Signatures : detecting and characterizing complex recurrent behavior in sequential data

Clément Gautrais

UNIVERSITE BRETAGNE LOIRE MATHSTIC

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITE DE RENNES 1
COMUE **U**NIVERSITE **B**RETAGNE **L**OIRE

Ecole Doctorale N°601
Mathèmatique et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par

« Clément GAUTRAIS »

« Signatures: Detecting and Characterizing Complex Recurrent Behavior in Sequential Data »

««»

Thèse présentée et soutenue à RENNES , le 16 Octobre 2018
Unité de recherche : UMR 6074 IRISA

Rapporteurs avant soutenance :

Bruno CRÉMILLEUX Professeur à l'Université de Caen
Jilles VREEKEN Senior Researcher at Max Planck Institute for Informatics

Composition du jury :

Examinateurs : Céline ROUVEIROL Professeur à l'Université Paris 13
Mehdi KAYTOUE Maitre de conférences à l'INSA de Lyon
Alexandre TERMIER Professeur à l'Université de Rennes 1
Peggy CELLIER Maitre de conférences à l'INSA de Rennes
René QUINIOU Chercheur Inria Rennes
Thomas GUYET Maitre de conférences à Agrocampus Ouest

Dir. de thèse : Alexandre TERMIER Professeur à l'Université de Rennes 1

Co-dir. de thèse :

Invité(s)

REMERCIEMENTS

Je souhaite tout d'abord remercier mes encadrants de thèse : Alexandre, Peggy, Thomas et René. J'ai énormément apprécié travailler avec vous pendant ces trois années. Chacune de nos réunions était pour moi l'occasion d'avoir de nombreux et précieux conseils sur l'apprentissage de la recherche. Vos retours enthousiastes sur les différentes visualisations illustrant le travail effectué m'ont également encouragé à chercher la façon la plus adaptée de présenter mes résultats. Vous m'avez aussi donné la liberté de découvrir différents domaines (notamment le traitement de la langue naturelle) que je n'avais pas initialement envisagés. Merci donc de m'avoir accompagné pendant cette thèse, je suis fier et heureux du travail que nous avons réalisé.

Je remercie également les membres du jury d'avoir accepté d'évaluer ma thèse. Je suis honoré du temps qu'ils ont consacré à lire et juger ma thèse.

Au quotidien, j'ai eu la chance de travailler dans une excellente atmosphère, et c'est pour cela que je remercie toute l'équipe LACODAM. Les différentes pauses café et pauses déjeuner se sont toujours déroulées dans la bonne humeur, et j'ai pu apprécier cette jovialité tout au long de la thèse. L'environnement de travail offert par l'IRISA a également été déterminant, que ce soit à travers la cafétéria et son personnel toujours de bonne humeur, l'accès au restaurant de Supelec d'une qualité inégalée ou les opportunités de pouvoir jouer au football avec de nouveaux collègues.

C'était également un plaisir d'entendre occasionnellement les rires de différent(e)s collègues pour enjoliver la journée. Différentes personnes rencontrées sur des périodes plus courtes ont aussi compté dans ma thèse : Benjamin m'a notamment donné l'occasion de développer mon sens critique et Ahmed m'a fait redécouvrir les joies du tennis de table. Je remercie également mes collègues de bureau, notamment Yann et Maël, avec qui la collecte de données de la pause déjeuner était toujours appréciée.

Pendant ma thèse, j'ai également pu compter sur le soutien de mes différents amis. Les soirées colorées à la coloc, les nouvels ans aux roupettes, les retrouvailles insaïennes et bien d'autres encore ont largement contribué à me donner l'énergie nécessaire à la réalisation de cette thèse.

Je tiens à remercier plus particulièrement Arnaud ainsi que les différents colocataires que j'ai eus pendant ces trois années. Leur présence et soutien quotidien ont été plus qu'appréciés.

Enfin, je souhaite remercier ma famille, et plus particulièrement mes parents, Hubert et Sylvie, mon frère Jérémy et ma soeur Marie, d'avoir toujours été à mes côtés. Leur support est une force majeure sur laquelle j'ai pu m'appuyer tout au long de mon parcours.

J'ai également une pensée pour mon grand-père Léon, qui n'a pas pu m'accompagner jusqu'au bout de la thèse. Je lui dédie ce manuscrit.

RÉSUMÉ EN FRANÇAIS

De plus en plus de données sont collectées à l'heure actuelle et valoriser ces données en y extrayant des connaissances est capital. Parmi cette masse de données, de nombreuses sont issues d'activités humaines : historique de navigation, commentaires sur des produits, résultats sportifs, consommation d'énergie, etc.

Un des axes d'analyse de ces données consiste à prédire des évènements futurs : prédire le prochain achat d'un consommateur, prédire le vainqueur d'un match, prédire la demande en électricité pour la semaine suivante, etc. Pour réaliser ces prédictions, différentes techniques peuvent être utilisées, de la simple régression linéaire aux complexes réseaux de neurones profonds. Ces modèles apprennent des associations entre les variables d'entrées (consommation d'énergie sur l'année précédente par exemple) et la variable que l'on souhaite prédire (par exemple, la demande d'électricité pour la semaine suivante). Les motifs correspondent à des associations entre différentes variables : entre les variables d'entrée et la sortie à prédire, ou entre différentes variables d'entrée. Bien que ces modèles soient capables d'identifier des motifs complexes, ils ne sont généralement pas capables d'expliquer ces motifs à l'analyste humain. Ainsi, ces modèles peuvent être très performants sur des tâches complexes telles que la reconnaissance d'objets ou la reconnaissance de la parole, tout en ayant beaucoup de difficultés à transmettre les motifs appris à l'utilisateur. Avec l'introduction de nouvelles lois, telles que le Règlement Général sur la Protection des Données (RGPD)[1], comprendre et expliquer les motifs qui ont servi à effectuer une prédiction est une obligation légale.

Pour que les motifs identifiés soient compréhensibles par un utilisateur humain, d'autres techniques ont été développées. Ces techniques ne se focalisent pas uniquement sur la prédiction d'évènements futurs, mais se concentrent principalement sur l'apprentissage de motifs qui peuvent être compris par un analyste. Du fait que ces motifs doivent être compréhensibles par un analyste humain, ces techniques généralement demandent à l'analyste de définir le type de motifs qu'elle ou il souhaite trouver.

Database $\mathscr{D}$		
Client Id	Date	Items
1	3 Juin	Comté, Pinot noir, Leonidas Maxi Amandes
1	6 Juin	Samsung Galaxy S9, Carte 128GB, Lindt Chocolat, Chimay Bleue
1	10 Juin	Comté, Samsung Galaxy S9 pour les Nuls, Pinot noir
1	12 Juin	Cailler – Noisettes 500g, Jus d'Orange, Protection écran
1	17 Juin	Chimay Bleue, Comté, Pinot noir, Lindt Chocolat
1	24 Juin	Samsung Galaxy S9: les bases, Pinot noir
1	25 Juin	Comté, Lindt Chocolat
2	29 Mars	Comté, Pinot noir, Pringles Barbecue
2	23 Avril	Samsung Galaxy S9, Royal Canin Felin Nutrition
2	12 Mai	Camembert, Coat-albret Cidre
2	13 Mai	Samsung Galaxy S9 pour les Nuls, Royal Canin Felin Nutrition
2	2 Juillet	Royal Canin Felin Nutrition, STAR WARS BB-8 Jouets Felin
3	24 Janvier	Galettes, Beurre avec cristaux de sel de Guérande
3	25 Juin	Samsung Galaxy S9, Choux, Bo Bun Vegetarien, Comté
3	26 Juin	Samsung Galaxy S9: les bases, Eau Minérale Volvic

Table 1: Historique d'achats de 3 consommateurs de supermarché.

Différence avec les Méthodes Existantes

Un des types de motifs les plus simples sont les *itemsets fréquent* [AS+94]. Les itemsets fréquents ont été initialement introduits pour analyser les achats de clients de supermarchés : les itemsets fréquents correspondent aux ensembles de produits fréquemment achetés dans le même panier. Par exemple, si l'on regarde l'historique d'achats présenté en table 1, *Comté* et *Pinot noir* sont achetés 4 fois ensemble : le 3 Juin, le 10 Juin et le 17 Juin par le client 1, et le 29 Mars par le client 2. Si l'analyste considère qu'acheter ces produits 4 fois ensemble est équivalent à dire qu'ils ont achetés fréquemment ensemble, alors l'itemset *{Comté, Pinot noir}* est considéré comme fréquent. Trouver les itemsets fréquents sur un grand nombre de consommateurs permet à l'analyste d'apprendre quels produits sont fréquemment achetés ensemble. Il est important de noter que l'analyste a un rôle clé dans ce processus d'extraction de connaissances. En effet, elle ou il doit d'abord décider de chercher les produits qui sont achetés dans le même panier. Ensuite, cet analyste doit définir le nombre minimal d'achats pour qu'un itemset soit considéré comme étant fréquent.

Identifier ces corrélations d'achats au sein d'un même panier est utile pour les supermarchés. Par exemple, lorsque l'on effectue un achat sur un site tel qu'Amazon, ce site nous montre les

1. https://eur-lex.europa.eu/legal-content/FR/TXT/?uri=CELEX:32016R0679

produits qui sont fréquemment achetés avec notre achat courant. Cette fonctionnalité du site web est illustrée en figure 1. Du point de vue de la fouille de motifs, Amazon montre à l'utilisateur un itemset fréquent de taille 3, contenant l'achat courant. Lorsque Agrawal et Srikant ont introduit l'algorithme Apriori [AS+94] pour trouver rapidement les itemsets fréquents, ils ont permis aux supermarchés d'utiliser ces motifs afin d'augmenter leurs ventes. De nombreux travaux ont ensuite amélioré le processus de fouille de motifs fréquents, pour analyser de grandes bases de données en un temps raisonnable.

Figure 1: Capture d'écran du site web Amazon. Lorsque l'on cherche à acheter un téléphone portable, le site web propose d'acheter d'autres produits, qui sont fréquemment achetés avec ce téléphone.

Néanmoins, les itemsets fréquents ne permettent pas de prendre en compte les corrélations d'achats entre différents paniers. En effet, dans la table 1, on note que les clients 1 et 2 ont acheté le *Samsung Galaxy S9* puis le livre *Samsung Galaxy S9 pour les Nuls* et que les clients 1 et 3 ont acheté le *Samsung Galaxy S9* puis le livre *Samsung Galaxy S9: les bases*. Pour identifier ces liens à travers différents paniers, un nouveau type de motifs doit être défini. Agrawal et al. [AS95] ont ainsi défini les motifs *séquentiels fréquents* pour identifier les corrélations d'achats entre paniers. Ces nouveaux motifs sont capables d'identifier le fait que le livre *Samsung Galaxy S9 pour les Nuls* est souvent acheté après le *Samsung Galaxy S9*. D'un point de vue applicatif, ces motifs séquentiels fréquents sont utiles pour recommander de futurs achats. Par exemple, Amazon envoie fréquemment des courriels promotionnels contenant des recommandations d'achats personnalisées. Une façon de déterminer ces recommandations est de regarder les achats précédents d'un consommateur et d'identifier les motifs séquentiels fréquents où ces achats apparaissent en premier. Les produits fréquemment achetés après ces précédents achats

devraient être de bonnes recommandations.

En poursuivant l'exemple de la recommandation d'achats futurs, on peut se rendre compte que prendre en compte uniquement la séquentialité des achats n'est pas suffisant. En effet, recommander l'achat du livre *Samsung Galaxy S9 pour les Nuls* longtemps après l'achat du téléphone *Samsung Galaxy S9* ne semble pas être une bonne idée. Pour cela, les *séquences temporelles annotées* [GNP06] ont été définies. Elles permettent de prendre en compte la temporalité, c'est-à-dire l'information du temps écoulé entre deux achats. Un autre type de motifs prenant en compte la temporalité sont les motifs *périodiques fréquents* [MH01]. L'objectif de ces motifs est d'identifier les itemsets qui apparaissent périodiquement. Par exemple, l'itemset *{Comté, Pinot noir}* a un rythme d'achat hebdomadaire. Apprendre ces habitudes périodiques permet par exemple d'anticiper les futurs achats.

Grâce à ces différents types de motifs, l'analyste a maintenant la possibilité d'identifier des corrélations entre items au sein d'un même panier, ou entre items présents dans différents paniers. Ces deux possibilités permettent de résoudre des problématiques différentes : les itemsets fréquents sont utiles pour suggérer des achats dans le même panier, tandis que les motifs séquentiels fréquents sont utiles pour suggérer des achats dans de futurs paniers.

Dans le domaine du supermarché, une question intéressante à laquelle nous souhaitons répondre est l'identification des *produits favoris* d'un consommateur, ainsi que ses périodes de réapprovisionnement. Intuitivement, les produits favoris d'un consommateur correspondent aux produits que ce consommateur souhaite avoir tout le temps chez lui. Ces produits doivent donc être achetés plusieurs fois sur tout l'historique d'achat. Par exemple, dans la table 1, le *Comté* et le *Pinot noir* peuvent être considérés comme des produits favoris de ce consommateur. Dans une moindre mesure, *Chimay Bleue* et *Lindt Chocolat* peuvent également être considérés comme des produits favoris de ce consommateur. À partir de cet exemple, on voit qu'il est donc difficile de définir précisément ce qu'est un produit favori. Pour prendre en compte cette difficulté, nous disons donc qu'un produit favori doit être acheté plusieurs fois durant tout l'historique d'achats.

Pour identifier ces produits favoris, nous introduisons un nouveau motif appelé *signature*. La signature segmente l'historique d'achats d'un consommateur, de façon à maximiser la taille de l'ensemble de produits qui apparait dans tous les segments. La maximisation de la taille de l'ensemble de produits s'explique par le fait que l'on souhaite identifier le plus de produits favoris. Un exemple de segmentation est présenté en table 2, dans le cas où un produit favori doit être acheté au moins 3 fois. A partir de l'historique d'achat d'un consommateur et du nombre d'achats nécessaire pour un produit favori, la signature donne deux types d'informations : 1) les

produits favoris du consommateur et 2) les périodes de réapprovisionnement, qui correspondent aux segments. Par exemple, en table 2, les seuls produits qui apparaissent dans tous les segments sont *Comté*, *Pinot noir* et *Lindt Chocolat*. Ils correspondent ainsi aux produits favoris du consommateur. Chaque période de réapprovisionnement est délimitée par une ligne horizontale. Par exemple, la dernière période de réapprovisionnement se situe lors des 24 et 25 Juin.

Database $\mathscr{D}_1$	
Time	Items
3 Juin	**Comté**, **Pinot noir**, Leonidas Maxi Amandes
6 Juin	Samsung Galaxy S9, Carte Mémoire 128GB, **Lindt Chocolat**, Chimay Bleue
10 Juin	**Comté**, Samsung Galaxy S9 pour les Nuls, **Pinot noir**
12 Juin	Cailler – Noisette entières 500g, Jus d'Orange Tropicana, Protection écran
17 Juin	Chimay Bleue, **Comté**, **Pinot noir**, **Lindt Chocolat**
24 Juin	Samsung Galaxy S9: les bases, **Pinot noir**
25 Juin	**Comté**, **Lindt Chocolat**

Table 2: Segmentation réalisée par la signature, lorsqu'un produit favori doit être acheté au minimum 3 fois. Les produits favoris sont *Comté*, *Pinot noir* et *Lindt Chocolat*. Chaque période de réapprovisionnement est délimitée par une ligne horizontale. Le client analysé est le client 1 de la table 1.

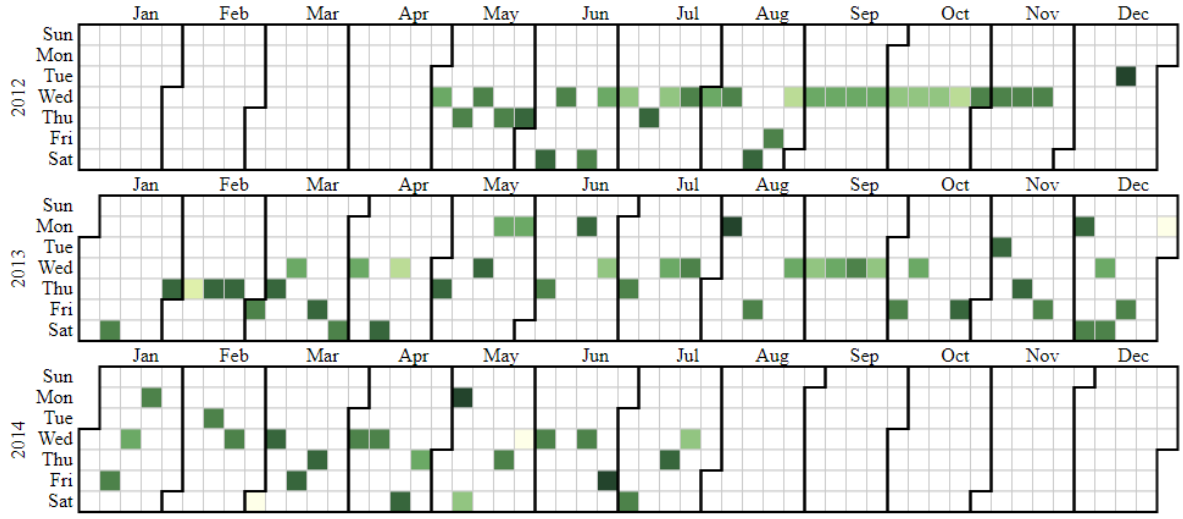

Figure 2: Comportement d'achat d'un consommateur de supermarché. Les rectangles verts représentent les achats du consommateur. Plus le rectangle est vert, plus le nombre de produits achetés est important.

La signature est capable de s'adapter aux difficultés liées à l'identification des produits favoris. Tout d'abord, les consommateurs ont des produits favoris avec différents taux de réappro-

visionnement. En effet, l'achat de produits tels que la lessive s'effectue à un rythme lent, tandis que l'achat de produits comme les oeufs ou le lait s'effectue à un rythme plus rapide. Néanmoins, ces deux types de produits peuvent être considérés comme des produits favoris. Cela veut dire que les produits favoris peuvent être achetés sur des échelles de temps différentes, et donc pas nécessairement dans la même transaction, ou dans le même ordre. La table 2 montre que la signature est capable d'identifier les produits favoris, même si ceux-ci sont achetés dans différentes transactions, sans ordre précis. Les autres motifs décrits précédemment ne sont pas capables de s'adapter à cette particularité des produits favoris.

Une deuxième difficulté vient du fait que les consommateurs de supermarché n'ont pas nécessairement un rythme d'achat régulier. Pour illustrer cela, les achats d'un consommateur au cours du temps sont représentés en figure 2. De mai à août 2012, ce consommateur effectue environ un panier par semaine, sans avoir de régularité sur le jour de la semaine correspondant à ses achats. Ensuite, de septembre à novembre 2012, ce même consommateur effectue ses achats tous les mercredis. De décembre 2012 à janvier 2013, ce consommateur effectue seulement 3 achats. La suite de son historique d'achats est également assez irrégulier, mélangeant une période d'achats réguliers en Septembre 2013 avec des achats plus variables le reste de l'année. Pour prendre en compte ce type de comportement irrégulier, les *épisodes* ont été définis [MTV97], mais l'irrégularité de chaque consommateur doit rester dans une limite fixée par l'utilisateur, afin de limiter le temps de calcul des épisodes. Fixer cette limite d'irrégularité pour chaque consommateur à l'avance n'est pas facile, ce qui rend l'utilisation des épisodes peu adaptée pour identifier les produits favoris. Les signatures sont en revanche faites pour gérer ces irrégularités, car elles ne mettent aucune contrainte sur la taille des périodes de réapprovisionnement.

Enfin, une dernière difficulté est liée au fait que les méthodes de fouilles de motifs fréquents renvoient un très grand nombre de résultats. Ce nombre de résultats peut même rapidement dépasser le nombre d'achats, ce qui rend l'analyse de ces résultats encore plus complexe que de simplement regarder les achats d'un consommateur. Comme la signature ne renvoie qu'un seul ensemble de produits et ses périodes de réapprovisionnement, ces résultats peuvent être rapidement analysés par des experts.

Contributions

Cette thèse apporte quatre contributions principales.

- Cette thèse introduit un nouveau motif appelé *signature*, afin d'identifier les items récurrents d'une séquence. Ce motif se différencie des approches existantes car il identifie les items récurrents qui 1) peuvent apparaître à différentes échelles temporelles, 2) peuvent avoir des occurrences irrégulières et 3) peuvent être rapidement compris par des analystes.
- Cette thèse introduit également différents algorithmes, basés sur le domaine de la fouille de motifs et la segmentation de séquences, afin de calculer rapidement la signature d'une séquence.
- Cette thèse propose également deux extensions de la signature. La *sky-signature* permet de résumer les signatures associées à différentes échelles temporelles. La *signature MDL* permet de trouver la signature qui correspond le mieux aux données.
- L'applicabilité de la signature et de ses deux extensions est illustrée sur deux cas d'usages principaux : l'analyse de consommateurs de supermarché et l'analyse de discours politiques.

La première contribution de cette thèse est donc la définition d'un nouveau motif appelé *signature*. L'objectif de la signature est de trouver les items récurrents dans une séquence. La définition de la signature est initialement liée à l'analyse de consommateurs de supermarchés. Dans ce contexte, la signature permet de retrouver les produits favoris d'un consommateur à partir de son historique d'achats.

Trouver la signature d'une séquence est initialement défini comme un problème appartenant à la fouille de motifs. Une des contributions de cette thèse est d'avoir fait le lien avec le domaine de la segmentation de séquences. Ce lien a permis de définir deux catégories d'algorithmes trouvant des signatures. Ces algorithmes sont complémentaires et permettent de trouver rapidement une signature. Les algorithmes basés sur des techniques de fouille de données sont généralement efficaces lorsque le nombre d'occurrences des items récurrents est élevé. A l'inverse, les algorithmes basés sur les techniques utilisées dans la segmentation de séquence sont généralement efficaces lorsque le nombre d'occurrences est faible.

Après avoir introduit ces différents algorithmes permettant de trouver rapidement la signature d'une séquence, les signatures ont été utilisées pour analyser des consommateurs de supermarché. Ces analyses ont permis de montrer que la signature est capable d'identifier les

régularités trouvées par d'autres types de motifs (motifs périodiques par exemple), mais qu'elle détecte également de nouvelles régularités. Pour valider les signatures associées aux consommateurs de supermarché, les caractéristiques temporelles des signatures ont été évaluées. Par exemple, nous avons montré que lorsque la signature identifie un rythme hebdomadaire chez un client, celui-ci a tendance à avoir plus de produits favoris dans sa signature. Etant donné que les clients effectuant toujours leurs courses le même jour de la semaine sont plus susceptibles d'être des clients réguliers du magasin, il semble normal que ces clients aient en moyenne plus de produits favoris que les autres clients moins réguliers.

