \leftarrow NewL$
19. **end while**
20. **return** $Hash$

Résumé de séquences d'événements : théorie et applications
Résumé

Les domaines de la médecine, du web, du commerce ou de la finance génèrent et stockent de grandes masses d'information sous la forme de séquences d'événements. Ces archives représentent des sources d'information très riches pour des analystes avides d'y découvrir des perles de connaissance. Par exemple, les biologistes cherchent à découvrir les facteurs de risque d'une maladie en analysant l'historique des patients, les producteurs de contenu web et les bureaux de marketing examinent les habitudes de consommation des clients et les opérateurs boursiers suivent les évolutions du marché pour mieux l'anticiper. Cependant, ces applications requièrent l'exploration de séquences d'événements très volumineuses, par exemple, la finance génère quotidiennement des millions d'événements, où les événements peuvent être décrits par des termes extraits de riches contenus textuels. La variabilité des descripteurs peut alors être très grande. De ce fait, découvrir des connaissances non triviales à l'aide d'approches classiques de fouille de données dans ces sources d'information prolixes est un problème difficile. Une étude récente montre que les approches classiques de fouille de données peuvent tirer profit de formes condensées de ces données, telles que des résultats d'agrégation ou encore des résumés. La connaissance ainsi extraite est qualifiée de connaissance d'ordre supérieur. À partir de ce constat, nous présentons dans ces travaux le concept de « résumé de séquence d'événements » dont le but est d'amener les applications dépendantes du temps à gagner un facteur d'échelle sur de grandes masses de données. Un résumé s'obtient en transformant une séquence d'événements où les événements sont ordonnés chronologiquement. Chaque événement est précisément décrit par un ensemble fini de descripteurs symboliques. Le résumé produit est alors une séquence d'événements, plus concise que la séquence initiale, et pouvant s'y substituer dans les applications. Nous proposons une première méthode de construction guidée par l'utilisateur, appelée TSaR. Il s'agit d'un processus en trois phases : i) une généralisation, ii) un regroupement et iii) une formation de concepts. TSaR utilise des connaissances de domaine exprimées sous forme de taxonomies pour généraliser les descripteurs d'événements. Une fenêtre temporelle est donnée pour contrôler le processus de regroupement selon la proximité temporelle des événements. Dans un second temps, pour rendre le processus de résumé autonome, c'est-à-dire sans paramétrage, nous proposons une redéfinition du problème de résumé en un nouveau problème de classification. L'originalité de ce problème de classification tient au fait que la fonction objective à optimiser dépend simultanément du contenu des événements et de leur proximité dans le temps. Nous proposons deux algorithmes gloutons appelés G-BUSS et GRASS pour répondre à ce problème. Enfin, nous explorons et analysons l'aptitude des résumés de séquences d'événements à contribuer à l'extraction de motifs séquentiels d'ordre supérieur. Nous analysons les caractéristiques des motifs fréquents extraits des résumés et proposons une méthodologie qui s'appuie sur ces motifs pour en découvrir d'autres, à granularité plus fine. Nous évaluons et validons nos approches de résumé et notre méthodologie par un ensemble d'expériences sur un jeu de données réelles extraites des archives d'actualités financières produites par *Reuters*.

Mots-clés : Séquence d'événements, résumé, temps, données catégorielles, fouille de données, classification, motifs séquentiels

Time Sequence Summarization : Theory and Applications
Abtract

Domains such as medicine, the WWW, business or finance generate and store on a daily basis massive amounts of data. This data is represented as a collection of time sequences of events where each event is described as a set of descriptors taken from various descriptive domains and associated to a time of occurrence. These archives represent valuable sources of insight for analysts to browse, analyze and discover golden nuggets of knowledge. For instance, biologists could discover disease risk factors by analyzing patient history, web content producers and marketing people are interested in profiling client behaviors, traders investigate financial data for understanding global trends or anticipating market moves. However, these applications require mining massive sequences, e.g., finance can generate millions of events daily, where the variability of event descriptors could be very high, since descriptors could be extracted from textual contents. For these reasons, discovering golden nuggets of knowledge for such domains with conventional data mining techniques is a challenging task. Recent studies show that data mining methods might need to operate on derived forms of the data, including aggregate values, previous mining results or summaries. Knowledge extracted in such a way is called Higher Order Knowledge. In this thesis work, we propose to address this challenge and we define the concept of "Time sequence summarization" whose purpose is to support chronology-dependent applications to scale on very large data sources. Time sequence summarization uses the content and temporal information of events to generate a more concise, yet informative enough, time sequence that can seamlessly be substituted for the original time sequence in the desired application. We propose a user-oriented approach called TSaR built on a 3-step process: generalization, grouping and concept formation. TSaR uses background knowledge in the form of taxonomies to represent event descriptors at a higher level of abstraction. A temporal parameter controls the grouping process and only allows events close on the timeline to be gathered. Also, we propose to make the time sequence summarization process parameter-free. For this purpose, we reformulate the summarization problem into a novel clustering problem. The originality of this clustering problem relies on the specificity of the objective function to optimize. Indeed, the objective function takes into account both the content and the proximity of events on the timeline. We present two greedy approaches called G-BUSS and GRASS to build a solution to this problem. Finally, we explore and analyze how time sequence summaries contribute to discovering Higher Order Knowledge. We analytically characterize the higher order patterns discovered from summaries and devise a methodology that uses the patterns discovered to uncover even more refined patterns. We evaluate and validate our summarization algorithms and our methodology by an extensive set of experiments on real world data extracted from Reuters's financial news archives.

Keywords : Time sequence, event sequence, summarization, categorical data, data mining, clustering, sequential pattern mining

Discipline : Informatique

N° :