Les signatures ont également été utilisées pour détecter et caractériser l'attrition de certains consommateurs (le fait qu'ils aillent chez un concurrent). Une méthode simple basée sur l'évolution de la signature d'un consommateur dans le temps a montré que les signatures permettaient de détecter un client en attrition quelques mois avant son départ définitif. L'autre avantage des signatures est qu'elles permettent de cibler les produits sur lesquels les consommateurs sont en attrition.

Ensuite, les *sky-signatures* ont été introduites. Cette extension de la signature se base sur une technique de sélection de motifs appelée *sky-pattern* [Sou+11]. Le but de la sky-signature est de résumer les signatures calculées à toutes les échelles de temps possibles. De nouveau, le fait que ce modèle soit lié à la fouille de motifs et à la segmentation de séquence a permis de développer un algorithme efficace pour trouver la sky-signature d'une séquence. La sky-signature a été utilisée pour étudier les discours de campagne de H. Clinton et D. Trump pendant la campagne présidentielle de 2016. La sky-signature a identifié les thèmes de campagne principaux de chaque candidat, ainsi que la dynamique temporelle de leurs campagnes. La sky-signature a notamment identifié un changement de stratégie vers la fin de la campagne de D. Trump. Le fait d'appliquer la sky-signature sur des données issues de discours a donc aussi permis de valider l'utilisation de la sky-signature sur des données différentes de celles issues des achats de consommateurs.

Enfin, la deuxième extension de la signature, basée sur le principe de longueur de description minimale (*MDL* pour *Minimum Description Length* en anglais) [Ris78], est définie dans cette thèse. Cette extension permet de trouver la signature qui correspond le mieux à la séquence étudiée. L'utilisation du principe MDL a notamment nécessité la définition de nouveaux algorithmes basés sur des techniques classiques d'optimisation, telles que la recherche en faisceau [LK12]. Cette extension de la signature a de nouveau été utilisée pour analyser des consommateurs de supermarché. Une analyse qualitative des résultats trouvés a permis de voir que les signatures trouvées par cette extension basée sur MDL correspondent effectivement à des

comportements intéressants pour les analystes.

Parmi les différentes perspectives de cette thèse, l'intégration de la dimension spatiale dans la signature est un axe intéressant. Une autre perspective serait d'appliquer la signature sur d'autres cas d'applications, comme la recommandation de produits ou l'optimisation de stocks de produits.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

It comes as no surprise that more and more data are being collected and that deriving valuable knowledge from these data is of paramount importance. Many of these data stem from human activity: browsing history, product reviews, sport results, energy consumption, etc.

One type of valuable action that one can derive from these data is to predict future events: predict the next purchase of a customer, predict the winner of a game, predict the electricity demand of next week, etc. Many techniques exist to perform these predictions, from the simple linear regression model to the complex deep neural network model. These models learn associations between input variables (energy consumption on the last year for example) and an output that one wishes to predict (the electricity demand of next week for example). Patterns are associations between different variables: between input and output or between different input variables. While these models are able to identify complex patterns to make accurate predictions, they are not able to explain these patterns to the (human) analyst. Hence these models can be very accurate on complex tasks such as object or speech recognition, but they have difficulties to convey the patterns they learned to the user. With the introduction of new regulations, such as the General Data Protection Regulation (GDPR)[1], understanding the patterns that lead to a prediction is legally required and thus becomes essential.

To communicate understandable patterns to the user, other techniques have been developed. These techniques do not necessarily focus on making predictions, but instead, focus on learning patterns that can be understood by analysts. Because patterns have to be understandable, these techniques usually require that the analyst defines the type of patterns she/he is looking for.

One of the most simple type of patterns are *frequent itemsets* [AS+94]. Frequent itemsets were initially introduced to analyze retail customers purchases: frequent itemsets correspond to sets of products that are frequently purchased in the same basket (also called transaction). For example, if we look at the purchase histories presented in Table 1.1, *Comte Cheese* and *Pinot noir* are purchased together 4 times: on June 3, June 10 and June 17 for customer 1 and on March 29 for customer 2. If the retail analyst considers that purchasing these items 4 times

1. https://www.eugdpr.org/

Database $\mathscr{D}$		
Customer Id	Time	Items
1	June 3	Comte Cheese, Pinot noir, Leonidas Maxi Tablet Almond
1	June 6	Samsung Galaxy S9, Card 128GB, Lindt Chocolate, Chimay Blue
1	June 10	Comte Cheese, Samsung Galaxy S9 for Dummies, Pinot noir
1	June 12	Cailler – Whole Hazelnuts 500g, Orange Juice, Screen Protector
1	June 17	Chimay Blue, Comte Cheese, Pinot noir, Lindt Chocolate
1	June 24	Samsung Galaxy S9: Learning the Essentials, Pinot noir
1	June 25	Comte Cheese, Lindt Chocolate
2	March 29	Comte Cheese, Pinot noir, Pringles Barbecue
2	April 23	Samsung Galaxy S9, Royal Canin Feline Health Nutrition
2	May 12	Camembert, Coat-albret Cider
2	May 13	Samsung Galaxy S9 for Dummies, Royal Canin Feline Nutrition
2	July 2	Royal Canin Feline Health Nutrition, STAR WARS BB-8 Cat Toys
3	January 24	Buckwheat Pancakes, Butter with Guérande Salt Crystals
3	June 25	Samsung Galaxy S9, Cabbage, Vegetarian Bo Bun, Comte Cheese
3	June 26	Samsung Galaxy S9: Learning the Essentials, Volvic Mineral Water

Table 1.1: Purchase history of 3 retail customers.

means that they are frequently purchased together, the itemset *{Comte Cheese, Pinot noir}* can be considered frequent. By finding frequent itemsets across many customers, the retail analyst learns which products are frequently bought together. It should be emphasized that the analyst has a key role in this process of learning from retail data. First, the analyst decides to look for items that are purchased together. Then, the analyst defines the minimal number of purchases for an itemset to be considered frequent.

Identifying such correlation in purchases is of practical interest for sellers. For example, when purchasing a product from Amazon, the website shows the items that are frequently bought with this product. Figure 1.1 shows an example of this feature. From a pattern mining point of view, Amazon is outputting a frequent itemset of size 3, containing the current product. When Agrawal and Srikant introduced the Apriori algorithm [AS+94] as an efficient solution to mine such frequent itemsets, they enabled vendors to use frequent itemsets to boost their sales. Many work have then improved frequent itemsets mining, to process larger databases in a reasonable time.

However, one can see that frequent itemsets are not able to capture correlations between purchases made on different transactions. For example, in Table 1.1, it seems that purchasing the books *Samsung Galaxy S9 for Dummies* or *Samsung Galaxy S9: Learning the Essentials* is correlated with purchasing a *Samsung Galaxy S9* smartphone. To capture correlations across

several transactions, one needs to introduce a new type of patterns. Agrawal et al. [AS95] introduced *frequent sequential patterns* to find correlations across transactions. These patterns would be able to capture the fact that *Samsung Galaxy S9 for Dummies* is often bought after buying a *Samsung Galaxy S9* smartphone. From a practical point of view, such frequent sequential patterns are useful to recommend future purchases. For example, Amazon frequently sends promotional emails with tailored product recommendations. One way to make relevant recommendations is to take a previous purchase of the customer at hand and look at the most frequent sequential pattern containing this product as the first item. Items frequently purchased after this product are then relevant personalized recommendations.

Figure 1.1: Screenshot of the Amazon website. When attempting to purchase a smartphone, the website shows the items that are frequently purchased with this smartphone.

From this recommendation example, one can suggest that using sequentiality only is not sufficient to produce meaningful recommendations. Indeed, recommending *Samsung Galaxy S9 for Dummies* when *Samsung Galaxy S9* has been purchased many months ago might not be a good idea. *Temporal annotated sequences* [GNP06] have been introduced to take temporality into account, that is the information of time delay between purchases. Another type of patterns that take into account temporality are *frequent periodic patterns* [MH01]. The goal of these patterns is to capture itemsets that occur periodically. For example, the itemset *{Comte Cheese, Pinot noir}* occurs weekly in Table 1.1. Learning these periodic habits can then help to anticipate customer purchases.

Provided with these new kinds of patterns, the analyst now has a choice to either look for correlations between items within a transaction, or correlations between items across transactions. Both methods are useful to tackle different challenges: frequent itemsets are useful to

suggest purchases in the current basket, while frequent sequential patterns are useful to suggest purchases in future baskets.

Within this retail context, one interesting question we would like to answer is finding the *favorite* products of a customer. We also want to identify the *replenishment periods* of these favorite products. A replenishment period correspond to a time period where all favorite products are purchased at least once. Intuitively, the favorite products of a customer correspond to the products that this customer always want to have at home. Therefore, these products are bought several times throughout the whole purchase history of the customer. We say that these products are *recurrent*. For example, in Table 1.1 one could identify *Comte Cheese* and *Pinot noir* as part of the favorite products. *Chimay Blue* and *Lindt Chocolate* could also be considered as part of the favorite products. Looking at the example in Table 1.1, it is already noticeable that precisely defining a favorite product is difficult. To reflect this difficulty to precisely define what a favorite product is, we simply say that the favorite products have to be bought several times and regularly over the whole purchase sequence of one customer.

To perform this identification of the favorite products, we introduce a new pattern called *signature*. The signature segments the purchase history of a customer, so that the set of products that occurs in every segment is the largest. We look for the largest set of products, as we are interested in finding most favorite products. Such segmentation is presented in Table 1.2, under the constraint that favorite products have to be bought at least 3 times. From the purchase history of a customer and a number of purchases, the signature yields 2 types of information: 1) the favorite products, that are in every segment and 2) the replenishment periods, that correspond to the segments. For example, in Table 1.2, the only products that occur in every segment are *Comte Cheese*, *Pinot noir* and *Lindt Chocolate*, hence they are the favorite products of this customer. Each replenishment period is delimited by a horizontal bar. For example, the last replenishment period happens on June 24 and 25.

The signature model is able to cope with some of the difficulties to identify favorite products. First, customers have favorite products with different replenishment rates. Indeed, retail customers usually buy some products, such as washing powder, at a slow rate, while purchasing fresh products such as eggs or fruits at a faster rate. Nevertheless, both types of products can be considered as part of the favorite ones. This means that favorite products can be purchased in different transactions, without any particular order. Table 1.2 shows that the signature is able to capture products that are bought at different rates, and not necessarily in the same transaction. The patterns described above cannot adapt to this particularity.

Then, another difficulty is that the purchase behavior of a customer is usually not regular.

Database $\mathscr{D}_1$	
Time	Items
June 3	**Comte Cheese**, **Pinot noir**, Leonidas Maxi Tablet Almond
June 6	Samsung Galaxy S9, Card 128GB, **Lindt Chocolate**, Chimay Blue
June 10	**Comte Cheese**, Samsung Galaxy S9 for Dummies, **Pinot noir**
June 12	Cailler – Whole Hazelnuts 500g, Orange Juice, Screen Protector
June 17	Chimay Blue, **Comte Cheese**, **Pinot noir**, **Lindt Chocolate**
June 24	Samsung Galaxy S9: Learning the Essentials, **Pinot noir**
June 25	**Comte Cheese**, **Lindt Chocolate**

Table 1.2: Segmentation performed by the signature, when favorite products have to be bought 3 times. The favorite products are *Comte Cheese*, *Pinot noir* and *Lindt Chocolate*, and each replenishment period is delimited by a horizontal bar. The signature is computed on customer 1 from Table 1.1.

To illustrate this, a purchase behavior of a real retail customer is plotted in Figure 1.2. From May 2012 to August 2012, this customer goes to the store about once a week on different week days. Then, from September to November, this customer only goes to the store every Wednesday. Then in December 2012 and January 2013, this customer goes only 3 times to the store. The rest of this customer purchase history follows a somewhat erratic behavior, with a stable period in September 2013. To cope with erratic behavior in purchase history, episodes have been defined [MTV97], but the erratic behavior is required to stay within a user specified time range to limit the runtime of the mining process. As bounding the erratic behavior of retail customers is difficult, episodes are not really adapted to tackle this challenge. Signatures are able to cope with this difficulty, as there is no constraint on the size of the replenishment periods.

Finally, pattern mining algorithms typically output a very large number of frequent patterns, that can largely exceed the number of purchases. Analyzing these patterns then becomes more costly than looking at the raw data. As the signature outputs a set of products and its associated replenishment periods, these results can be quickly analyzed by the retailer's experts.

1.1 Contributions

The contributions of this thesis are the following.

- This thesis introduces the *signature* model to find recurrent items in a sequence. The particularity of this model is that these recurrent items can 1) occur at different time scales, 2) have non regular occurrences and 3) be easily understood by practitioners.

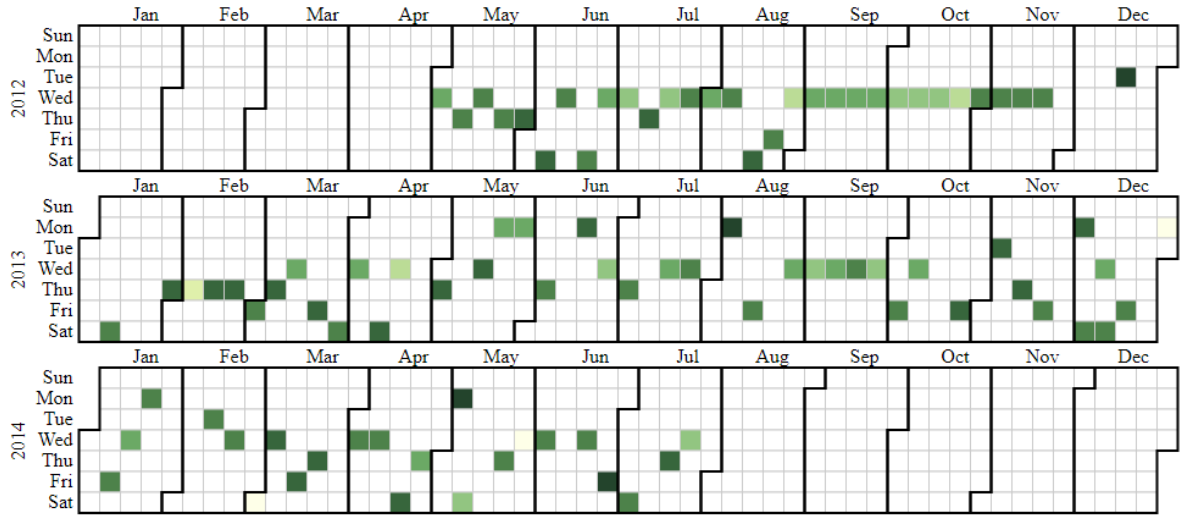

Figure 1.2: Purchase behavior of a real retail customer. Green rectangles correspond to purchases. The more green the rectangle is, the more products were bought during that visit to the store.

- Algorithms, stemming from pattern mining and sequence segmentation fields, are proposed to efficiently mine a signature.
- Two extensions of the signature model are proposed. The *sky-signature* extends the signature by summarizing signatures that occur at different time scales. The *MDL based signature* extends the signature by mining the signature that best fits the data.
- The practical use of the signature model is illustrated on two main use cases: the analysis of retail customers (the motivating use case) and the analysis of political speeches.

1.2 Thesis Outline

The remaining of this thesis is organized as follows.

The first part details the State-of-The-Art methods related to the signature model. Within this first part, **Chapter 2** presents in more details the core ideas of pattern mining, as well as patterns that are closely related to the concept of signature. **Chapter 3** details core ideas that will be used to present extensions of the signature: the sky-signature and the MDL based signature. **Chapter 4** introduces core ideas that will be used in the signature mining algorithm.

Then, the second part introduces the signature model, as well as several experiments showing the relevance of this model. Within this second part, **Chapter 5** presents the signature model,

as well as some solutions to mine it. **Chapter 6** compares the signature with existing patterns, and shows the relevance of signatures to study retail customers. **Chapter 7** looks at the use of signatures to tackle a practical retail use case.

Next, the third part presents two extensions of signatures: the sky-signature and the MDL based signature. **Chapter 8** defines the sky-signature extension of signatures and its application to political speeches analysis, and **Chapter 9** presents the MDL based signature extension. Finally, **Chapter 10** concludes the thesis and discusses perspectives of this work.

PART I

State of The Art

CHAPTER 2

PATTERN MINING

Contents

The signature model introduced in this thesis relies heavily on the ideas of *pattern mining*. Indeed, the definition of the signature relies on a specific type of pattern called an *episode*. The algorithms developed to mine a signature are also based on algorithmic principles developed in the pattern mining field. For all these reasons, the fundamental definitions of pattern mining are presented, as well as the different types of patterns that are relevant to understand the link between the signature and existing types of patterns. Some well-known algorithmic approaches are also presented, with a focus on the algorithms that are used to develop signatures mining algorithm.

2.1 Frequent Itemset Mining

The pattern mining field historically started with the *frequent* pattern mining problem. The *frequent* pattern mining problem has first been studied in transactional databases. A transactional database $\mathscr{D}$ is a set of transactions, where each transaction t is a set of items and each transaction has a unique identifier named *tid*. A transactional database $\mathscr{D}$ can also be viewed as a set of tuples $\langle tid, t \rangle$. $\mathscr{I} = \{i_1, i_2, \ldots, i_n\}$ is the set of all possible items and $\mathscr{T}$ is the set of all *tid*. Therefore we have $t \subseteq \mathscr{I}$. $X \subseteq \mathscr{I}$ is an *itemset*. Table 2.1 gives an example of a transactional database.

Database $\mathscr{D}_1$	
TID	Items
T_1	Anchovies, Bread, Chocolate
T_2	Anchovies, Chocolate
T_3	Anchovies, Diaper
T_4	Anchovies, Bread, Chocolate, Diaper
T_5	Chocolate

$$\mathscr{I} = \{Anchovies, Bread, Chocolate, Diaper\}$$
$$t(\{Anchovies\}) = \{T_1, T_2, T_3, T_4\}$$
$$support(\{Anchovies\}) = 4$$

Table 2.1: An example of a transactional database.

Database $\mathscr{D}_1$ ordered	
TID	Items
T_1	Anchovies, Chocolate, Bread
T_2	Anchovies, Chocolate
T_3	Anchovies, Diaper
T_4	Anchovies, Chocolate, Bread, Diaper
T_5	Chocolate

Item count	
Item	Count
Anchovies	4
Chocolate	4
Bread	2
Diaper	2

Table 2.2: An example of a transactional database reordered according to descending item frequency.

Definition 1. *The tidset $t(X)$ of an itemset X is defined as the set of all tids that contains X as a subset:* $t(X) = \{t \in \mathscr{T} | X \subseteq t\}$

Definition 2. *The support of an itemset X is defined as the cardinal of its corresponding tidset:* $support(X) = |t(X)|$

Definition 3. *A frequent itemset X is an itemset that has a support greater than a user defined threshold θ.*

Definition 4. *The set $\mathscr{F}$ of all frequent itemsets is defined as* $\mathscr{F} = \{f \subseteq \mathscr{I} \mid support(f) \geq \theta\}$ *with θ the minimal support defined by the user.*

The frequent itemset mining problem is to find all frequent itemsets in a database, given a minimal support. In Table 2.1, if the minimal support is set to 4, the frequent itemsets are $\{Anchovies\}$ and $\{Chocolate\}$. If the minimal support is set to 3, $\{Anchovies\}$, $\{Chocolate\}$ and $\{Anchovies, Chocolate\}$ are frequent. Even if the database transactions have been ordered by item alphabetical order, it should be noted that there is no underlying order in itemsets, so the set $\{Anchovies, Chocolate\}$ is equivalent to $\{Chocolate, Anchovies\}$.

2.1.1 Mining Algorithms

Enumerating all possible itemsets and computing their support to decide whether they are frequent or not is not a feasible approach because the number of itemsets to enumerate is $2^{|\mathscr{I}|}$. To avoid this brute force approach, the first idea is to prune the search space. An efficient pruning strategy is proposed by Agrawal et al.[AS+94]. The pruning relies on the fact that if an itemset is frequent, then all its subsets are also frequent. In the example of Table 1, with a minimal support set to 3, since $\{Anchovies, Chocolate\}$ is frequent, then $\{Anchovies\}$ and $\{Chocolate\}$ are also frequent. This property is rather intuitive because if a transaction contains $\{Anchovies, Chocolate\}$, it also contains $\{Anchovies\}$ and $\{Chocolate\}$. We can deduce that $t(\{Anchovies, Chocolate\}) \subseteq t(\{Anchovies\})$ and $t(\{Anchovies, Chocolate\}) \subseteq t(\{Chocolate\})$. By definition of the support, it results that $support(\{Anchovies, Chocolate\}) \leq support(\{Anchovies\})$ and
$support(\{Anchovies, Chocolate\}) \leq support(\{Chocolate\})$. If we generalize this example, we have the following property: $\forall X \subseteq Y, Y \subseteq \mathscr{I} : support(X) \geq support(Y)$. From this anti-monotonicity property of itemset support with respect to itemset inclusion, we can deduce that if an itemset is not frequent, then all of its supersets are also not frequent. This allows to prune the search space by discarding all supersets of a not frequent itemset.

The well known *Apriori algorithm* [AS+94] exploits this idea. It explores the search space using a breadth-first search approach. The main idea of the Apriori algorithm is to generate candidate patterns of size k from frequent patterns of size $k-1$. The set of candidates of size k is built by joining frequent patterns of size $k-1$ that have a common prefix of size $k-2$. The result of the joining operation is kept in the candidate set if all of its subsets are frequent. This subset checking uses the Apriori property to prune patterns that cannot be frequent. Once the candidate set is computed, the Apriori algorithm counts the support of each pattern in the candidate set to determine whether it is frequent or not. It repeats this two steps process until there is no frequent candidate at some level. The set of the patterns mined at the k-th step corresponds to the set of all frequent patterns of size less than or equal to k.

The main costs of the Apriori algorithm come from the candidate set generation and the multiple database scans to count the support of each candidate. These costs become larger with low support or long patterns. To address these drawbacks, a new data structure has been developed by Han et al. [HPY00]. The algorithm they introduced (named FP-growth) tries to compress the database representation by using a frequent pattern tree (FP-tree). This structure is an extension of a prefix-tree and stores the database more compactly. The FP-growth algorithm then mines this tree to extract the frequent patterns. Another difference with the Apriori algorithm is that it mines the frequent patterns using a depth-first search. The multiple scans problem of the APriori algorithm is solved by the FP-tree structure and the candidate storage problem is solved by the depth-first search.

However, if the FP-tree does not fit in memory, the FP-growth algorithm approach becomes inefficient. To tackle this issue, a possible solution is to decompose the FP-tree bulding problem into independant subproblems. The idea is that each subproblem will correspond to building only a branch of the FP-tree [PK03]. Then, once the problem has been decomposed, one can develop efficient parallelization techniques so that each sub-problem fits in memory [Li+08].

2.1.2 Condensed Representations Mining

Some types of databases cannot always be efficiently mined using the previously described algorithms to find all frequent itemsets. First, finding long frequent itemsets requires an important number of operations, even for the FP-growth algorithm. Furthermore, the number of frequent patterns can be very high, especially if the dataset is dense or if the support is low. One can also notice that there is some redundancy within frequent itemsets. For example, in Table 2.2, the itemsets $\{Anchovies, Bread\}$ and $\{Anchovies, Bread, Chocolate\}$ always occur together. The information carried by $\{Anchovies, Bread, Chocolate\}$ is therefore sufficient and

outputting $\{Anchovies, Bread\}$ only adds redundancy. If we have a minimal support of 1 in Table 2.2, one can also see that $\{Anchovies, Bread, Chocolate, Diaper\}$ is frequent. Thanks to the Apriori property, it can be deduced that all its subsets are also frequent. Knowing that $\{Anchovies, Bread,$
$Chocolate, Diaper\}$ is frequent provides enough information to know that all of its subsets are also frequent, so outputting $\{Anchovies, Bread, Chocolate, Diaper\}$ only can be interesting.

To formalize the intuition of the previous example, the notion of *maximal* frequent itemsets has been defined.

Definition 5. *A frequent itemset $X \in \mathscr{F}$ is maximal iff* $\forall Y, Y \in \mathscr{F}, X \subset Y, |t(X)| \geq \theta \Rightarrow |t(Y)| < \theta$.

This subset of $\mathscr{F}$ is noted $\mathscr{M}$. Mining maximal frequent itemsets can drastically reduce the number of output patterns but their main drawback is that they are not sufficient to retrieve the support of any frequent itemsets. For example, with a minimal support of 2, $\{Anchovies, Bread,$ $Chocolate\}$ is the only maximal frequent itemset in Table 2.2. With this information only, deducing the support of its subsets is not possible. Since the support of all frequent itemsets is mandatory for some data mining tasks, such as association rule mining [AS+94], it is interesting to find a subset of frequent itemsets from which the support of every frequent itemset can be deduced. The subset of *closed* frequent itemsets from $\mathscr{F}$ carries all the information needed to compute the support of all frequent itemsets.

Definition 6. *A frequent itemset $X \in \mathscr{F}$ is closed iff* $\nexists Y \in \mathscr{F}, X \subset Y$ *and* $t(Y) = t(X)$.

Hence, a frequent itemset X is *closed* if none of the frequent supersets of X has the same support as X. Even if the number of closed frequent itemsets is generally one order of magnitude larger than the number of maximal frequent itemsets, the number of closed frequent itemsets is generally one order of magnitude smaller than the number of frequent itemsets. The compression efficiency of closed frequent itemsets is generally high in dense datasets. An interesting property of mining closed frequent itemsets is that is does not result in a loss of information.

Different algorithms are able to efficiently mine closed frequent itemsets [ZH02; UKA04]. Zaki et al. [ZH02] present CHARM, that mines closed frequent itemsets by exploring both the itemset space and the tidset (the set of transactions where an itemset occurs) space. Uno et al. [UKA04] introduce LCM, that efficiently mines closed frequent itemsets by defining a tree-shaped exploration of the search space, that solely looks at closed frequent itemsets. The complexity of these algorithms is in $O(2^{|\mathscr{I}|})$.

2.2 Episode Mining

Frequent itemsets are interesting to capture the correlation between items belonging to the same transaction. However, they disregard the temporal dependency between events. For example, in the retail context, it might be interesting to learn that when a customer buys chocolate, the next receipt is more likely to contain sugar free yogurts. Mining itemsets can only give the information that chocolate and sugar free yogurts are often bought together, not that their purchase is correlated in time.

To this end, sequential patterns have been introduced [AS95]. However, these sequential patterns are computed on a set of sequences. As this thesis focuses on analyzing one single sequence instead of a set of sequences, this part will focus on techniques analyzing a single sequence of events.

Definition 7. *An event sequence* α *is defined as* $\alpha = \langle (t_1, T_1), (t_2, T_2), ..., (t_n, T_n) \rangle$, $T_i \subseteq \mathscr{I}$, $t_i < t_{i+1}$ *for* $i = 1 \dots n-1$ *and* $t_1 \leq t_i \leq t_n$ *for* $i = 1 \dots n$.

T_i is the i-th transaction and t_i is the timestamp of that transaction. When $t_i = i$ for all i, we simply define $\alpha = \langle T_1, T_2, ..., T_n \rangle$.

To identify patterns in a single event sequence, episodes have been introduced [MTV97].

Definition 8. *An episode e is a triple* $(V, \leq, g)$ *where V is a set of nodes,* $\leq$ *is a partial order on V, and* $g : V \rightarrow E$ *is a mapping associating each node with an item. The items in g(V) have to occur in the order described by* $\leq$.

The size of e, denoted $|e|$, is $|V|$. An episode e is *parallel* if the partial order $\leq$ is trivial (i.e., $x \not\leq y \; \forall x, y \in V$ such that $x \neq y$). An episode e is serial if the relation $\leq$ is a total order (i.e., $\forall x, y \in V$, $x \leq y$ or $y \leq x$). Examples of episodes are illustrated in Figure 2.1.

As a parallel episode is uniquely defined by a set of items, we abuse the notation by denoting $g(V)$ by e. Hence, we have $e \subseteq \mathscr{I}$ a parallel episode.

In the remaining of this thesis, we will solely focus on parallel episodes, as they are closely linked to the signature model that we propose.

An episode occurrence relies on the definition of windows.

Definition 9. *A window* $\mathbf{w} = (w, t_s, t_e)$ *on* α *is an event sequence, with w containing all pairs* (t_i, T_i) *in* α *such that* $t_s \leq t_i \leq t_e$. $t_e - t_s$ *is defined as the window length. The set of all windows of size win in* α *is denoted* $\mathscr{W}(\alpha, win)$.

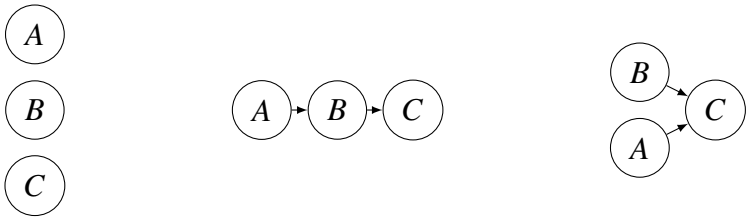

Figure 2.1: Some examples of episodes. An arc from left to right means that the source node precedes the end node in the sequence. The left episode is a parallel episode: the items can occur in any order. The middle episode is a serial episode, where the order between all items is total. The right episode is a general episode. The mapping g between each node and event is represented by the item name in each node.

For example, in Table 2.3, the window defined by $t_s = 2$ and $t_e = 4$ is such that $w = \langle(2, \{\text{Anch-}$
$\text{ovies, Chocolate}\}), (3, \{\text{Anchovies, Diaper}\}, (4, \{\text{Anchovies, Bread, Chocolate, Diaper}\})\rangle$

Definition 10. *A parallel episode $e = (V, \leq, g)$ occurs in a window $\mathbf{w} = (w, t_s, t_e)$ if $g(V) \subseteq \bigcup_{T_i \in w} T_i$.*

Definition 11. *A minimal occurrence of e in α is a window $\mathbf{w} = (w, t_s, t_e)$ where e occurs, and such that e does not occur in any sub-window $\mathbf{w}_s = (w, t_{s1}, t_{e1}), t_s < t_{s1} \leq t_e$, and $t_s \leq t_{e1} < t_e$ of $\mathbf{w}$.*

A parallel episode e is *frequent* if it has more than θ minimal occurrences in α, with θ the support threshold. The maximal number of occurrences of an episode e in sequence α is denoted $support(e, \alpha)$.

Definition 12. *Two occurrences of e are* non-overlapping *if their associated windows $\mathbf{w}_1 = (w_1, t_{s1},$*
$t_{e1})$ and $\mathbf{w}_2 = (w_2, t_{s2}, t_{e2})$ do not overlap, that is $t_{s1} > t_{e2}$ or $t_{s2} > t_{e1}$.

The maximal number of non overlapping occurrences of an episode e in sequence α is denoted $NoOverlapSupport(e, \alpha)$. With this new definition of support, one can mine *frequent non overlapping* episodes, that is episodes with a $NoOverlapSupport$ greater than θ.

For example, in Table 2.3, the parallel episode $\{Anchovies, Bread, Diaper\}$ has 2 minimal occurrences. The first minimal occurrence corresponds to the window containing transactions 1, 2 and 3. The second minimal occurrence corresponds to the window containing transaction 4. The support of $\{Anchovies, Bread, Diaper\}$ is therefore 2.

As noted in [CG03], this definition of episode requires another parameter: the maximal size of a window. For example, if the maximal window size is 2, the first occurrence of $\{Anchovies, Bread, Diaper\}$ is discarded, as its window contains 3 transactions.

To overcome the necessity of defining a maximal window size, Casas-Garriga [CG03] introduced unbounded episodes, where instead of defining the maximal window size, one has to define the maximal time interval *tus* between two consecutive items of the episode. While this allows the window size to grow with the number of items it contains, the maximal window size of an episode containing k items is $(k-1)*tus$.

The method for mining frequent episodes is based on the same principles as frequent itemset mining. Indeed, the Apriori [AS+94] property still holds for episodes, hence approaches developed to efficiently mine frequent itemsets have been adapted to efficiently mine frequent episodes [MTV97; CG03].

2.3 Periodic Pattern Mining

While episodes can identify complex patterns in a sequence, these patterns are sometimes difficult to understand. Moreover, episodes do not take into account event timestamps, only their temporal order.

A natural type of sequential patterns based on timestamps are periodic patterns. Indeed, many of our habits are based on a regular period. For example, lunch hours repeat themselves on a daily basis, with some weekly changes during week-ends. Therefore, periodic patterns have been of interest in the pattern mining community [HDY99; ORS98; MH01]. The basic idea of frequent periodic pattern mining is to detect the recurrent itemsets that occur at a fixed period. The next paragraphs present how the periodic behavior has been defined in pattern mining.

Definition 13. *Given an itemset X and a period p, the cycle of the periodic pattern (X, p), denoted cycle(X, p), is the maximal number of transactions containing X, that are separated by a distance of p time units.*

For example in Table 2.3, we have cycle$(\{Anchovies\}, 1) = 4$. Indeed, $\{Anchovies\}$ occurs in transactions T_1, T_2, T_3 and T_4, so there are 4 transactions separated by a timestep of 1 such that $\{Anchovies\}$ occurs in all of them. As adding T_5 to this sequence of transactions would violate the requirement that $\{Anchovies\}$ has to occur in each transaction, we cannot add a new transaction to this cycle, hence cycle$(\{Anchovies\}, 1) = 4$. Likewise we have cycle$(\{Anchovies, Bread\}, 3) = 2$. We also have cycle$(\{Anchovies\}, 2) = 2$, with 2 cycles

Database $\mathscr{D}_1$			
	Timestamp	TID	Items
●	1	T_1	Anchovies, Bread, Chocolate
●	2	T_2	Anchovies, Chocolate
●	3	T_3	Anchovies, Diaper
●	4	T_4	Anchovies, Bread, Chocolate, Diaper
	5	T_5	Chocolate

Table 2.3: An example of a timestamped transactional database. The blue dots represent the first cycle of the periodic pattern $(\{Anchovies\},2)$. The red dots represent the second cycle of the periodic pattern $(\{Anchovies\},2)$.

matching this description. Indeed, the cycle composed of T_1 and T_3 and the cycle composed of T_2 and T_4 both contain $\{Anchovies\}$ in all of their transactions, with a timestep of 2 between all transactions. Both cycles are highlighted in Table 2.3. Note that when the cycle of a pattern is equal to 1, we discard it, as a cycle with only one element is not considered a valid cycle.

Definition 14. *The* support *of a periodic pattern* (X,p) *is the sum of non-overlapping cycles.*

For example in Table 2.3, we have support$(\{Anchovies\},1)=4$ and support$(\{Anchovies\},2)=2$. As described above, there are two cycles of size 2 that match cycle$(\{Anchovies\},2)=2$, but they are overlapping, as the first transaction T_2 of the second cycle is situated in the middle of the first cycle $T_1 \rightarrow T_3$. This is visible in Table 2.3, as the blue dots and red dots overlap. This definition of periodic pattern corresponds to *partial periodic* patterns [MH01], where gaps of arbitrary size between two consecutive cycles are allowed. A frequent periodic pattern is a periodic pattern having a support greater than θ.

Periodic patterns have mainly been applied to study logs of programs to detect bugs, or classify program behaviors [Li+06; LXM08; Lo+09]. Overall, periodic patterns are very well suited to analyze program logs, as their periodic behavior is strongly enforced by the underlying code. It should be noted that while periodic patterns are useful to study event sequences with underlying periodicity, the periodic constraint on the occurrences of an itemset is rarely satisfied when analyzing less regular event sequences, such as alarm logs.

Summary

- The set of frequent itemsets can be summarized by some of its subsets: maximal or closed itemsets
- Episode are useful to identify temporal regularities in a single sequence. These temporal regularities are defined as a partial order on items occurrences
- Periodic patterns are able to identify periodic behavior in temporal data, with some flexibility on the periodic occurrences

CHAPTER 3

MODEL SELECTION IN PATTERN MINING

Contents

All the pattern mining techniques described so far only take into account the frequency of each pattern to decide whether it is interesting or not. While this information is generally interesting, one can obtain a huge amount of frequent patterns using this support constraint only. Analyzing such large sets of patterns is not an easy task and the user is generally flooded with too many patterns to be able to extract meaningful conclusions from it. Moreover, the user might want to mine *interesting* patterns, where *interesting* is not necessarily defined by pattern frequency. For example, one might want to mine patterns that have a minimal length. One could also specify a constraint on the sum of the itemset values. For example, in a retail database, one could associate a price to each item and only mine the itemsets having a total price greater than a certain threshold.

A first approach to deal with the pattern explosion problem is to build a summary of all frequent patterns. This summary can, for example, be the result of a clustering process [LSW97] or the result of a coverage optimization process [AGM04]. The goal of these approaches is to be able to represent the whole set of frequent patterns efficiently. The *interestingness* of the resulting patterns is not taken into account in these types of approach.

Another classical approach to reduce the number of output patterns is the pattern set approach. The idea is to find the k best patterns, that maximize a given measure, while satisfying a set of constraints [DRZ07]. These pattern sets can be mined using contraint programming [GNDR13] or optimization techniques [LK12; Bos+18]. One advantage of such a technique is that constraints can be applied to a whole set of patterns, hence enabling practitioners to define measures such as interestingness and redundancy on a whole set of patterns. To solve the pattern

Database $\mathscr{D}_1$	
TID	Items
T_1	Anchovies, Bread, Chocolate
T_2	Anchovies, Chocolate
T_3	Anchovies, Diaper
T_4	Anchovies, Bread, Chocolate, Diaper
T_5	Chocolate

$\mathscr{I} = \{Anchovies, Bread, Chocolate, Diaper\}$
$t(\{Anchovies\}) = \{T_1, T_2, T_3, T_4\}$
$support(\{Anchovies\}) = 4$

Table 3.1: An example of a transactional database.

set discovery problem, constraint programming has been used successfully.

However, setting the k value is not easy in practice. Skyline patterns have been introduced [Sou+11] to find patterns that correspond to an optimal compromise between a set of measures. These patterns are presented in more details in Section 3.1.

Another approach to fix the number of output patterns is to rely on compression principles [LV14]. The main goal of this technique is to mine the set of most informative patterns, while avoiding redundancy. This technique is presented in more details in Section 3.2. A more complete survey of techniques to mine fewer interesting patterns is presented in [VT14].

The following sections present two techniques selecting sets of interesting patterns: the skyline approach and the compression based approach. Both approaches were chosen because they are parameter free.

3.1 Skyline Patterns

In [Sou+11], the authors defined a new type of patterns to mine, called *skyline patterns*. These patterns rely on the fact that, according to a set of measures, they are dominant in a set of patterns.

Definition 15. *A pattern X dominates another pattern Y with respect to a set of measures M iff for any measure $m \in M$, $m(X) \geq m(Y)$ and there exists at least one $m \in M$ where $m(X) > m(Y)$.*

For example, in Table 3.1, $\{Anchovies, Chocolate\}$ dominates $\{Anchovies,$ $Bread\}$ on the set of measures $\{support, length\}$ because $length(\{Anchovies, Chocolate\}) = length(\{Anchovies, Bread\}) = 2$ and $support(\{Anchovies, Chocolate\}) = 3 >$

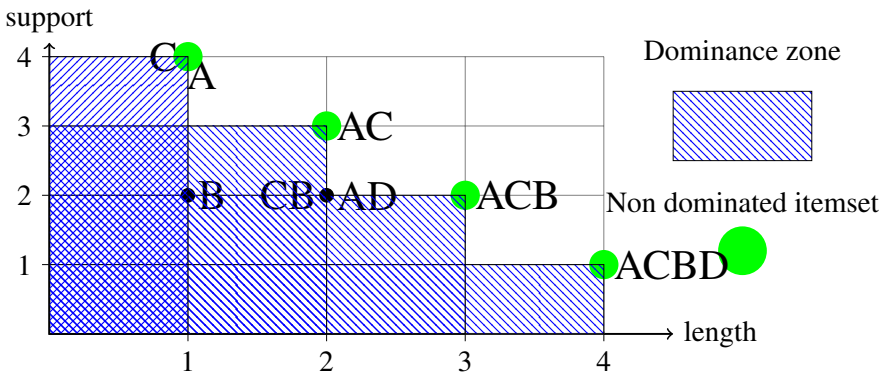

Figure 3.1: Dominance of some itemsets in Table 3.1. Each dominance zone is represented by a dashed rectangle. Each item is replaced by its first letter.

$support(\{Anchovies, Bread\}) = 2$. An illustration of the pattern dominance on some itemsets of Table 3.1 is shown in Figure 3.1.

Once the dominance is defined, a skyline pattern is defined as follows: a pattern X is a skyline pattern on a set of measures M and a set of patterns P if X is not dominated in P with respect to M. In database $\mathscr{D}_1$ represented in Table 3.1, the skyline patterns with respect to the set of measures $\{support, length\}$ are $Sky(D_1, \{support, length\}) = \{\{Anchovies\}; \{Chocolate\}; \{Anchovies, Chocolate\}; \{Anchovies, Bread, Chocolate\}; \{Anchovies, Bread, Chocolate, Diaper\}\}$.

Once again, a naive method would be to select these skyline patterns by post processing the frequent itemsets. This method is not very efficient and a more interesting algorithm has been defined in [Sou+11]. One can notice that the skyline patterns computation cost grows exponentially with the pattern set size. Therefore, the idea of the algorithm presented in [Sou+11] is to reduce the pattern set before computing the skyline patterns. This smaller set is determined using a selected subset of the measure set defined by the user. This measure subset has a property that allows the algorithm to exclude some patterns from the skyline pattern computation, without any loss of information. For example, if the measure set is $\{support, area\}$, where $area(X) = support(X) * length(X)$, one can notice that if the support remains constant, the area augments with the length of X. Therefore, if the support is constant, the dominant pattern is the one with the highest length. Such a measure is said to be maximally $\{support\}$-skylineable.

The formal definition of skylineability states that a set of measures M is said to be minimally M'-skylineable, $M' \subseteq M$, iff for any pattern, $X =_{M'} Y$ such that $X \subset Y$, then $X \geq_M Y$. A measure set is maximally M'-skylineable if the previous definition holds for $Y \subset X$. It has been shown that any set of measures is minimally and maximally skylineable. The authors of [Sou+11] define converters to compute the subsets of measures where a measure is minimally or maximally

skylineable. Then, these measure subsets are used in conjunction with a new operator to select a subset of patterns that contains all dominant patterns. This new operator is named *distinct* and generalizes the idea of closed itemsets to a broader set of measures. Once the subset of patterns is computed, the computation of dominant patterns is performed on this subset. The resulting pattern set contains all the information needed to retrieve (when necessary) the whole set of dominant patterns. Since the dominant patterns computation is achieved on a subset of all patterns, this method is efficient and uses constraints to reduce the size of this subset.

3.2 Compression Based Pattern Mining

While skyline patterns aim at reducing the number of output patterns, the resulting patterns can still be redundant. Having redundant patterns usually means that the number of resulting patterns can be further reduced. To this end, pattern mining research has shifted to the extraction of *pattern sets*: finding a subset of patterns that together optimize some interest criterion [DRZ07].

One of the most interesting criterion is based on the use of the Minimum Description Length (MDL) principle [Grü07]. The main idea of the MDL principle is that *the best model compresses data the best*. The paragraph below introduces the basic notions of the minimum description length principle (MDL) as it is commonly used in compression-based pattern mining.

Given a set of models $\mathscr{M}$ and a dataset $\mathscr{D}$, the best model $M \in \mathscr{M}$ is the one that minimizes $L(\mathscr{D},M) = L(M) + L(\mathscr{D}|M)$, with $L(M)$ the length, in bits, of the encoding of M, and $L(\mathscr{D}|M)$ the length, in bits, of the encoding of the data given M. This way of defining the length of the data is named two-part MDL. To compare all models on an even footing, the encoding of the data has to be *lossless*. In practice, the model class has to be defined, as well as how to compute the length of the model and the length of the data given the model. It should be noted that only the *encoded length* of the data is of interest, not the encoded data themselves.

Based on this principle, the authors of KRIMP [VLS11] and SLIM [SV12] have proposed in the context of transactional databases, to compute a subset of itemsets that yield the best lossless compression of the database. The idea of both algorithms is to mine the *code table* that minimizes the code length. A code table CT associates a code to each itemset. The code length of each itemset in the code table is linked to the number of times this itemset is used to cover transactions of the database. The more the itemset is used, the shorter the code.

To encode the database using the code table, one first needs to cover the database with codes from the code table. This transaction covering process is also solved heuristically by considering non-overlapping coverings and by introducing an order in the covering process (largest itemsets

Database $\mathscr{D}_1$	
TID	Items
T_1	Anchovies, Chocolate, Bread
T_2	Anchovies, Chocolate
T_3	Anchovies, Diaper
T_4	Anchovies, Chocolate, Bread, Diaper
T_5	Chocolate

Code Table	
Itemset	Usages
Anchovies, Chocolate, Bread	2
Anchovies, Chocolate	1
Anchovies	1
Chocolate	1
Bread	0
Diaper	2

Table 3.2: An example of a code table (CT) associated with a database. The usages correspond to the number of times the itemset is used to cover a transaction from $\mathscr{D}_1$.

first).

For example, in Table 3.2, the transaction T_1 is covered as follows. The largest itemset in the code table $\{Anchovies, Chocolate, Bread\}$ is first considered. As it is included in T_1, we will use it to cover T_1, hence we increase $\{Anchovies, Chocolate, Bread\}$ usage by 1. Then, we look at the remaining items from T_1, that are not already covered: none in this case. We repeat this process to cover all the transactions in the database.

Once we have the usages of each itemset, the code length of an itemset X is $L(code(X)) = -\log(P(X|D)) = -\log \frac{usage(X)}{\sum_{X \in CT} usage(X)}$. Hence, if an itemset is often used, its associated code will be shorter. This property is a key to mine a condensed set of itemsets, as the algorithm will favor using existing itemsets over using rarely used new itemsets to cover transactions.

To mine the code table, both KRIMP and SLIM greedily select the best itemset that can be added to the code table. In KRIMP, the itemsets that can be added to the code table have to be mined beforehand, whereas SLIM generates them according to the current code table. The resulting code table gives the set of itemsets that is optimal according to the MDL criterion.

Other works [TV12; BV17; Lam+12] have considered the MDL principle for databases of event sequences and extract a subset of sequential patterns with respect to the MDL criterion. These approaches also rely on using code tables to mine the optimal pattern set according to the MDL criterion.

For approaches mining more complex patterns than itemsets (episodes or sequential patterns), the code table is more complex than the one used in KRIMP or SLIM. Indeed, new symbols marking the presence of a gap in the pattern and the continuation of a previous pattern have to be added. The encoding process is also more complex, as covering transactions with episodes requires the information about when gaps and continuations of patterns occur.

Summary

- Outputting all patterns is not desirable, as it quickly becomes impractical to look at all of them.
- The skyline and the compression based approaches are valuable, mainly because they are parameter free. Both approaches can be used to select a subset of signatures.
- The skyline approach outputs compromises on a given set of measures.
- The compression based approach outputs the most informative patterns, while avoiding redundancy.

CHAPTER 4

SEQUENCE SEGMENTATION

Contents

This section, presents the sequence segmentation problem. While the signature model introduced in this thesis initially belonged to the pattern mining field, we could also relate the signature model to the sequence segmentation field.

This allows us to use existing approaches to efficiently tackle the sequence segmentation problem. It is especially interesting to be able to use 2 different fields to tackle one problem, as each field has its own strengths and weaknesses.

The sequel presents the sequence segmentation problem, as well as some state of the art algorithms that are used to efficiently tackle this problem.

4.1 Sequence Segmentation Problem

Sequence segmentation [TT06] aims at efficiently summarizing a long sequence by a few key representatives. A typical summary consists in computing a piece-wise constant approximation of the sequence. The idea is to split the sequence into consecutive segments, and compute a single value that will approximate the whole segment. This value is called the *segment representative*. Summarizing long sequences is of interest in many applications, such as bio-informatics, market analysis or text processing.

To evaluate how good such an approximation is, one needs to define an error function that measures how well the sequence is approximated by all its segment representatives. Typically, the sum of squared differences between each point in a segment and its representative is used.

Given a sequence, an error function and a number of segments, the segmentation problem consists in finding the segmentation and the associated representatives that minimize the approximation error.

Database $\mathscr{D}_1$		
TID	Items	Binary vector
T_1	Anchovies, Chocolate, Bread	$(1,1,1,0)$
T_2	Anchovies, Chocolate	$(1,0,1,0)$
T_3	Anchovies, Diaper	$(1,0,0,1)$
T_4	Anchovies, Chocolate, Bread, Diaper	$(1,1,1,1)$
T_5	Chocolate	$(0,0,1,0)$

Table 4.1: An example of a 3-segmentation of $\mathscr{D}_1$. We have $S_1 = \langle T_1 \rangle$, $S_2 = \langle T_2, T_3, T_4 \rangle$, $S_3 = \langle T_5 \rangle$ and $S = \langle S_1, S_2, S_3 \rangle$.

In [Bin10], the segmentation problem is formulated as follows. Let $\alpha_{\mathbb{R}} = \langle T_1, T_2, ..., T_n \rangle$ be a d-dimensional sequence where $T_i \in \mathbb{R}^d$. An event sequence, as defined in Definition 7, can be transformed into a $|\mathscr{I}|$-dimensional sequence by transforming each $T_i \in \mathscr{I}$ into a $|\mathscr{I}|$-dimensional binary vector v_i, with $v_{ij} = \mathbf{1}_{j \in T_i}$.

A k-segmentation $S(\alpha, k)$ of α is a partition of α into k non-overlapping contiguous subsequences called *segments*, i.e. $S(\alpha, k) = \langle S_1, S_2, ..., S_k \rangle$ and $\forall i \in 1 \dots k$, $S_i = \langle T_{b(i)}, \dots, T_{b(i+1)-1} \rangle$, where $b(i)$ is the index of the first element of the i-th segment. An example of a 3-segmentation is given in Table 4.1. $S(\alpha, k)$ is denoted S whenever both α and k are clear from the context.

A segmentation associates a *representative*, $\mu(S_i)$, with each segment by aggregating the values of the segment. Generally $\mu(S_i)$ is a single value such as mean or median, or a pair of values such as (min, max) or (mean, slope). This reduction results in a loss of information in the sequence representation which can be measured by the reconstruction error defined as:

$$E_p(S(\alpha, k)) = \sum_{S_i \in S} \sum_{T \in S_i} dist(T, \mu(S_i))^p$$

where $dist(T, \mu(s))$ represents the distance between the d-dimensional point T and the representative of the segment it belongs to. Parameter p refers to the Lp norm. In practice, the median ($p = 1$) or the mean ($p = 2$) usually serves as segment representatives. The *segmentation problem* consists in finding the segmentation that minimizes the reconstruction error:

$$S_{opt}(\alpha, k) = \arg\min_{S \in \mathbb{S}_{n,k}} E_p(S(\alpha, k))$$

where $\mathbb{S}_{n,k}$ represents the set of all k-segmentations of sequences of length n.

4.2 Algorithms

To optimally solve this segmentation problem, dynamic programming (DP) [Bel61] can be used. Given a sequence $\alpha = \langle T_1, T_2, ..., T_n \rangle$, k a number of segments and E an error function, algorithm DP finds the optimal segmentation by solving the optimal segmentation for numbers of segments smaller than k and for subsequences $\langle T_1, \ldots, T_i \rangle$, $i \in [1, n]$. The complexity of this approach is $O(n^2 k)$. One byproduct of this algorithm is that it also gives the optimal segmentations for number of segments smaller than k.

When processing large sequences, the quadratic complexity with respect to the sequence length makes this approach impractical. To this end, heuristic approaches have been developed to solve the segmentation problem [TT06].

A commonly used heuristics is the top-down greedy algorithm. This algorithm starts with the whole sequence as a single segment. Then, a new segment is added, such that it is the one that greedily minimizes the error function. The greedy addition of a new segment is repeated until the number of segments is equal to k. The complexity of this algorithm is in $O(nk)$.

Likewise, one can use the bottom-up greedy algorithm. In that case, one starts with each transaction being a segment. Then, at each step of the algorithm, two consecutive segments are merged, such as this merge greedily minimizes the error function. Again, the algorithm stops when the number of segments is equal to k. The complexity of this algorithm is in $O(kn \log n)$.

While both algorithms work well in practice [HG04; TT06], there is no theoretical guarantee that the resulting segmentation is close to the optimal one. To tackle this issue, Terzi et al. [TT06] have proposed the divide and segment (DnS) algorithm, with guarantees that the resulting segmentation error is at most 3 times the minimal error. The complexity of this algorithm is in $O(n^{4/3} k^{5/3})$.

To summarize, the segmentation problem can be solved more or less efficiently, depending on how much guarantees are required on the resulting segmentation.

Summary

- The sequence segmentation problem consists in segmenting a sequence into segments that are good local approximations
- Dynamic programming can optimally solve this problem
- Heuristic approaches have been developed to efficiently find an approximate solution

PART II

The Signature Model

CHAPTER 5

SIGNATURE MINING

Contents

The signature model is the main contribution of this thesis. This part introduces this new type of pattern, as well as an efficient algorithm, based on sequence segmentation, to compute it. Then, I compare the signature with some state-of-the-art techniques, and demonstrate the practical use of signatures.

5.1 Motivations

The idea of signature comes from the retail domain In the context of the analysis of retail customers, the main purpose of signatures is to capture the *favorite products* of a customer.

Defining what are the favorite products of a customer is not an easy task. An intuitive idea is that these products are those that a customer always want to have at home. These products can have different replenishment rates (you do not buy milk as often as cleaning products, yet you always want to have both at home), and hence they can be bought across several transactions. The order in which these products are bought is not really of interest, only the fact that they are bought regularly is important.

Moreover, customers can be erratic in their behavior. Indeed, as purchase data mainly comes from loyalty cards, if a customer forgets the loyalty card, then the purchase is skipped in the

dataset. Furthermore, when taking vacations, a customer is likely to exhibit a non regular purchase behavior. There are many other human behaviors that can cause purchase data to be incomplete.

From this intuition, we can see that the models presented in the previous part are not suited to tackle the many challenges posed by retail data. Indeed, periodic patterns are not suited to analyze products with different replenishment rates, and they have a hard time adapting to erratic behaviors. Episodes generally can adapt to erratic behavior, but analyzing products with different replenishment rates makes it difficult to set a window size beforehand.

Hence, the main contribution of this thesis is to introduce a new pattern mining model, called a *signature*, that is able to cope with erratic human behavior and heterogeneous replenishment rates to identify the favorite products of a retail customer. As will see in the experiments, this model can also be applied successfully to other datasets, that share similar properties, such as political speeches.

5.2 Signatures Definition

This section, presents the signature model in the sequence segmentation framework. Indeed, mining a signature from an event sequence α, as defined in Definition 7, can be seen as segmenting α into k non-overlapping and non-empty segments that cover all transactions from α and such that every segment contains a maximal common subset of items, called the *signature*. The signature model can thus be fitted to the segmentation model, providing the opportunity to use the many, exact or approximate, algorithms that have been proposed in the sequence segmentation field.

5.2.1 Defining Signatures with Sequence Segmentation

Section 4.1 introduced the problem of segmenting an event sequence $\alpha = \langle T_1, T_2, ..., T_n \rangle$ into k segments. Let $\mathcal{S}_{n,k}$ denote the set of all k-segmentations of a sequence α of length n and $S(\alpha, k) = \langle S_1, S_2, ..., S_k \rangle$ be an element of $\mathcal{S}_{n,k}$. Following the representation proposed in SPAM [Ayr+02], a transaction can be represented by a bitmap[1] of dimension d such that if item i_j belongs to the transaction then the j-th bit of the bitmap is set to 1, otherwise the j-th bit is set to 0.

1. For the sake of simplicity, we focus here on a bitmap representation. To cope with memory consumption, a more efficient representation method, such as the dynamic bit vector (DBV) architecture, could be used [VHL12].

Definition 16. *The representative r_i of a segment is then defined as the set of items that belongs to at least one transaction in the segment, i.e. the union of the segment transactions. This union can be computed by a boolean disjunction on bitmaps: $r_i = \bigcup_{t \in S_i} t$.*

For example, in Table 5.1, we depicted a 2-segmentation of a database. The first segment S_1 contains T_1 and T_2 and the second segment S_2 contains T_3 to T_7. $r1$, the representative of S_1, corresponds to the union of T_1 and T_2: $r_1 = T_1 \bigcup T_2$, $r_1 = \{Bread, Milk, Chocolate, Cheese\} \bigcup \{Bread, Orange\,juice, Steak\} = \{Bread, Milk, Chocolate, Cheese, Orange\,juice, Steak\}$. Similarly, we derive r_2, the representative of S_2: $r_2 = \{Bread, Milk, Chocolate, Apples, Fish, Orange\,juice\}$

Definition 17. *By definition $\widetilde{Sig}(S(\alpha,k)) = \bigcap_{i=1}^{k} r_i$ are the recurrent items of $S(\alpha,k)$. In other words, $\widetilde{Sig}(S(\alpha,k))$ contains the set of all the recurrent items that are present in each segment of $S(\alpha,k)$.*

For example, in Table 5.1, the intersection of the two representatives r_1 and r_2 will be $\{Bread,$
$Milk, Chocolate, Orange\,juice\}$. Therefore, $\widetilde{Sig}(S(\alpha,2)) = \{Bread, Milk, Chocolate, Orange\,juice\}$.

As we intend to represent a sequence by its signature, the reconstruction error is related to the loss of information in the signature. A simple way to estimate the error is to count the items that are not present in the signature, i.e. the number of bits equal to 0 in the bitmap:

$$E(S(\alpha,k)) = |\mathscr{I}| - \|\widetilde{Sig}(S(\alpha,k))\| = \|\overline{\widetilde{Sig}(S(\alpha,k))}\|$$

where $\|X\|$ represents the number of bits equal to 1 in bitmap X and $\overline{X}$ represents the complement of X. This error therefore penalizes short signatures. The idea is that the longer the signature the more habits are captured, hence we prefer large signatures to short ones.For example, in Table 5.1, there are 8 items, and the signature contains 4 items. Hence, the loss value associated with the signature is 4.

This error function differs from the error function of the sequence segmentation presented in Section 4.1, as it maximizes the size of the intersection of *all* segment representatives. Because the error function depends on all segments, it does not verify the Bellman optimality principle [Bel13], hence the dynamic programming based approaches developed in the sequence segmentation field are not directly applicable to optimally solve our segmentation problem. Nevertheless, Section 5.3.1 presents how dynamic programming based approaches can be adapted to optimally solve our segmentation problem.

Definition 18. *A k-signature, $Sig(\alpha,k)$, is defined as a* maximal *set of recurrent items in a k-segmentation of α. $Sig(\alpha,k)$ is* maximal *means that it is obtained from a k-segmentation of α*

A 2-segmentation of a database	
TID	Items
T_1	Bread, Milk, Chocolate, Cheese
T_2	Bread, Orange juice, Steak
T_3	Bread, Milk
T_4	Bread, Chocolate, Apples
T_5	Bread, Fish
T_6	Milk, Orange juice
T_7	Bread, Milk

The segment representatives	
RID	Items
r1	**Bread**, **Milk**, **Chocolate**, Cheese, **Orange juice**, Steak
r2	**Bread**, **Milk**, **Chocolate**, Apples, Fish, **Orange juice**

Table 5.1: An example of a 2-segmentation S of a database α and its associated segment representatives r_1 and r_2. The signature items are represented in bold.

that maximizes the size of the recurrent items set: $Sig(\alpha,k) = \widetilde{Sig}(S_{max}(\alpha,k))$ *with* $S_{max}(\alpha,k) = argmax_{S(\alpha,k)\in\mathbb{S}_{n,k}} |\widetilde{Sig}(S(\alpha,k))|$.

As the number of items is fixed, minimizing the loss $E(\alpha,k)$ is equivalent to maximizing the number of items in the signature $\|Sig(\alpha,k)\|$. For example, in Table 5.1, if we iterate over all possible 2-segmentations, we notice that the maximal number of items in the signature is 4. Hence, the segmentation depicted in Table 5.1 yields a signature.

The segmentation size k is given a priori, either as an integer or as a percentage of n, the size of the input sequence (similar to support count and support [AIS93]). The latter is called the **relative number of segments** denoted by **RNS**.

This definition of the signature meets our definition of the favorite products of a customer. Indeed, as a segment may contain many transactions, we can detect purchases spread over many transactions. Furthermore, there are no constraint on the size of a segment, hence the signature is able to adapt to different replenishment rates, as well as cope with erratic behavior. Finally, the minimal number of segments ensures that the signature only contains products that are bought several times. We look for the largest set of products that occurs in k segments, as we want to find all the purchase habits of a customer, according to our definition of favourite products.

5.2.2 Signature as an Episode

The signature model can also be defined as a particular episode. Indeed, a signature with k segments can be seen as a maximal parallel episode without repetition, with k minimal non-overlapping occurrences and no window size constraint.

Let us recall the fact that a parallel episode $e \subseteq \mathscr{I}$ occurs in a window $\mathbf{w} = (w, t_s, t_e)$ if $e \subseteq \bigcup_{T_i \in w} T_i$. If we consider the window $\mathbf{w}$ as a segment, the representative of w would be $r_w = \bigcup_{T_i \in w} T_i$. Hence, we have $e \subseteq r_w$.

A maximal parallel episode e_{max} with k non-overlapping occurrences (as defined in Definition 12) in α is defined as $e_{max} = \arg\max_{e \subseteq \mathscr{I}}(|e|)$, $NoOverlapSupport(e, \alpha) \geq k$. It is important to note that unlike classical episodes, there is no constraint on the window size.

From e_{max} and its occurrence windows, we can retrieve the corresponding signature. Indeed, each segment of the signature corresponds to one non-overlapping occurrence, and any transaction that does not belong to a minimal occurrence can be allocated to any neighboring segment.

For example, in Table 5.2, the maximal parallel episode having 2 non-overlapping minimal occurrences is represented. From these non-overlapping minimal occurrences of the maximal episode, we can build the segmentation of the signature, as explained above. Indeed, the first segment has to contain transactions 1 and 2. Likewise, the second segment has to contain transactions 4 to 6. Then, transaction 3 can either belong to the first or second segment, and transaction 7 can only belong to the second segment, as segments have to contain contiguous transactions.

The main difference between this definition of the signature based on episodes and the one based on sequence segmentation is that the arg max does not iterate on the same objects. For the definition using sequence segmentation, the maximization of the product set size is performed over the set of k segmentations of α. For the definition using episodes, the maximization of the product set size is performed over the set of itemsets of $\mathscr{I}$.

The fact that we can either iterate over the set of itemsets or the set of segmentations shows that both sets are related through the signature model. Indeed, given a segmentation S, the associated itemset corresponds to the signature $Sig_k(\alpha, S)$, i.e. the intersection of the segment representatives. Given an itemset $I \subseteq \mathscr{I}$, we can find a segmentation S, such that each segment representative of S contains I. The segmentation associated with I is the segmentation that has

A 2-segmentation of a database	
TID	Items
T1	Bread, Milk, Chocolate, Cheese
T2	Bread, Orange juice, Steak
T3	Bread, Milk
T4	Bread, Chocolate, Apples
T5	Bread, Fish
T6	Milk, Orange juice
T7	Bread, Milk

Table 5.2: A maximal episode with 2 minimal non-overlapping occurrences. The first minimal occurrence is depicted in blue, the second one in red.

the greatest number of segments, where each segment representative contains I. It should be noted that this segmentation is generally not unique.

One easy way to compute a segmentation associated to I is to iterate over α and record what items occur in the transactions seen so far. Once I is included in this set of items, we place a segment boundary at the current transaction and set the set of items seen so far to the empty set. We repeat the process until the end of α is reached.

Therefore, we can see that a segmentation defines an associated itemset, and an itemset defines an associated segmentation. This gives an intuition of why signatures can fit within both the sequence segmentation and the episode models.

5.3 Algorithms

This section presents 3 signature mining algorithms. All three algorithms are based on existing algorithms from the sequence segmentation field. Algorithms based on the definition of the signature as an episode will be presented in Section 8.3.

5.3.1 Dynamic Programming for Signature Mining by Segmentation

In this subsection, a straightforward adaptation of the dynamic programming (DP) algorithm is presented. This adaptation is not optimal, but helps in understanding how the DP algorithm can be adapted to the signature mining problem. Then, an optimal adaptation of the DP algorithm is described.

Basic Non-Optimal Algorithm

Bingham [Bin10] presents a formulation of DP for sequence segmentation. It is based on a matrix A of size $k \times n$ where k is the size of the segmentation and n is the number of transactions (receipts) in the input sequence α. Rows of A represent segments and columns represent the itemsets of the input sequence. Let $\alpha[j,i]$ denote the subsequence of α starting at index j and ending at index i. A cell $A[s,i]$ denotes the error of segmenting the sequence $\alpha[1,i]$ using s segments, formally defined by:

$$A[s,i] = \min_{2 \leq j \leq i} (A[s-1, j-1] + E(S_{opt}(\alpha[j,i],1))) \tag{5.1}$$

where $E(S_{opt}(\alpha[j,i],1))$ is the minimum error that can be obtained for the subsequence $\alpha[j,i]$ when representing it as one segment. In our case, it is simply the size of the union of the segment transactions.

In the signature mining problem, as presented in Section 5.2.1, the representative of a segment is not a numeric value but a set of items. In order to compute the reconstruction error, a cell in A stores the bitmaps of the best signatures obtained so far. Since several signatures may exhibit the same reconstruction error value, an A cell contains a set of bitmaps. Intuitively, $A[s,i]$ is computed by considering, for all $j \in [s,i]$, the composition of a signature obtained for an $(s-1)$-segmentation of a subsequence $\alpha[1,j-1]$, stored in $A[s-1,j-1]$, and the signature of the new segment $\widetilde{Sig}(S(\alpha[j,i],1))$.

Formally, it is defined by:

$$A[s,i] = \underset{s \leq j \leq i}{\text{amaxN}}\left(\underset{Sig_{s-1} \in A[s-1,j-1]}{\text{amaxN}}\left(Sig_{s-1} \cap Sig_1(\alpha[j,i])\right)\right) \tag{5.2}$$

where

$$\underset{i}{\text{amaxN}}\, P(i) \equiv \underset{P(i)}{\arg\max}\, |P(i)|$$

(*amaxN* returns the maximal elements of a set with respect to norm $|.|$). The representation error associated with cell $A[s,i]$ is simply $E(P)$, which is identical for every bitmap $P \in A[s,i]$. Thus, all such signatures having a maximal size are stored in $A[s,i]$.

Algorithm 1 presents a DP algorithm for sequence segmentation and signature extraction. The first row of the DP table is initialized in lines 4-6. Then rows are added iteratively until reaching the *min_seg* threshold. To build A_k, for $k \in [2, min\_seg[$, we just have to add the row k to A_{k-1} (lines 9-13). Finally, $A[n, min\_seg]$ provides the best signatures and related *min_seg*-segmentations (line 16).

The core of the algorithm goes from line 6 to 14. On line 6, s denotes the number of segments for the current iteration. Then, on line 7, i ranges from the number of segments to the whole sequence size. i corresponds to the end index of the subsequence that is considered for segmentation: $\alpha[1,i]$. If i is below s, then no segmentation of size s can be found in a subsequence of size smaller then s, as no segment can be empty. j ranges from the segmentation size to the number of segments. j corresponds to index of the subsequence that was optimally segmented in $s-1$ segments. Then, on line 10, a new candidate signature is added to the set *Sig* by composing the best $s-1$ segmentation of the sequence $\alpha[1,j-1]$ with the representative of the segment $\alpha[j,i]$. Finally, on line 12, we select the largest signatures within this set. The returned signature is the largest one that segments the whole sequence α into *min_seg* segments:

Algorithm 1: Dynamic Programming for segmentation-based signature extraction

Input: $\alpha = \langle T_1, \ldots, T_n \rangle$: transaction sequence of length n, *min_seg*: the minimal segmentation size
Result: *Sig*: signatures

$A[1,1] = T_1$;
/*Initialization of the first row of A*/
for $i = 2..n$ **do**
  $A[1,i] = T_i \cup A[1,i-1]$;
end
for $s = 2..min\_seg$ **do**
  for $i = s..n$ **do**
    $Sig = \emptyset$;
    for $j = s..i$ **do**
      $Sig = Sig \cup \{(\bigcup_{T \in \alpha[j,i]} T) \cap A[s-1, j-1]\}$;
    end
    $A[s,i] = \arg\max_{p \in Sig} |p|$;
  end
end
$Sig = A[min\_seg, n]$;
return *Sig*

$A[min\_seg, n]$. The dynamic programming algorithm has a complexity in $O(n^2 k)$. However, as we will see in the following example, the resulting segmentation is not guaranteed to be the optimal one.

Table 5.3 displays the progressive segmentation by Dynamic Programming of sequence $\alpha = \langle (de)(de)(abc)(abc)(abcde)(ade) \rangle$. The leftmost column gives the indices of segments. The bottom row gives the indices over sequence α as well as their associated itemset and bitmap. Other cells of Table 5.3 detail the results of operations performed by DP as formalized by equations (5.1) and (5.2). m represents the error minimization operation of segmentation. Note that signature mining looks for maximal signatures w.r.t. the number of items and thus, m is a *max* operator on signatures. For example, cell $[2,3]$ computes the best signature obtained by segmenting sub-sequence $\alpha[1,3]$ into 2 segments. There are 2 ways to segment $\alpha[1,3]$: $\alpha[1,2] - \alpha[3,3]$ and $\alpha[1,1] - \alpha[2,3]$. In the first case, the representative of $\alpha[3,3]$ (*i.e.* its associated bitmap, 11100 representing itemset $\{abc\}$) is composed with the best signature obtained for sub-sequence $\alpha[1,2]$ given by $A[1,2]$. Actually, the composition operation (denoted by $\cap$ in the table), is simply a logical AND on bitmaps. In the second case, the representative of $\alpha[2,3]$ is composed with the best signature for sub-sequence $\alpha[1,1]$ given by $A[1,1]$. The representative

Table 5.3: DP segmentation table for sequence $\alpha = \langle (de)(de)(abc)(abc)(abcde)(ade) \rangle$, with $min\_seg = 4$. To be read from bottom-left to top-right.

4				m(α[4,4]∩A[3,3]) =m(11100∩00000) ={00000}	m(α[5,5]∩A[3,4], α[4,5]∩A[3,3]) =m(11111∩00000 , 11111∩00000) ={00000}	m(α[6,6]∩A[3,5], α[5,6]∩A[3,4], α[4,6]∩A[3,3]) =m(10011∩11100 , 11111∩00000 , 11111∩00000) ={10000}
3			m(α[3,3]∩A[2,2]) =m(11100∩00011) ={00000}	m(α[4,4]∩A[2,3], α[3,4]∩A[2,2]) =m(11100∩00011, 11100∩00011} ={00000}	m(α[5,5]∩A[2,4], α[4,5]∩A[2,3], α[3,5]∩A[2,2]) =m(11111∩11100, 11111∩00011 , 11111∩00011} ={11100}	m(α[6,6]∩A[2,5], α[5,6]∩A[2,4], α[4,6]∩A[2,3], α[3,6]∩A[2,2]) =m(10011∩11111, 11111∩11100 , 11111∩00011 , 11111∩00011} ={11100,10011}
2		m(α[2,2]∩A[1,1]) =m(0011∩00011) ={00011}	m(α[3,3]∩A[1,2], α[2,3]∩A[1,1]) =m(11100∩00011, 11111∩00011) ={00011}	m(α[4,4]∩A[1,3], α[3,4]∩A[1,2], α[2,4]∩A[1,1]) =m(11100∩11111, 11100∩00011, 11111∩00011) ={11100}	m(α[5,5]∩A[1,4], α[4,5]∩A[1,3], α[3,5]∩A[1,2], α[2,5]∩A[1,1]) =m(11111∩11111, 11100∩11111, 11111∩00011, 11111∩00011) ={11111}	m(α[6,6]∩A[1,5], α[5,6]∩A[1,4], α[4,6]∩A[1,3], α[3,6]∩A[1,2], α[2,6]∩A[1,1]) ={11111}
1	m(α[1,1]) ={00011}	m(α[1,2]) ={00011}	m(α[1,3]) ={11111}	m(α[1,4]) ={11111}	m(α[1,5]) ={11111}	m(α[1,6]) ={11111}
	1: *(de)* 00011	2: *(de)* 00011	3: *(abc)* 11100	4: *(abc)* 11100	5: *(abcde)* 11111	6: *(ade)* 10011

Sequence indices →

of several sequence elements is simply a logical OR on their associated bitmaps. The best (but possibly not optimal) signature for the whole sequence α and a 4-segmentation is given by $A[4,6]$.

The best segmentation is obtained by traversing A from top-right to bottom-left, as indicated by the blue lines. The best segmentation S_{best} in Table 5.3 is $S_{best} = \langle S_1, S_2, S_3, S_4 \rangle$ with $S_1 = \langle (de)(de)(abc) \rangle$, $S_2 = \langle (abc) \rangle$, $S_3 = \langle (abcde) \rangle$ and $S_4 = \langle (ade) \rangle$, with an error of 4.

One can notice that the optimal segmentation in Table 5.3 is $S_{opt} = \langle S_1, S_2, S_3, S_4 \rangle$ with $S_1 = \langle (de) \rangle$, $S_2 = \langle (de) \rangle$, $S_3 = \langle (abc)(abc)(abcde) \rangle$ and $S_4 = \langle (ade) \rangle$, with an error of 3.

Refined Optimal Algorithm

Table 5.3 clearly shows that Algorithm 1 does not find the optimal segmentation. Intuitively, the algorithm is not able to find the optimal segmentation because in cell $A[3,5]$, we discard

the fact that α[4,5]∩A[2,3] could lead to a future optimal segmentation. Indeed, α[4,5]∩A[2,3] gives an error of 3, while α[5,5]∩A[2,4] gives an error of 2, which leads Algorithm 1 to discard α[4,5]∩A[2,3] as a possible optimal solution. However, α[4,5]∩A[2,3] is the only choice to get to the optimal segmentation. α[4,5]∩A[2,3]= 00011, α[5,5]∩A[2,4]= 11100 and α[6,6]= 10011, hence the result of α[4,5]∩A[2,3] maximizes the intersection between α[6,6] and a 3-segmentation of α[1,5]. Because Algorithm 1 does not consider the fact that α[4,5]∩A[2,3] and α[5,5]∩A[2,4] have different resulting itemsets, it cannot find the optimal segmentation. To tackle this issue, one needs to store all results that are not subsets of existing results. Therefore, these results correspond to the maximal itemsets.

Algorithm 2: Optimal Dynamic Programming for segmentation-based signature extraction

Input: $\alpha = \langle T_1, \ldots, T_n \rangle$: transaction sequence of length n, *min_seg*: the minimal segmentation size
Result: *Sig*: signatures

```
1  A[1,1] = T_1;
2  /*Initialization of the first row of A*/
3  for i = 2..n do
4  |  A[1,i] = T_i ∪ A[1,i-1];
5  end
6  for s = 2..min_seg do
7  |  for i = s..n do
8  |  |  Sig = ∅;
9  |  |  for j = s..i do
10 |  |  |  Sig = Sig ∪ {(⋃_{T∈α[j,i]} T) ∩ A[s-1,j-1]};
11 |  |  end
12 |  |  A[s,i] = maximal(Sig);
13 |  end
14 end
15 Sig = arg max_{p∈A[min_seg,n]} |p|;
16 return Sig
```

Algorithm 2 presents the pseudo-code of this optimal algorithm. The main difference with Algorithm 1 is in line 12, where the arg max operator on the size of elements in *Sig* has been replaced by the operator returning the maximal itemsets of *Sig*. The arg max operator on line 15 ensures that only valid signatures are outputted.

However, the set of maximal itemsets is larger than the set of itemsets with maximum size. This leads to higher computation costs, that will be studied in Section 6.3.

Table 5.4: Optimal DP segmentation table for sequence $\alpha = \langle (de)(de)(abc)(abc)(abcde)(ade) \rangle$, with $min\_seg = 4$. To be read from bottom-left to top-right.

4				m(α[4,4]∩A[3,3]) =m(11100∩00000) ={00000}	m(α[5,5]∩A[3,4], α[4,5]∩A[3,3]) =m(11111∩00000 , 11111∩00000) ={00000}	m(α[6,6]∩A[3,5], α[5,6]∩A[3,4], α[4,6]∩A[3,3]) =m(10011∩11100 , 10011∩00011 , 11111∩00000 , 11111∩00000) ={00011, 10000}
3			m(α[3,3]∩A[2,2]) =m(11100∩00011) ={00000}	m(α[4,4]∩A[2,3], α[3,4]∩A[2,2]) =m(11100∩00011, 11100∩00011} ={00000}	m(α[5,5]∩A[2,4], α[4,5]∩A[2,3], α[3,5]∩A[2,2]) =m(11111∩11100, 11111∩00011 , 11111∩00011 , 11111∩00011} ={11100, 00011}	m(α[6,6]∩A[2,5], α[5,6]∩A[2,4], α[4,6]∩A[2,3], α[3,6]∩A[2,2]) =m(10011∩11111, 11111∩11100 , 11111∩00011 , 11111∩00011 , 11111∩00011} ={11100,10011}
2		m(α[2,2]∩A[1,1]) =m(0011∩00011) ={00011}	m(α[3,3]∩A[1,2], α[2,3]∩A[1,1]) =m(11100∩00011, 11111∩00011) ={00011}	m(α[4,4]∩A[1,3], α[3,4]∩A[1,2], α[2,4]∩A[1,1]) =m(11100∩11111, 11100∩00011, 11111∩00011) ={11100, 00011 }	m(α[5,5]∩A[1,4], α[4,5]∩A[1,3], α[3,5]∩A[1,2], α[2,5]∩A[1,1]) =m(11111∩11111, 11100∩11111, 11111∩00011, 11111∩00011) ={11111}	m(α[6,6]∩A[1,5], α[5,6]∩A[1,4], α[4,6]∩A[1,3], α[3,6]∩A[1,2], α[2,6]∩A[1,1]) ={11111}
1	m(α[1,1]) ={00011}	m(α[1,2]) }={00011}	m(α[1,3]) ={11111}	m(α[1,4]) ={11111}	m(α[1,5]) ={11111}	m(α[1,6]) ={11111}
	1: *(de)* 00011	2: *(de)* 00011	3: *(abc)* 11100	4: *(abc)* 11100	5: *(abcde)* 11111	6: *(ade)* 10011

Sequence indices ⟶

Table 5.4 shows the execution of Algorithm 2. The main difference with Algorithm 1 is that function *m* now represents the computation of maximal itemsets instead of the error minimization operation of segmentation. This difference is especially noticeable in cell A[3,5], where we retain maximal itemsets, hence keeping the result of α[4,5]∩A[2,3]. It should also be noted that there are different optimal segmentations. This especially noticeable in cell A[3,5], where there are 3 optimal choices.

Algorithm 3: Greedy top-down algorithm for segmentation-based signature extraction

Input: α: sequence, *min_seg*: the minimal segmentation size
Result: *Sig*: signature, *Segments*: segmentation

$n = |\alpha|$;
$Segments = \{\alpha\}$;
$Sig = \bigcup_{T_i \in \alpha} T_i$;
for $i = 2..min\_seg$ **do**
    $Qual = 0; Best = \emptyset$;
    $BL = \emptyset; BR = \emptyset$;
    forall $S \in Segments$ **do**
        $\{L, R\} = BestSplit(Sig, S)$;
        $Qual1 = |Sig \cap L \cap R|$;
        if $Qual1 > Qual$ **then**
            $Qual = Qual1$;
            $BestS = S$;
            $BL = L$;
            $BR = R$;
        end
    end
    $Segments.replace(BestS, BL, BR)$;
    $Sig = Sig \cap BL \cap BR$;
end
return $Sig, Segments$

5.3.2 Top-Down Greedy Approach

As explained in Section 4.2, the quadratic complexity of the dynamic programming approach makes the analysis of long sequence intractable. To this end, several algorithms have been defined in the sequence segmentation literature to quickly approximate the optimal segmentation. Among these algorithms, we implemented the *greedy top-down* algorithm [TT06].

The greedy top-down algorithm (GTD) starts with the whole sequence as one segment, and then splits one segment at each step until reaching *min_seg* segments. The segment chosen for splitting is the one that leads to the smallest reconstruction error increase. The split point is chosen by iterating on the transactions of a segment such that the intersection of left part L and right part R with the current signature leads to the smallest reconstruction error increase.

Algorithm 3 achieves a greedy top-down search. **Lines 1 to 3** initialize the segments and compute the signature with 1 segment. **Line 4** is the main loop of the algorithm. At each step of the loop, one segment in added until we have *min_seg* segments. **Lines 7 to 16** compute the optimal segment split: the one maximizing the size of the signature corresponding to the new segmentation. The function $BestSplit$ on **Line 8** computes the optimal segment split for a given segment. Its implementation is detailed in Algorithm 4. **Lines 10 to 15** record the current best split and reset the variables if a new optimal signature is found. **Line 17** replaces the segment where the optimal split was found by the new segments resulting from the split. **Line 18** computes the signature corresponding to the new segmentation.

Unlike the DP approach, once a segment is split, it remains split for the rest of the execution. There is no guarantee to obtain the best (maximal) signature at the end of the execution. But, with a complexity in $O(nk)$, this algorithm is well-known for having good runtime performance in practice.

In this section, we have presented different signature mining algorithms using ideas from the sequence segmentation field. Two of these algorithms are based on dynamic programming ideas, while the third one is a based on a greedy approach. The algorithm based on dynamic programming returns the optimal signature, while the two other algorithms return approximate solutions. Algorithms based on pattern mining approaches are presented in Section 8.3.

Algorithm 4: function BestSplit(Sig, S)

Input: *Sig*: best signature so far, *S*: current segment
Result: The optimal segment split

```
1  Qual = 0;
2  BestL = 0;
3  BestR = 0;
4  forall k ∈ [2, |S|] do
5  |   L = ⋃_{1≤i≤k−1} T_i;
6  |   R = ⋃_{k≤i≤|S|} T_i;
7  |   Qual1 = Sig ∩ L ∩ R;
8  |   if Qual1 > Qual then
9  |   |   Qual = Qual1;
10 |   |   BestL = L;
11 |   |   BestR = R;
12 |   end
13 end
14 return BestL, BestR
```

Summary

- Signatures can be defined in both Pattern Mining and Sequence Segmentation fields
- Dynamic Programming can be adapted to mine signatures
- Approximate algorithms can be used to speed up signature mining

CHAPTER 6

EXPERIMENTS

Contents

This chapter presents experiments conducted to evaluate the different signature mining algorithms, as well as the relevance of the signature model to analyze recurrences in a sequence. The first Section presents the real datasets that are analyzed in these experiments. Next, experiments conducted on synthetic datasets are presented. Runtimes of the different algorithm presented in

the previous chapter are then compared. Afterwards, the signature model is compared to State of Art models. Finally, an analysis of the signature relevance on retail data is performed.

6.1 Datasets

The experiments are performed on different types of sequences. As the motivating example for the signature model is the retail use case, the experiments are mainly performed on retail customer purchase data. Nevertheless, signatures are applicable to other sequence data, hence I will present the different datasets that are used throughout the experiments.

6.1.1 French Retail Data

The main dataset I used to perform experiments comes from a major French retailer. This retail dataset contains anonymized basket data, collected from may 2012 to august 2014 (27 months) from customers owning a loyalty card.

As we are looking for regularities in each purchase sequence of customers, we removed occasional customers. In collaboration with the French retailer, we defined occasional customers as customers having less than 20 baskets during the 27 months. 149 942 distinct customers, worth 16.6 GB of data, remained.

The resulting database contains 3,887,979 distinct items. The retailer also provided a taxonomy that relates items to subcategories (item class). An extract of this taxonomy is presented in Figure 6.1. As the product labels are changing with packaging and discounts, we decided to mainly work on the *Segment* level of the taxonomy. This level allows us to have a detailed view of the products, while overcoming minor changes in products. We ended up with a total of 3388 *Segments*, also called item categories.

6.1.2 Instacart Retail Data

We also used a second retail dataset from Instacart: the Instacart 2017 dataset[1].

This dataset contains orders from about 200 000 Instacart customers. There are between 4 and 100 purchases per customer. As we are looking for regularities in customer purchases, we selected customers having more than 60 baskets. We only looked at customers having many pur-

1. The Instacart Online Grocery Shopping Dataset 2017, accessed from https://www.instacart.com/datasets/grocery-shopping-2017 on 05/04/2018

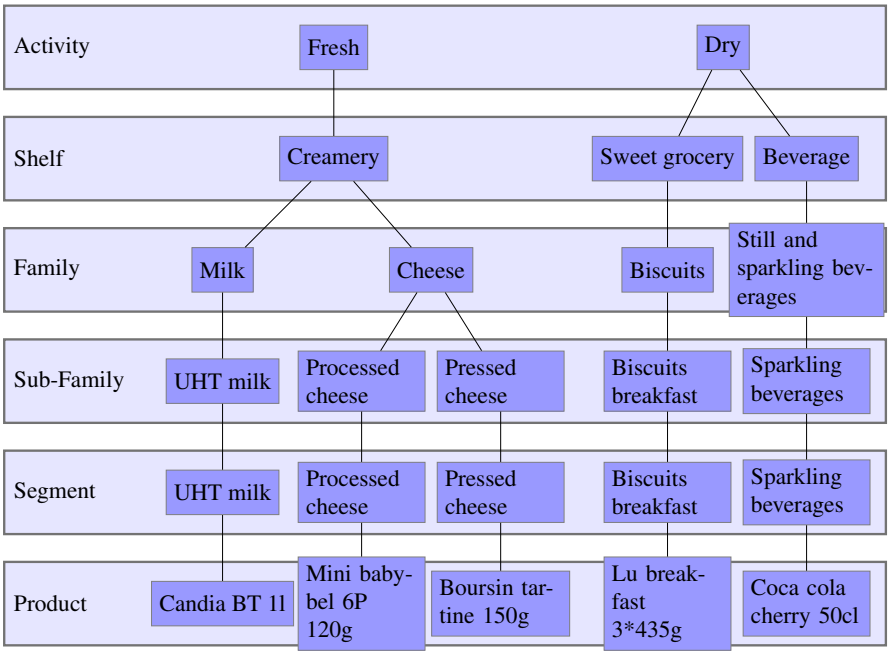

Figure 6.1: Example of product labels taxonomy

chases, as identifying recurrent behaviour on customers having few purchases is less relevant. This yielded a total of 6076 customers.

A taxonomy of products is also provided with this dataset. We worked on the lowest level of the taxonomy: the product level. It contains 49 688 products. Unlike the previous dataset, we use the product level, as the next level of the taxonomy only contains 134 categories, which does not allow to distinguish different products. As we have more products than in the French retail dataset (49 688 versus 3 388) we select customers having more baskets (60 baskets versus 20 baskets) to select regular customers.

6.1.3 TV Broadcast Data

To explore the use of signatures in different types of sequences, we also analyzed non retail datasets. One of these dataset is based on TV broadcast from a French TV channel.

The TV broadcast dataset [NG08] contains all the TV events that occurred during 23 days of May 2005 on the french TV channel *France 2*. Each TV event is timestamped and has an associated label. There are 616 different events, for a total of 7591 records. Compared to retail datasets, this dataset is more structured, as some events, such as news, commercials and weather forecast, occur at regular times each day. It should also be noted that the channel broadcast the

Roland-Garros Grand Slam tennis tournament at the end of May. Therefore the last week of the dataset contains many events associated with this tournament.

6.2 Synthetic Datasets

This part presents the experiments conducted on synthetic datasets. In our experiments, we generated three types of sequences according to a model based on the French retail dataset. The goal of these experiments is to study how well the signature model can capture retail customers behavior.

The first type of sequences is generated with no prior knowledge of a customer product preference, i.e. each product is chosen according to a uniform probability distribution over all products. The sequences of the first type contain no underlying pattern, and can be seen as baseline **random sequences**.

The second type of sequences aims at reproducing the behavior of a given customer. Therefore we generate sequences using the probability distributions over products of a real customer. We also use the transaction length probability distribution of the same customer. This second type of sequence can be seen as a **simulation of a real customer**. It should be noted that this simulation does not take into account the temporal habits of the simulated customer. It only exploits the frequencies of the products and of transaction lengths.

Finally, the third type of sequences is a mix between the two previous types. Indeed, we generated sequences by balancing the amount of noise in the customer simulation. In that case, the noise is generated following the method used in the first type of sequence. This last type of sequence can be seen as a **noisy simulation of a real customer**, where the noise is controlled by a user defined parameter. To sum this up, the first type of sequence is only made of noise, while the second type is a realistic simulation of a real customer and the third one is a mix between both of them.

6.2.1 Random Baseline Sequences

Sequence Generation

To generate the random sequences, we took the average characteristics from a sample of 150 000 customers. We found that the mean number of transactions per customer is 181, the mean number of different products bought by a customer is 242, and each transaction contains on average 12 products. Given these three characteristics, we generated sequences as follows. First,

we determined the number of transactions for the current sequence by sampling a Poisson distribution with parameter value equal to 181. Then, for each transaction, we chose its length by sampling a Poisson law with parameter value equal to 12. Finally, we generated the items of the transaction by sampling uniformly at random over 242 items, with replacement.

Results

We generate 1000 sequences following the procedure described above. We compute signatures on all these sequences, with a *RNS* (Relative Number of Segments, see Section 5.2.1) of 0.15. This *RNS* value is chosen, as it is the one that is usually used when analyzing real retail customers. All signatures on these generated sequences are empty, whereas only 2.2 % of the signatures computed on real customers are empty, with the same *RNS*. Given that the generated sequences do not contain any model of customer preferences over products (products were chosen uniformly at random), there are no regularities to detect. The fact that the signatures on these random sequences are empty shows that the signature mining does not find any interesting pattern if there is no regularity in the data.

6.2.2 Customer Simulation Sequences

Sequence Generation

To generate sequences simulating the behavior of a given customer, we first compute the histogram of the purchased products of a customer, selected at random among all customers. This provides the empirical probability distribution of the items. We also compute the distribution of the transaction lengths of the customer. Once we have these two distributions, we generate sequences following a process similar to the one described for the random generation. First, we choose the number of transactions equal to the number of transactions of the customer. Then, we sample each transaction length from the empirical distribution of the transaction lengths. Finally, we sample each item according to the empirical item distribution. It should be noted that we do not take into account the temporal regularity during this generation process. We also do not take into account correlations between product purchases. We only take into account the product frequencies and the transaction length frequencies.

Results

Simulations of 100 customers were generated. For each customer, we have 50 simulated sequences, which gives us a total of 5000 sequences. For each of these sequences, we compute the signature with a *RNS* of 0.15. The results on the synthetic sequences are quite similar to the one obtained on the real sequences. Indeed, the length of the signature of the real customer only differs by 0.2 products with the signatures computed on the synthetic sequences, on average. The Jaccard similarity between the signature of the real customer and the signature of the synthetic sequences is equal to 0.55 on average. If we compare this similarity with the Jaccard similarity between the signatures of 2 customers, which is 0.05 on average, we can see that the signatures of the synthetic sequence are close to the signatures of the real client. This demonstrates that signatures are similar with one another if the customer product preferences are similar.

Nevertheless, the resulting signatures are not identical to the signatures of the simulated customer. This is also interesting because it shows that only taking into account product frequencies is insufficient to reproduce customer behaviour. This is understandable, as the signature enforces some temporal regularity through the purchases segmentation. Methods modeling customer behaviour exclusively based on product frequencies (such as top-k items [Han+02]) are therefore not totally adapted in our case, as they cannot capture all purchase habits.

6.2.3 Noisy Customer Simulation Sequences

In this experiment, we study how the signature content evolves when we introduce some controlled level of noise in the customer simulation sequences.

Sequence Generation

To generate sequences simulating a customer, with a certain level of noise, we add a parameter λ. This parameter controls the level of noise in the simulated sequence. With this additional parameter, the process generating the sequence is similar to the one used for the customer simulation sequence generation in Section 6.2.2. The only change occurs during item sampling. Before generating an item, we choose the distribution to sample it from. We choose the uniform distribution (noise) with probability λ and we choose the customer distribution with probability $1-\lambda$. Therefore, λ controls the level of noise in the sequence. The first type of sequence corresponds to $\lambda = 1$ (only noise) while the second one corresponds to $\lambda = 0$ (pure simulation).

Results

For each λ, ranging from 0 to 1 with a step of 0.1, 50 simulations of 50 customers were generated. This gives us a total of 27500 sequences. Then, the Jaccard similarity between the signature of the real customer and the signature computed on the generated sequences was computed. Both computations are made with a *RNS* of 0.15. The results are presented in Figure 6.2. We can see that the Jaccard similarity between the real signature and the signature computed on the synthetic sequences starts to decrease when the level of noise is greater than 0.3.

Given that the Jaccard similarity between the signatures of 2 customers is 0.05 on average, we can see that with a noise level up to 0.7, the signature of the generated sequence is still very close to the signature of the real customer.

This shows that the signature computation is robust to a high level of noise, and that it selects items that are good representatives of the customers.

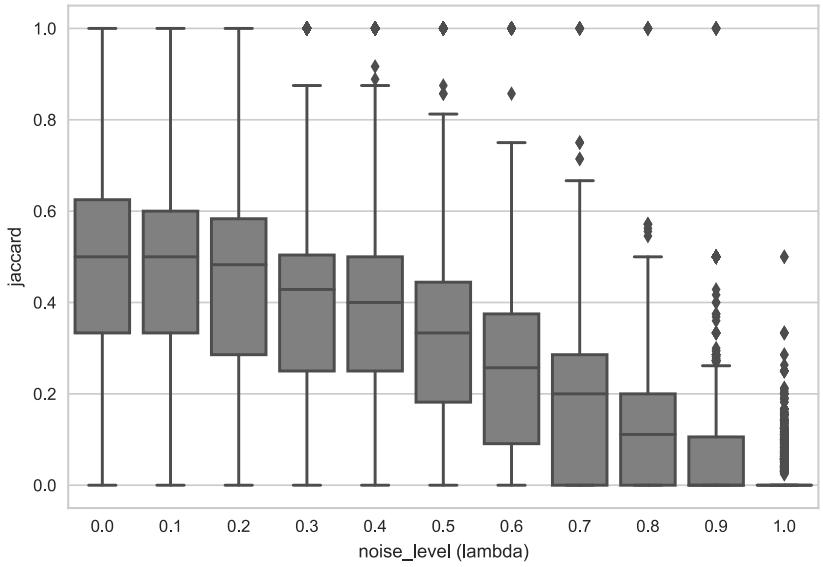

Figure 6.2: Evolution of the Jaccard similarity between the real signature and the signature computed on synthetic sequences with respect to the noise level. The greater λ, the higher the noise level.

Summary

- In synthetic sequences generated with no item preferences (uniformly distributed), signatures are empty.
- In synthetic sequences generated according to empirical items distribution of true customers, the signature is similar to the one of the true customer.
- Adding uniform noise on an existing sequence does not affect the signature too much.

6.3 Analysis of the Quality and Runtime of Heuristic Approaches

Once the datasets have been defined, the first experiment compares the Optimal Dynamic Programming (OptDP) algorithm, the Basic Dynamic Programming (BasicDP) and the Greedy Top-Down (GTD) approach to mine signatures. This comparison was performed on 10 000 customers from the French retail dataset.

The top-down heuristic algorithm in Section 5.3.2 aims at decreasing the runtime, while expecting a result not too far from the complete solution found by *OptDP*. *BasicDP* should be a compromise in terms of quality and runtimes between *GTD* and *OptDP*. One point that should be noted is that both DP algorithms outputs all the sets of products respecting the constraints of the signature, while GTD outputs only one set.

First, let us look at the runtime performance presented in Table 6.1. These runtimes were obtained on a cluster of 4 machines, each machine having 4 GB of memory and an Intel(R) Xeon(R) E5-2650 v4 CPU. Each signature computation had a timeout of 10 hours.

As expected, *GTD* is the fastest algorithm, with a mean run time of 380 ms. Then, the *BasicDP* algorithm runs about 10 times slower and finally the *OptDP* algorithm runs about 100 times slower than the *BasicDP*. When looking at median runtimes, one can see that all 3 algorithms have less runtime differences. This is understandable as both DP algorithms tend to have few very high runtimes skewing the mean towards higher values. Some runtimes for *OptDP* even exceeded the 10 hours timeout. These large runtimes are an issue, as they significantly impact the runtime of the whole experiment. This is reflected through the high standard deviation of the runtime for the *OptDP* algorithm.

Table 6.1: Runtimes of algorithms with *RNS* (Relative Number of Segments) set to 0.15.

Algorithm	Mean runtime (seconds)	Standard dev (seconds)	Median runtime (seconds)
GTD	0.38	0.41	0.26
BasicDP	4.47	18.90	0.77
OptDP	369s.45	1488.75	1.68

Table 6.2: Relative quality of the 3 algorithms with *RNS* set to 0.15.

Algorithm	Mean Relative Quality	Standard dev of Relative Quality
GTD	0.5226	0.39
BasicDP	0.9380	0.11
OptDP	0.9994	0.02

Table 6.2 presents the approximation quality of the 3 algorithms. This quality is then compared with the best quality found by all 3 algorithms on a given customer. In most cases, *OptDP* returns the largest (and optimal) signature size. However, in 8 cases, *OptDP* timed out. In these cases, the best signature is the largest (possibly non optimal) signature found by the other 2 algorithms. This explains why the Mean Relative Quality of *OptDP* is not exactly equal to one.

Overall, we can say the *BasicDP* returns a good approximation of the best signature, while *GTD* returns a rough approximation of the best result.

We performed the same experiments with *RNS* set to 0.05 and 0.1. The results for both running times and quality of the approximation were similar, with the exception of the *GTD* quality increasing to 0.77 with *RNS* set to 0.1.

Summary

- The Greedy Top-Down approach quickly finds rough approximations of the signature.
- The Optimal Dynamic Programming algorithm has very large runtimes in a few cases.
- The Basic Dynamic Programming algorithm is a good compromise, outputting good approximation in a reasonable amount of time (a few seconds per customer on average).

Given the good properties of the Basic Dynamic Programming algorithm, we prefer to use it over the 2 other algorithms. Therefore, when we refer to the **Dynamic Programming** (DP)

algorithm in the remaining of this thesis, we are referring to the **Basic Dynamic Programming** algorithm.

6.4 Comparison with State of The Art Techniques

Mining methods that extract patterns while giving some insight of regularity, go from top-k item mining [Han+02] to periodic pattern mining. Top-k items are the k most frequently bought items within all customer's baskets. However, top-k items do not provide an explicit information about purchase regularity. Yet, item frequency can be considered as a rough regularity. Moreover, top-k items are considered separately from one another, whereas the signature considers a whole set of items .

Periodic patterns [MH01] represent items that are purchased at a strict periodicity. However, some purchase delay could break the periodicity and prevent a pattern to be periodic. Signatures stand in the middle: they represent sets of items that are bought within a limited period of time and such items are bought together several times but under a non strict periodicity.

In the sequel, we compare signatures with top-k items and periodic patterns to exhibit some common and distinctive features.

6.4.1 Signatures vs Top-k Items

In this experiment, we compare the signature content with the top-k items for each customer. We compute signatures with a *RNS* of 0.15. We try different values of k for the top-k items method, and compare all of them with the signature content in Figure 6.3, on the left. Setting the value of k to the signature length for each customer is not possible in practice, as we do not know the signature length before hand. We therefore do not show experiments with this particular value of k. More elaborate methods to adapt the k value to each customer, such as elbow methods [TWH01], did not bring better results than the ones presented in Figure 6.3-left.

The Jaccard similarity between the signature and the top-k items of most customers is between 0.5 and 0.3. This means that top-k items and signature products overlap partially. When the k value is low, the number of top-k items is significantly lower than the number of items in the signature. This leads to a low Jaccard value, even though most of the top-k items are included in the signature. A similar behavior is observed for large values of k, where the top-k contains more items than the signature, leading to a low Jaccard value. For values of k close to the mean signature length: between 5 and 10 items, the Jaccard goes higher, as both sets have a

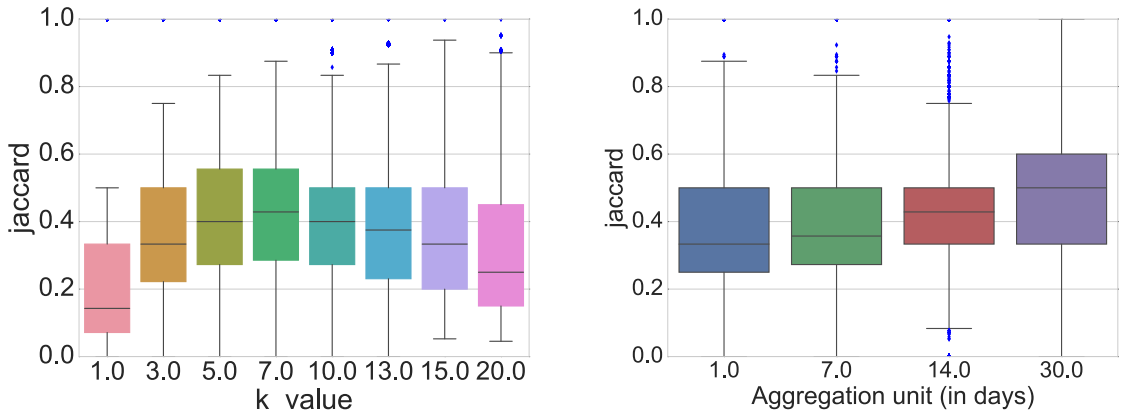

Figure 6.3: On the left: Jaccard similarity between the signature and the top-k items, for different k values. This has been computed on 149 942 customers. On the right: Jaccard similarity between the signature and the longest periodic pattern, for different time scales. This was computed on 20 000 customers.

similar size.

Overall, the signature overlaps partially with the top-k items, and the main source of difference comes from the fact that the number of items in the signature changes for each customer, whereas it is constant for all customers in the top-k computation. The number of items in the signature could therefore be seen as a way to estimate a relevant value of k for the top-k items of a given customer.

Another source of difference between signatures and top-k items is the fact that items that are very frequently bought during a short period of time do not appear in the signature, while they are more likely to appear in the top-k items.

To further investigate the differences caused by this phenomenon, we computed the similarity between the signature and the top-k, where k is equal to the size of the signature. While this setting is not fair towards the signature, as we do not know in advance the size of the signature, it allows us to understand how different the signature is from the top-k. To deepen this analysis, we also allowed some mismatch between the top-k and the signature, by computing the similarity between the signature and top-k items, with k larger than the signature size. The results are presented in Figure 6.4, for the retail dataset (grey curve). One can see that the signature is quite similar to the top-k, especially when a few (up to 4) mismatch are allowed. This similarity between the signature and the top-k is understandable, as in retail, products that are bought a lot of times (they appear in the top-k) are usually bought over time (they therefore appear in the signature).

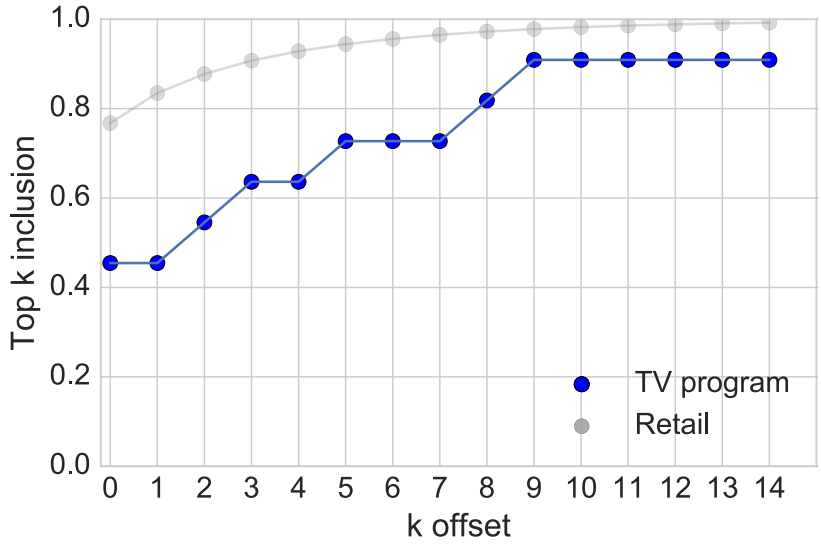

Figure 6.4: Ratio of signature items included in top-k items for values of k equal to the signature size plus a varying offset on two datasets: retail and TV.

To better understand how the signature is linked to the top-k, we did the same experiment on the TV broadcast dataset. This dataset differs from the retail one, as it contains events that appear a lot of times in a short period of time (the ones associated with the Roland Garros tennis Grand Slam). Our expectation is therefore that the signature will be less similar to the top-k on the TV dataset than on the retail dataset. The similarity is plotted in Figure 6.4, on the blue curve. The results are, as expected, that the signature is more different from the top-k in the TV broadcast dataset. More generally, the more the signature is similar to the top-k, the more regular the dataset is. The similarity between both sets decreases if the dataset contains peaky events, that appear in the top-k but not in the signature.

Another noticeable information is that the frequency is a good, but not perfect, approximation of regularity in the retail case. This is mainly because the behavior of customers is regular, with a slow purchase rate. In that setting, purchasing an item many times, usually implies buying it regularly over a large period of time.

6.4.2 Periodic Patterns Comparison

In this experiment, we compare the signature content with the periodic patterns for each customer. We used an algorithm that allows gaps between consecutive occurrences of periodic patterns [Cue+12].

As periodic patterns can only be found in a single time scale, a preprocessing step that aggregates the transactions on a given time scale (e.g., merge all transactions at the given granularity) is required.

As we do not know in advance what is the relevant time scale for each customer, we are using 4 time scales to compute the periodic patterns: daily, weekly, bi-weekly and monthly purchases. For each time scale, we computed the Jaccard similarity between the longest periodic pattern and the signature. We chose the longest periodic pattern as the signature finds the longest regular pattern. If several longest periodic patterns are found, we take the one that has the largest Jaccard similarity with the signature.

The results are presented in Figure 6.3, on the right. In this figure, we can see that the Jaccard similarity between the signature and the longest periodic pattern is mostly between 0.3 and 0.45. This means that these two sets have common elements, but still differ. Further analysis showed that the longest periodic pattern is almost totally contained in the signature. This means that the signature is composed of most items from the longest periodic pattern.

This periodic part of the signature represents between one third and one half of the total signature. The remaining part of the signature contains items that are not periodic but that are regularly bought. This highlights the flexibility of the signature, as it manages to capture periodic products, while also capturing non periodic regular purchases.

Signatures capture non periodic regularities because their segments can be of arbitrary length. More specifically, each customer signature segment can contain multiple baskets, and can therefore span on different time scales. On the other hand, periodic patterns have a fixed segment length and cannot span on different time scales. To illustrate this difference, we plot the coefficient of variation (mean divided by standard deviation) of the segment size for each customer in Figure 6.5, on the left. Most customers have a coefficient of variation greater than 0.4, which means that most customers have variations in their purchase rhythms. Almost no customer has a coefficient of variation equal to zero, whereas all customers have a coefficient of variation of 0 for periodic patterns by definition. Nevertheless, the coefficient of variation remains mostly below 1, which means that customers show a regular purchase behavior. Therefore, the signature segment still captures a regular behavior of the customer. This shows that introducing flexibility in the period allows us to capture more regular products than existing methods (see Figure 6.3-right), while capturing a regular behavior (see Figure 6.5-left).

Because signatures are more flexible, their detected temporal regularity can be different than the one found by periodic patterns. To compare the period found by both methods, we compared the difference between the mean segment size of the signature, with the largest period of the

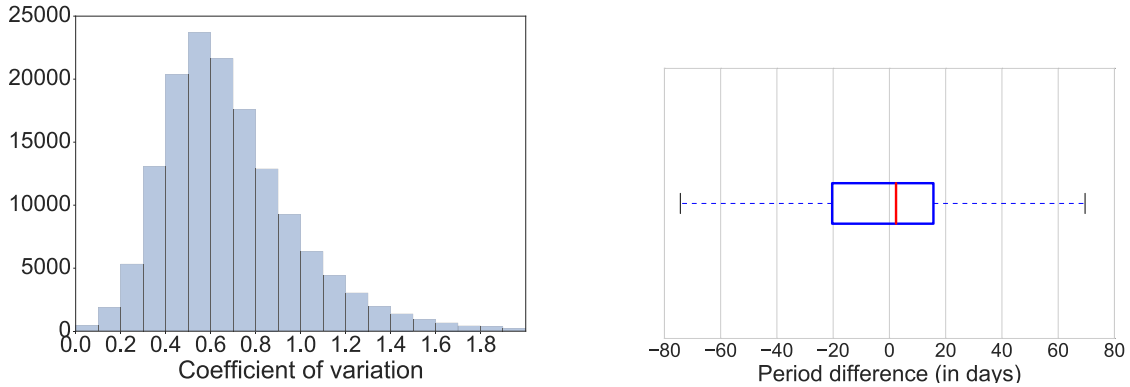

Figure 6.5: On the left: Distribution of the segment length coefficient of variation of 149 942 customers. On the right: difference between the signature period and period of the most similar periodic pattern.

periodic pattern that is the most similar with the signature (according to the Jaccard similarity). We choose the largest period of the periodic pattern, because signatures segments are as large as possible. This effect is due to the fact that segments have to cover the whole sequence. The results are presented in Figure 6.5, on the right. We can see that most periods found by the signature are close to the period found by the most similar periodic pattern. While there can be some differences between both periods, these differences are usually contained within a reasonable time span (20 days).

To summarize, signatures are able to find regularly purchased products, whether they are periodically bought are not. The flexibility of the regularity definition of the signatures allows us to find these products without any pre-processing step. Moreover, signatures are able to find the underlying period of customers, that is consistent with the one found by periodic patterns. Signatures therefore find the time regularity of a customer, along with the regular products. This regularity cannot be totally captured by existing methods.

Summary

- The link between the signature items and top-k items gives an idea about the link between frequency and recurrence in the sequence.
- The signature contains most periodic patterns having a periodicity close to the signature mean periodicity.
- The signature captures regularities that can not be captured by periodic patterns.
- Signature segments lengths suggest that signatures are indeed adapting to the sequence rhythm, while keeping a regular segment length.

6.5 Insights from Signatures

6.5.1 Individual Insights

As shown in the previous section, signatures group transactions into segments to find a set of regular products, that can not be totally found by existing methods. More specifically, let us consider the case of a real customer from the French retail dataset (named A). The store visits of customer A are represented in Figure 6.6, on the top. For comparison, we also consider customer B, from the French retail dataset, whose store visits have a signature identical to her largest periodic pattern (shown in Figure 6.6 on the bottom). The comparison of both Figures clearly shows that customer A has no clear buying pattern, while customer B has a clear buying pattern: he/she buys his/her groceries almost every Saturday.

Nevertheless, by computing the signature on customer A, we are able to detect her underlying period. Indeed, her signature contains 9 products: *Biscuits, Hazelnut spread, cheese, frozen meat, pasta, cream, butter, ham* and *chocolate powder*. Only some of them are bought during the same store visit, and these purchases usually spread over 4 transactions, for a segment length of 2 weeks on average. Among these products, some of them have a periodic buying pattern (*pasta, ham* and *hazelnut spread*), while the others are bought more sporadically. Nevertheless, this whole set of products has consistency and is related to meal and break food for children. The signature was therefore able to identify the purchase rhythm (both period and products) of a customer who had no clear buying pattern when using existing methods.

Signatures can also help marketers to answer the problem of finding the most appropriate

time and products to give a coupon on, for a given customer. To achieve this targeted coupon policy, it would be interesting to be able to know what kind of products this customer is likely to buy in the next visits, to be able to give this customer targeted coupons. Thanks to the signature, we can provide the marketer with information about the time and content of next purchases.

Indeed, if this customer has purchased *Biscuits, cheese, frozen meat, cream* and *butter* over 2 transactions in a week, we know from the signature that this customer is likely to be buying *Hazelnut spread, pasta, ham* and *chocolate powder* in the next 2 transactions over the next week. Indeed, the signature tells us that this customer has the habit of buying *Biscuits, Hazelnut spread, cheese, frozen meat, pasta, cream, butter, ham* and *chocolate powder* in 4 transactions over 2 weeks. As we are observing a portion of a signature segment, we can guess the products that are likely to be bought in the next week. This information is of prime interest for retailers, as they could then target their ads on the right products for each customer. It should be noted that periodic patterns would have missed the part related to break food for children, as only *pasta, ham* and *hazelnut spread* were considered periodic.

6.5.2 Temporal Regularities

We also look at the link between the temporal and the product dimensions of the signature. The signature length (product dimension) depending on the segment length regularity (temporal dimension) is represented in Figure 6.7. We can see that the signature length is increasing with the time span of the temporal regularity. This is understandable because the larger the periodicity, the more products are likely to be bought.

Another interesting aspect of this plot is that one can notice peaks for every multiple of 7. This is interesting because it shows that customers that have a weekly based regularity are more likely to buy more products. Even though we do not have a clear explanation of this phenomenon, we can link this behaviour with the fact that customers who have a weekly based regularity are more stable. Indeed, we plot in Figure 6.7 the evolution of the standard deviation of segment lengths, depending on the median segment length. We can notice that this standard deviation sharply decreases for each multiple of 7 (except for 7 itself, where the decrease is not noticeable). Therefore, we can make the hypothesis that customers with a weekly based regularity are “regulars”. Hence, they buy more products regularly and come more regularly to the store than an average customer.

In Figure 6.7, when the median segment length is 0, the mean signature length seems to be slightly lower than expected. This slight anomaly can be explained by the fact that customers with a median segment length of 0 have a segment made of a single receipt. Customers replen-

ishing all their favorite products in one receipt are likely to be occasional customers, as they do not regularly buy products with different replenishing rates. Indeed, if products with different replenishing rates are bought together, the segment length will increase to take into account the fact that signature products were not bought during the same store visit.

To summarize, we can see that studying both temporal and product dimensions of the signature allows us to identify typical customer behaviors. While these behaviors are expected to appear in the data, the fact that the signature is able to find them is an indirect validation of the relevance of this model. Moreover, the signature is able to tell whether a customer is likely to be a regular one, based on the segment information. One interesting experiment that could be made is to verify that these regular customers identified by the signature actually matches the regular customers identified by the retailer analytics team. However, our collaboration with the French retailer ended before we could perform this validation.

6.5.3 Purchase Habits

The experiments presented so far mainly analyze the temporal dimension of the signature. Signatures are not limited to finding the underlying purchase rhythm of a customer, they also find the product purchase habits. We analyze the purchase habits of customers by comparing the signature content of customers having a different purchase rhythm. The results are presented in Table 6.3.

To compare the signature content of 2 groups with different temporal regularities, we compute the frequency of each signature item in each group. Let Sig_1 be the signature items of group 1. The frequency of an item $i \in \mathscr{I}$ in Sig_1 is equal to $\frac{\sum_{S \in Sig_1} \mathbf{1}_{i \in S}}{\sum_{S \in Sig_1} |S|}$.

Table 6.3: Examples of signature content differences between customers with different purchases rhythms.

Group 1 rhythm	Group 2 rhythm	Product name	Group 1 frequency	Group 2 frequency
0 days	14 days	Shopping bags	0.037	0.005
0 days	14 days	Beers	0.017	0.004
7 days	21 days	Toilet paper	0.005	0.011
7 days	21 days	Yoghourts	0.057	0.040
7 days	28 days	Sponge	0.008	0.015
7 days	28 days	Cat food	0.011	0.005
7 days	28 days	Toilet paper	0.005	0.011
7 days	9 days	Shopping bags	0.009	0.013

In Table 6.3, we present the largest differences in the signatures content of customers with different purchase rhythms. The first 2 rows compare customers who replenish their signature in a single visit to the store, e.g. customers with a rhythm of 0 days, with customers replenishing their signature in 2 weeks. The main differences are observed on *shopping bags* and *beers*. These 2 products are more frequent (up to 6 times) in the signature of customers with a purchase rhythm of 0 days. The difference in the purchase of *shopping bags* can be explained by the observation made in Section 6.5.2: customers with a purchase rhythm of 0 days are likely to be non regular customers. Hence, they are less likely to bring their own shopping bags when shopping, as they are less used to purchasing regularly in the store. While this explanation is difficult to verify in practice, it seems to be a likely explanation. The higher percentage of *beers* in signatures could be linked with a profile of occasional buyers, such as students, who mainly come to the store to buy their beverages.

The remaining rows of the table mainly compare customers having a weekly rhythm with customers having a 2 or 3 weeks rhythm. In both cases, we chose to compare customers with an exact weekly basis, as they are exhibiting a more regular behavior than other customers (see Figure 6.7). The overall information is that customers with a weekly rhythm have fewer products with long replenishing periods, such as *Sponge* and *Toilet paper*, in their signatures compared to customers with a larger rhythm. The opposite tendency is observed for fresh products, such as *Yoghourts* and *Cat food*, that are more present in signatures of customers with a faster purchase rhythm.

This simple analysis shows that comparing the signature content of customers having different temporal regularities can provide knowledge to the retailer. For example, one could try to identify non regular customers by looking at whether shopping bags are in their signature. If so, one could decide to perform specific actions towards these customers, to try to make them purchase more regularly from the store.

Summary

- Signatures are able to find regularities, even in seemingly non regular sequences.
- The analysis of signatures segments shows different profiles of customers, highlighting regular customers.
- Using both the segment and items information of the signature provides practitioners actionable knowledge.

6.6 Signatures: A Naming Explanation

The choice of the word *Signature* to name the new pattern mining model I introduce in my thesis suggests that it can uniquely identify the sequence it represents. Indeed, a physical signature usually uniquely identifies the person who made the signature.

I here briefly present the experiment that made us name our model a signature, or at least provides a decent explanation for the name.

6.6.1 Identifying Customers

The goal of the experiment is to reliably identify a customer using his signature. To do so, we selected a subset of regular customers from the French retail dataset: 30 000 customers having more than 80 receipts over 27 months of purchase history. To give a rough approximation, this corresponds to customers coming every week to the store.

Protocol

To assess that a signature can uniquely identify a client, the idea of the experiment is that if we know a client signature on a given period of time, then we are able to identify this client on a new period of time, by comparing the signatures on these two periods.

More precisely, we compute the signatures of n clients over a period of 12 months. We therefore have n clients that are each associated with a signature. Then, we compute the signature of these n clients over the 12 following months. For this second signature computation, we do not associate each client to its signature.

Instead, each signature of the second period is associated with the client that has the closest signature (using a Jaccard distance) from the first period. Ideally, the client associated with the signature of the second period will be the one we computed the signature on. This experiment therefore aims at verifying that the signature contains enough information about each client to uniquely identify him. The experiment design is illustrated in Figure 6.8.

When using 2 segments to compute the signature, over 95.6% of the the label were correctly found by the method. When using 4 segments, the performance dropped to 85.7%. This is understandable, as signatures with more segments contain items that are more frequently bought. These items tend to be similar for most customers: Cheese, Pasta, Orange Juice.... Hence, it is more difficult to capture the uniqueness of each customer when looking only at products that are more frequently purchased.

As lowering the number of segments improves the performance of the method, one could argue that using all products instead of just the signature products might be a better method to identify a customer.

To test this idea, we decided to try a more standard approach such as Naive Bayes. Naive Bayes is a classification method but it does not aim at modeling a client behaviour. To apply Naive Bayes to this task, one first has to transform the data into vectors. To do so, we associate each client to a vector counting the number of purchases for each product over a given period of time. Products that were bought less than "the number of segment" times where discarded, to mimic the constraint of the number of segments in signatures.

Then, a first period of time was used for training and a second one was used for testing. The overall performances of the Naive Bayes was comparable to the signature, as we obtained a recognition rate of 95.5 % with a number of segments equal to 2. Changing the number of segments had an effect similar to the one observed for signatures.

To summarize, signatures are indeed able to identify a customer, if the number of segments is set to a low value. However, we do not insist on this particular aspect of the signature, as other simple methods achieve similar results. Nevertheless, as this experiment was one of the first we conducted, it had an influence on the naming of our model.

Summary

- Signatures can reliably identify a single customer.
- This ability is not specific to the signature, but had a large influence on the naming of the model.

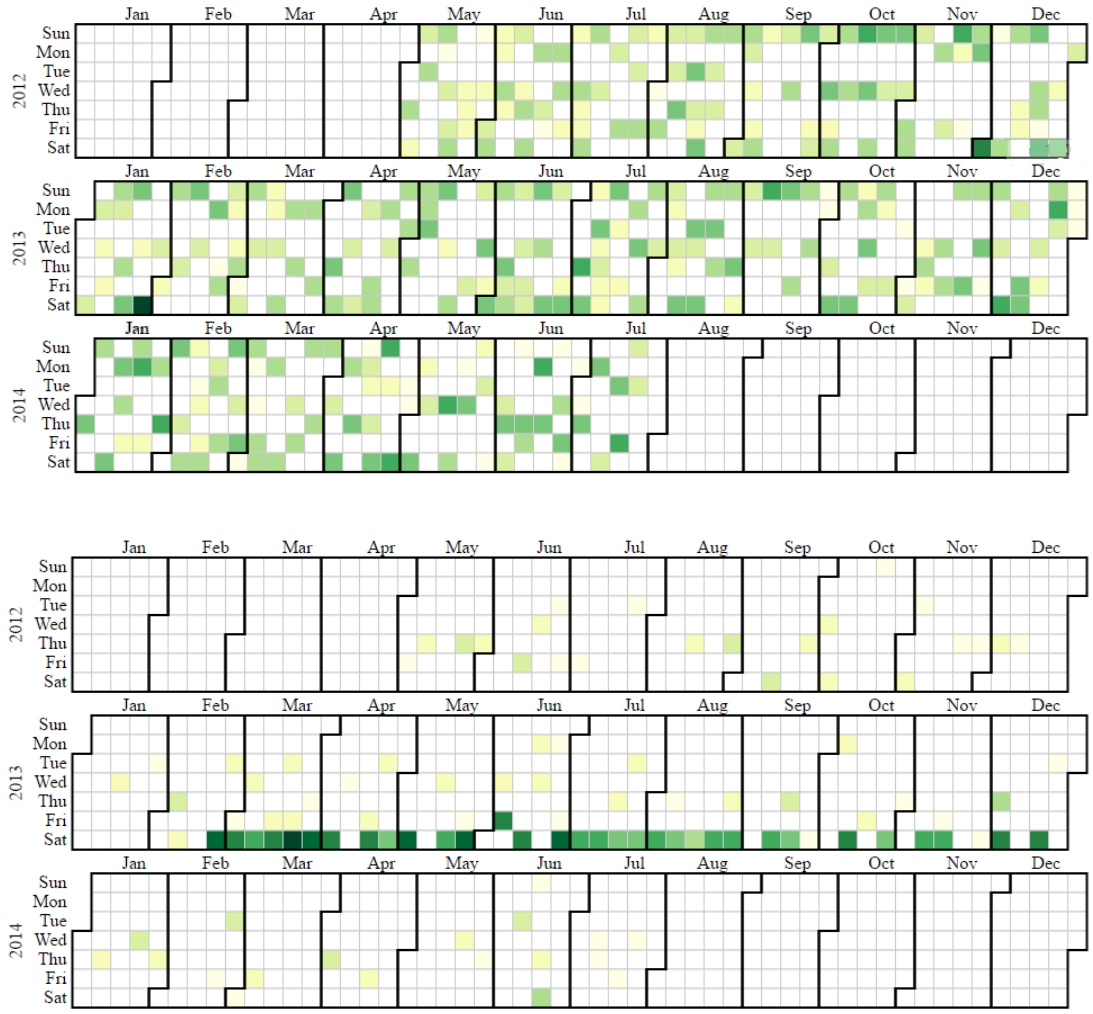

Figure 6.6: Transactions of a non periodic customer (A), on the top, and periodic customer (B), on the bottom. Each green rectangle represents a visit to the store. The darker the green, the more products were bought during that visit.

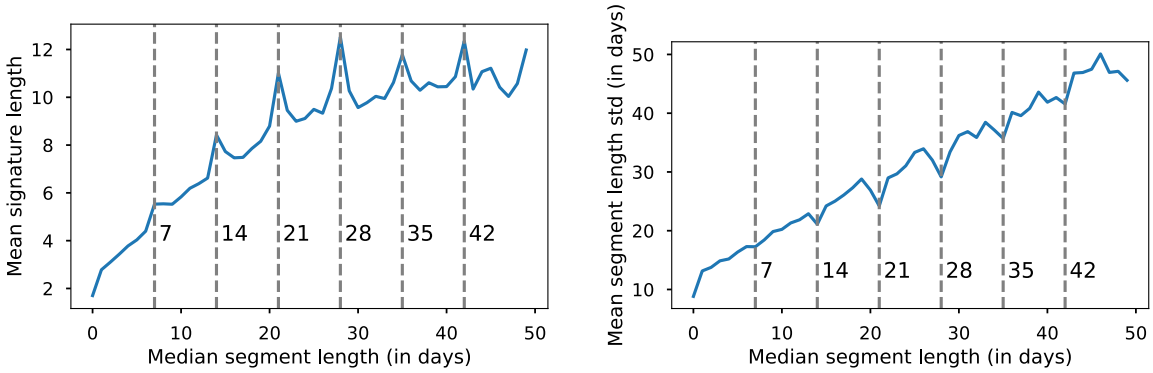

Figure 6.7: Left: evolution of signature length wrt median segment length. Right: evolution of segment length standard deviation wrt median segment size.

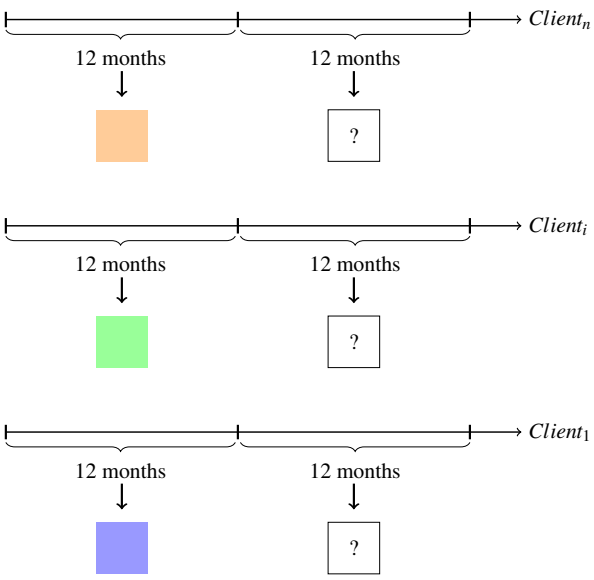

Figure 6.8: Design of the experiment to characterize a client. Each square is a signature and the color represents a client.

CHAPTER 7

SIGNATURE USE CASE: DETECTING ATTRITION

Contents

In this chapter, I present a typical retail use case of signatures: detecting attrition. So far, we have seen how signatures are defined, and how they can be used to detect individual habits. We have also seen how signatures can be used to analyze groups of customers.

However, we did not yet show how signature can be used to tackle practical issues. This chapter aims at illustrating the relevance of signatures to tackle real use cases, such as attrition detection.

7.1 Attrition Definition

The attrition represents the loss of customers to competitors. While this basic idea is simple to understand, there is no clear definition of the attrition in the literature. In the grocery retail context, defining attrition is even more complicated because customers can buy from different stores [JMB00] and defining attrition in such cases is not clear. Our definition of attrition is therefore the one that the French retailer we worked with is using. They defined the attrition as a drop in the number of visits or basket amounts over a significant period of time.

Attrition should not be studied on a short period of time because punctual events (such as vacations) could then be considered as attrition. Furthermore, attrition does not include sudden loss of customers because they are likely to be linked to other sudden events such as moving to another location. These customers should not be considered in attrition because the retailer cannot take actions to retain them. The main challenges of attrition are finding explanations for the attrition and predicting whether a customer is likely to leave the store definitely.

7.2 Visualization

To get an intuition about the relevance of signatures to analyze attrition, a first step is to visualize if signatures are evolving differently for a customer in attrition versus for a stable customer. While these visualizations do not provide a quantitative analysis of the relevance of signatures for attrition analysis, visualizing the model that is studied is helpful to get an intuition about what is differentiating a stable customer from a customer in attrition. Furthermore, it can help understand what are the limits of the model and what causes the attrition prediction to fail on some customers.

7.2.1 Customer Selection

Because we want to study attrition, we are not interested in all clients in the database. A first step was to select the relevant clients for the attrition analysis.

We are mainly interested in two types of clients. First, the one that were stable for a given time (generally about a year) and who became in attrition afterwards. These clients are interesting because we can study the evolution of the signature while a client is changing his behaviour. Nevertheless, defining attrition is difficult, so selecting clients that fit this profile is rather difficult. To tackle this, we consider that a client is in attrition if his number of visits to a store and

its purchases amount are significantly lower than previously. Different levels of significance can be chosen to select clients with different strength of attrition.

Figure 7.1 shows a typical example of the attrition profile. Indeed, the customer depicted in Figure 7.1 has a stable purchase amount from March 2012 to August 2013. Then, both purchase amounts and number of visits gradually drop, until June 2014, where the customer definitely left the store.

The second type of clients we are interested in are clients that are stable the whole time. These clients can help us to compare the evolution of the signature for a stable client with the one of a client in attrition. The expectation is that the signature for a stable client will remain stable, whereas the one for a client in attrition will significantly change.

Figure 7.2 shows a typical profile of a stable client. The customer has very similar values for both the purchase amount and the number of visits from March 2012 to May 2014. There are a few fluctuations in the customer purchases, but overall, this customer is still a regular customer at the end of the study period.

To obtain these profiles, we worked with the French retailer so that they could give us clients that are of interest for them and that fit the profiles described above.

Protocol

1. First, only customers coming regularly to the store are selected. These clients were already selected through a customer segmentation done by the French retailer.

2. Then, the clients that were not stable were discarded. The stability was defined in terms of standard deviation and range of the monthly purchase amount and monthly number of visits. Unstable customers are typically the ones having a few months of inactivity between two periods of regularity.

3. Finally, customers that are in attrition in the last months were selected. To identify stable clients that are in attrition in the last months, the evolution of the monthly purchased amount was studied. Customers who had the 30% drop of monthly purchase amount after a year and a half of history were labeled as in attrition. Finally, the remaining clients were labeled as stable customers over the whole period of time. This filtering process yielded a list of customers that were stable over a year and a half and in attrition afterwards, and a list of customers who were stable over the whole period.

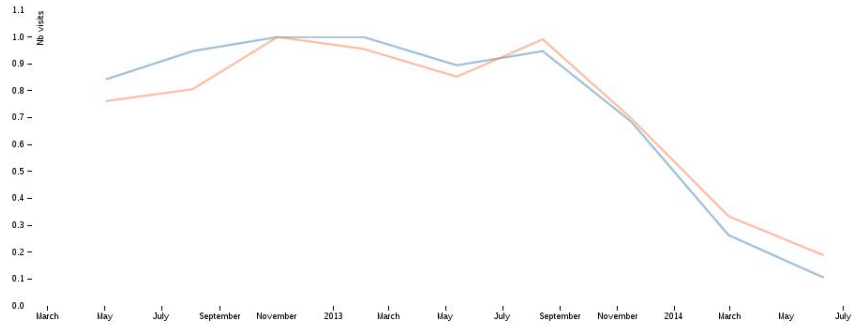

Figure 7.1: Profile of a customer stable for 16 months and in attrition afterwards. The blue line represents the number of visits each 3-months, relative to the maximum number of visits during a 3-months in the 27 months of history. The orange line represents the purchases amount during a 3-months period.

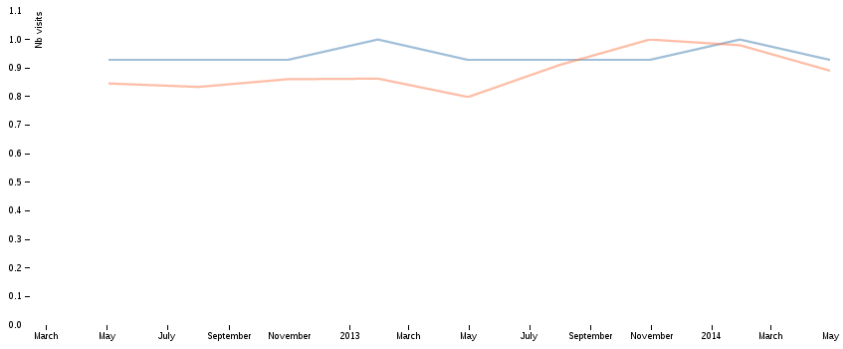

Figure 7.2: Profile of a stable customer. The blue line represents the number of visits each quarter, relative to the maximum number of visits during a 3-months in the 27 months of history. The orange line represents the purchases amount during a 3-months period.

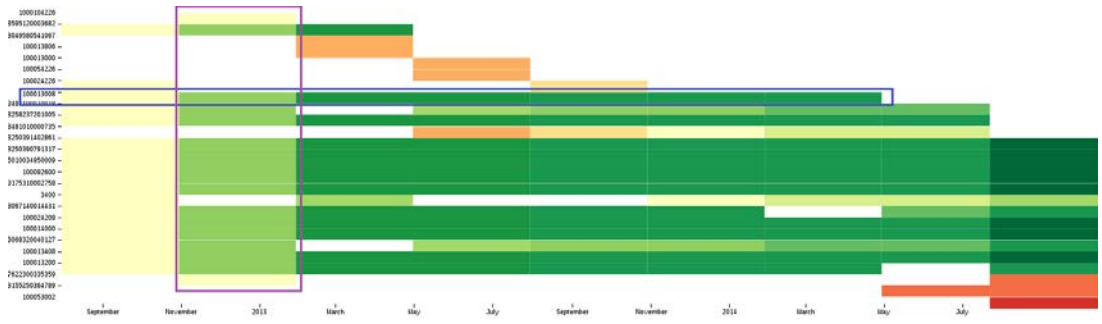

Figure 7.3: Plots the evolution of the signature for a very stable client. Each horizontal line corresponds to a product (the y axis). The evolution of the product labeled 100013008 is highlighted in a blue rectangle. Each vertical block represents a signature for 2 months. The signature of the second period is highlighted in a purple rectangle.

7.2.2 Signature Evolution Visualization

To visualize the evolution of the signature on these two types of profiles we sliced each customer sequence into consecutive periods of 2 months. Then, we compute the signature on each of these 2 months period and look at the product that are in the signature in each period. Different period duration have been tested but two months appeared to be the most appropriate. Indeed, it is the good compromise between being too sensitive to one basket change and being able to detect changes fast enough.

As stated in previous sections, the attrition is likely to be linked to customers' behaviour evolution. Signatures capture a customer's behaviour at a given time, so studying the evolution of a signature is likely to reveal insights about whether a customer is in attrition or not. The basic idea is that if a customer signature is changing, then he is likely to be in attrition.

To have a better idea of how a signature is evolving for the two selected customer profiles, we plot the signatures computed on each period of time (every two months in the following examples) in Figure 7.3. The more a product has been present in previous signatures, the more green its corresponding rectangle will be.

In Figure 7.3, one can see that most of the products belonging to at least one signature are present in almost every signature. This is visible because for each signature, most of the products are colored in green, which means that they were also present in previous signatures. The stable customer behaviour plotted in this figure is ideal and is not representative of all stable clients. A more representative evolution of a stable client is shown in Figure 7.4. We can see that stable clients have signature products that change from one period to another (the red rectangles), but products that are in many signatures (the green lines) tend to occur in all signatures.

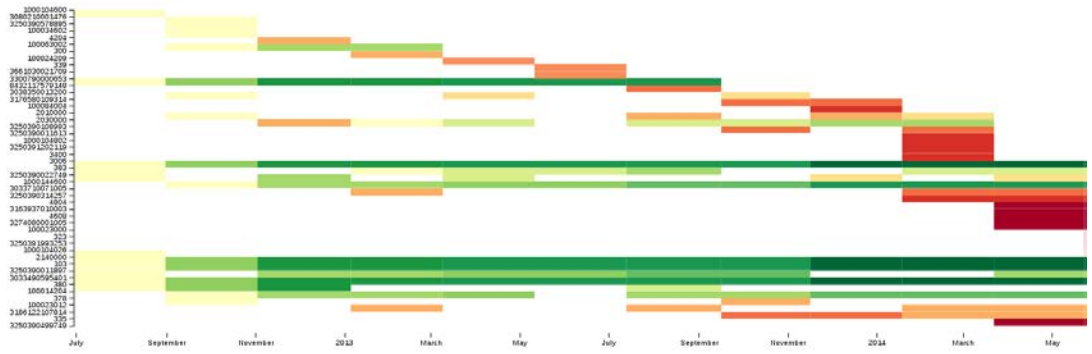

Figure 7.4: Plots the evolution of the signature for another stable client.

Figure 7.5: Plots the evolution of the signature for a client in attrition. This client is stable the first 16 months and is in attrition afterwards. Significant products loss are highlighted in red. In this plot, signatures are computed every 3 months.

Generally, we can see in Figures 7.3 and 7.4 that stable customers tend to have a set of products that are present in almost every signature. While this stability depends on how stable a customer is, it usually represents at least a quarter of all the products present in the signature.

Regarding stable customer behaviour visualization, the intuition is that even if stable clients do not have identical signatures from one period to another, a subset of products remains in almost every signature. To validate this intuition, it must be compared to the evolution of signatures of customers in attrition.

The same type of plot for a customer in attrition is represented in Figure 7.5. The first noticeable difference between Figure 7.3 and Figure 7.5 is that for the customer in attrition, the plot is sparser. Indeed, several products appear only two or three times in a signature in figure 7.5. This means that the customer is frequently switching between products and does not exhibit a strong stable behaviour.

Furthermore, this customer is progressively losing some of the few products that were fre-

quently present in signatures. These losses are highlighted in red in the plot and appear when frequency and monetary indicators start to suggest that attrition is beginning.

These visualizations show that stable customers have a subset of products that are present in almost every signature. While stable customers do change some products in their signature over time, only a few portion of products present in most signatures are lost over time. On the other hand, customer in attrition generally loose a noticeable portion of the products frequently present in signatures at the moment of attrition. Some of these customers also exhibit a non stable behaviour that is not detected by monetary or visit frequency indicators. Even if these visualizations do not prove that signatures can accurately determine whether a customer is in attrition or not, they provide an intuition about how signatures can be used to detect attrition.

7.3 Stability Criteria

In 7.2.2, the visualization of the three customers profile suggests that stable customers could be discriminated from those that become in attrition by looking at products that remain in signatures computed every k months ($k = 2$ was identified as an interesting starting point). Moreover, clients in attrition are likely to have an unstable behaviour prior to the attrition, which means that they tend to have products that appear only a few times in the computed signatures.

Validating this intuition can be achieved using different methods and the one that is presented here is one possible approach. The solution that has been developed uses a structure that enforces the fact that the more products are present in previous signatures, the stronger the attrition signal is when they disappear from the signature.

The structure developed to detect attrition can be seen as a "meta signature", built on top of all previous signatures, where each product is associated with a significance.

Let c be the number of previous signatures that contain p and l be the number of signatures that do not contain p. The *significance* s of a given product p can be defined as follows: $s(p) = \gamma^{c-l}$ if $c > 0$ and $s(p) = 0$ otherwise. γ is a user defined constant that defines how the significance of a product is evolving with the number of signatures containing it. Typically, we have $\gamma > 1$, so that products that appear in many signatures have a high significance. Let us name $\mathscr{S}_t$ the set of all products at time t.

To detect attrition, $\mathscr{S}_t$ is compared to $\mathscr{S}_{t+1}$. The stability of a given customer c_i at time $t+1$ is defined as follows: $stability(c_i) = 1 - \frac{\sum_{p \in \overline{\mathscr{S}_{t+1}}} s(p)}{\sum_{p \in \mathscr{S}_t} s(p)}$, where $\overline{\mathscr{S}_{t+1}}$ contains all products p that are not in the signature computed at $t+1$. The stability of a client at time $t+1$ can be seen as the

significance of all products that are not in the signature at time $t+1$, over the significance of all products. If a customer has always the same signature, the stability equals 1. Otherwise, this stability decreases according to the significance of the products that this customer lost in the latest signatures.

7.4 Experiments

7.4.1 Customer Selection

Since we want to analyze the attrition, two types of clients were selected. The first profile includes customers that are stable for a given period of time and that are in attrition (according to the French retailer) afterwards. The second type of customers are stable during the whole study period. The stability is defined in terms of frequency and monetary stability. These two types of profiles are the same as the one presented in Section 7.2.1 and their profiles are shown in Figures 7.1 and 7.2.

Clients that are not stable are not of interest because they are less profitable for the retailer. Therefore, the main purpose of the attrition analysis is to detect stable, and therefore profitable, customers that become in attrition. The two types of selected profiles are here to discriminate customers who remain stable from those who become in attrition.

7.4.2 Data Preparation

For this experiment, the receipt database was used along with the product taxonomy. This taxonomy was useful because there are 3 887 979 products and performing the analysis on such a large amount of products led to a very unstable signature evolution, even for stable clients. The abstraction level chosen to perform the experiments is the segment. There are 3388 segments so this level of abstraction can still model subtle changes in a customer behaviour. This level was chosen because it brings more stability into the attrition analysis. Indeed, if a customer replaces a product by another similar one, this behaviour should not be a strong signal of attrition. Using the segment level, the attrition detection method is more accurate. Moreover, this abstraction level was suggested by the retailer, who performs analysis on this dataset on a daily basis.

7.4.3 Protocol

To assess that the stability measure of a customer can discriminate stable customers from customers in attrition, the following experiment has been conducted: we select n stable customers over 18 months and in attrition afterwards and n stable customers over the whole purchase history. The goal is to predict the label of each of these customers, based on their stability. The first n customers are labeled as *attrition*, and the n remaining ones are labeled as *stable*.

For each of these $2n$ customers, we compute their signature and their stability every m months. Then, we compute the area under the ROC (AUROC) curve, using stability value. This ROC curve is based on the different possible values of the stability criteria. Each value is used as a prediction threshold and is associated to a point of the ROC curve. The AUROC measures how suited the stability is to separate customer in attrition from stable customers. The closer the AUROC is to 1, the better.

This experiment has been conducted for different values of m and γ (the significance factor).

7.4.4 Results

The results of the area under the ROC and Precision-Recall (AUPR) curves for $m = 2$ and $n = 1000$ are shown in Figure 7.6. These results are promising and show that the stability measure defined above can separate stable customers from customers in attrition. This is highlighted by the fact that both areas can accurately detect attrition in the two periods following the beginning of the attrition. Indeed, for the 20-th month, the area under both curves is between 0.75 and 0.8 and for the 22^{nd} month, it rises above 0.85. It should also be noted that customers in attrition are starting to totally leave the store in month 24. This means that the attrition can be efficiently detected by solely studying signatures evolution one or two period before the complete loss of a customer.

Another interesting result is that from months 10 to 18, the area under both curves is between 0.6 and 0.7. While this does not allow an accurate identification of customers that will be in attrition, it is interesting because it shows that a single stability measure can have an early detection of attrition. This detection was not visible through monetary or frequency measures because all clients are considered stable during the first 18 months.

These results are similar to the one obtained in [BP05], where the authors obtained an area under the ROC curve of 0.83 to predict whether a customer will be in attrition in the next 6 months. The advantage of signatures analysis is that it provides actionable knowledge to the retailer. Indeed, with the stability measure, one can identify which products mostly contributed

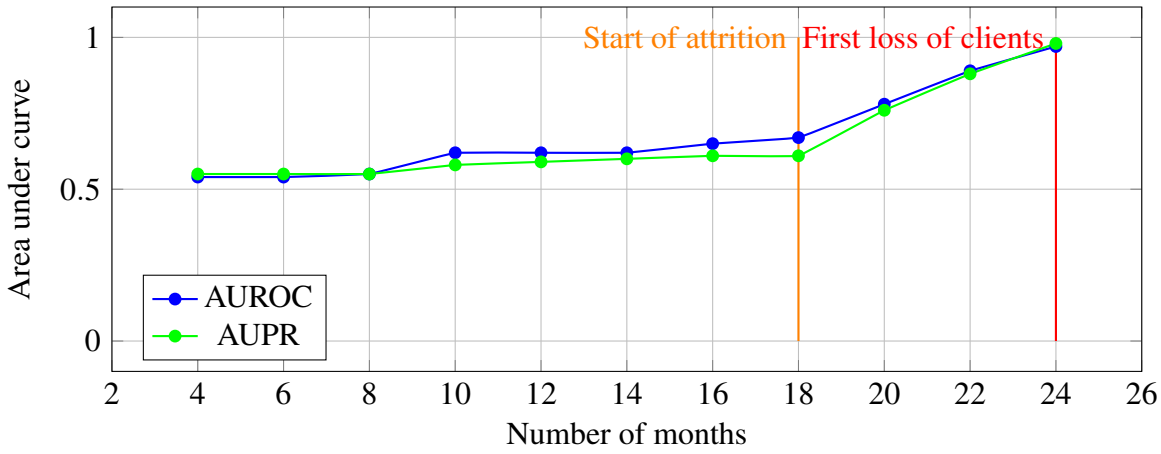

Figure 7.6: Performances of the attrition detection for 2000 clients.

to the loss of stability. The retailer then has more information about which products a customer who is starting to be in attrition is likely buying to a competitor.

7.4.5 Influence of the Window Length

In this part, we study the influence of m on the overall performance of the prediction. Because selected customers are generally starting to totally leave the store 6 months after the first signs of attrition, the window length can not be too large. In the current experiments, three values have been used: 1, 2 and 3 months. The results of each method are shown in Figure 7.7.

First of all, one can see that the attrition detection is almost similar for all window lengths. In the first months, a short window length seems to be interesting to detect of attrition early. This could mean that customers that are likely to be in attrition exhibit an unstable behaviour on short periods of time. This difference could also be due to the fact that the stability criteria must have a significant amount of past signatures to be effective. Indeed, this criteria needs to learn the significance of each product to then detect attrition accurately. When using a window length of one month, the number of past signatures grows quickly and the criteria is effective sooner than for other window lengths. This is interesting to notice that after a year, the three periods yield very similar performances.

At the moment of attrition (that is after the eighteenth month), all window lengths accurately detect attrition. Nevertheless, the longer the window length is, the later this good performance will be achieved. On the other hand, one can see that if the period is too short, the performances of the attrition detection tend to decrease. This is visible in Figure 7.7 where the area under the ROC curves is lower for a window length of one month in the last months.

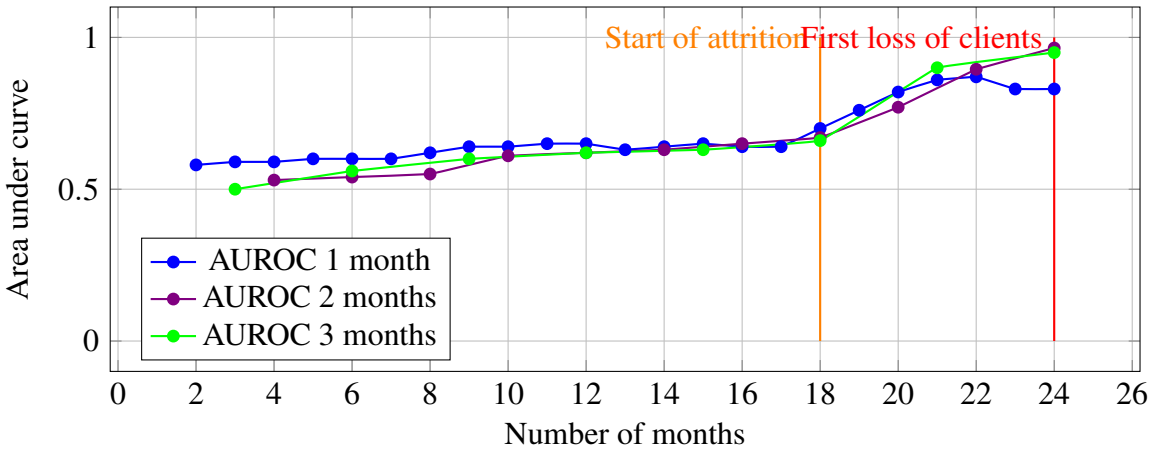

Figure 7.7: Performances of the attrition detection for different window lengths.

Figure 7.7 therefore shows that the window length does impact early and late attrition detection. Short windows seems interesting to detect early attrition, whereas larger windows seems interesting to keep a good performance after the attrition signal.

7.4.6 Individual Signature Analysis

To illustrate the fact that signatures provide actionable knowledge to the retailer, this part presents a use case study of a customer in attrition. The profile of this customer is shown in Figure 7.8, with signatures computed every two months ($m = 2$).

We can see that this customer has a drop of his stability criteria in September 2013 (green line). The main drop of the stability criteria at the beginning of the study should not be considered because we are still learning the significance of each product. We can also notice that the drop in purchase amount (orange line) and number of visits (blue line) starts in the first months of 2014. The stability criteria is therefore predicting the fact that this customer is in attrition 4 months before its impact on the purchase amount and the number of visits.

This detection of attrition is already illustrated in the previous parts. The most interesting point of this use case study is therefore the Figure 7.9. The product loss labeled one corresponds to the first drop in the stability criteria, that is in September 2013. Even if the label 1 is positioned in June 2013, the product loss occurs in September 2013, because we can only detect a loss of product when a signature does not contained this product. We therefore have to compute the signature ending in September 2013 to detect the loss labeled one. This explains that even if the loss is highlighted in June 2013, it corresponds to a drop of stability in September 2013.

This lost product labeled 1 corresponds to a sweet drink (Coca Cola light). We can therefore

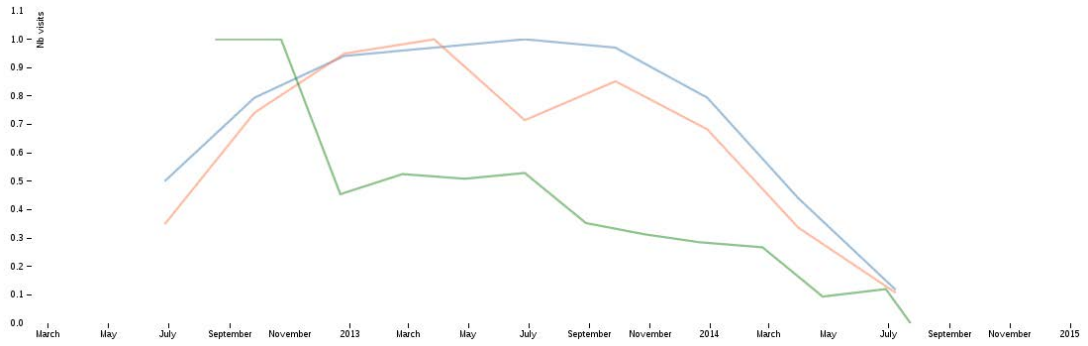

Figure 7.8: Profile of the studied customer. He is stable for 16 months and in attrition afterwards. The blue line represents the number of visits each quarter, relative to the maximum number of visits during a quarter in the 27 months of history. The orange line represents the purchases amount during a 3-months period. The green line is the stability criteria.

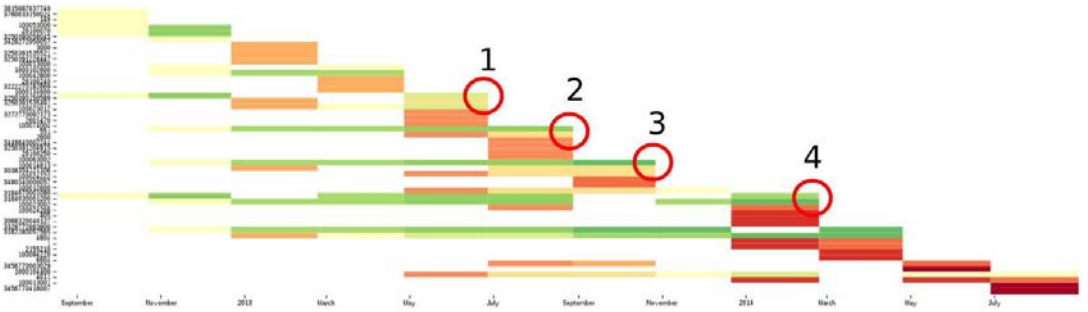

Figure 7.9: Plots the evolution of the signature for the studied client. This client is stable the first 16 months and is in attrition afterwards. Significant products loss are highlighted in red. In this plot, signatures are computed every 2 months.

link the first drop in stability to this product. In the following period, labeled 2, the customer looses another product. We can see on Figure 7.8 that the stability criteria continues to drop. This can be linked to the loss of a coffee bag "arabica". The same analysis can be conducted for the loss of products labeled 3 (a french cheese named "livarot") and 4. We do not have the label for the product 4 because it is not attached to the products taxonomy. Some products (such as fruits and vegetables) are not entirely integrated in the taxonomy, but the taxonomy is currently evolving to include these products.

This use case study therefore demonstrates that signatures provide actionable knowledge to the retailer by linking a drop of stability to a given product. This knowledge is adapted to each customer preferences so that the retailer can then perform a targeted, and therefore efficient, marketing.

Overall, we have illustrated how signatures can be used to tackle a practical issue for retailers. First, by visualizing signatures computed on consecutive periods, we were able to have an intuition about what differentiates a stable customer from a customer in attrition. Then, we developed a simple method based on signature to associate a score to each customer. This score allows us to separate stable customers from customers in attrition. Finally, by looking at individual signatures and stability measures, we are able to link the attrition to a specific set of products.

Summary

- Signatures can successfully be used to tackle attrition detection.
- Simple method built on top of the signature model gives good results.
- The signature model gives good prediction results, as well as actionable explanations.

PART III

Signature Selection

CHAPTER 8

SKY-SIGNATURES

Contents

8.1 Motivations

In the previous part, we have seen that signatures are able to identify recurrent behaviors in a symbolic sequence. The input of the signature algorithm are: the sequence $\alpha = \langle T_1, T_2, \ldots, T_n \rangle$ and the number of occurrences k.

Setting the value of k is not easy in practice. Indeed, it heavily depends on the sequence rhythm, as well as on the type of analysis a practitioner wants to perform. For example, when studying retail customers, one could analyze the most frequently bought items. In that case, the value of k should be set relatively high. One could also decide to look at rare recurrent

purchases, hence using a relatively low value for k. In both cases the term "relatively low or high" has to be determined through several tryouts.

To overcome the manual setting of k, we first introduced the *sky-signature*. The idea of the *sky-signature* is to output a summary of signatures for all possible values of k for a given sequence α. The *sky-signature* gives the full picture of the recurrent behavior of a sequence, as defined by the signature model. The *sky-signature* is based on *skyline patterns* [Sou+11].

8.2 Sky-signatures Definition

The definition of a sky-signature relies on the definition of a *skyline pattern*, presented in Section 3.1. I here briefly recall the definition of a skyline pattern.

Definition 19. *A skyline pattern is the result of a pattern selection method [Sou+11]. Given a set of patterns $\mathscr{P}$ and a set of measures $\mathscr{M}$, a pattern p is a skyline pattern if it is not dominated by any other pattern of $\mathscr{P}$ on $\mathscr{M}$.*

Skyline patterns can output compromises on products sets by selecting a restricted, though unbounded, set of patterns that are optimal on a given set of measures.

Using skyline patterns to define sky-signatures first requires us to define the set of patterns $\mathscr{P}$. In our case, $\mathscr{P}$ corresponds to the set of signatures, for all values of k. We therefore have $\mathscr{P} = \{Sig(\alpha,1), Sig(\alpha,2) \dots Sig(\alpha,n)\}$.

Next, we have to define the set of measures $\mathscr{M}$ that we will use. In our setting, we have $\mathscr{M} = \{k, \|Sig_k(\alpha)\|\}$. The two measures we are using to compute the skyline is therefore the number of segments, denoted k, and the length of the signature items denoted $\|Sig(\alpha,k)\|$.

Definition 20. *Let $Sig(\alpha,k_1)$ and $Sig(\alpha,k_2)$ be two signatures. $Sig(\alpha,k_1)$ dominates $Sig(\alpha,k_2)$ on the set of measures $\{k, \|Sig(\alpha,k)\|\}$ iff $k_1 > k_2$ and $\|Sig(\alpha,k_1)\| \geq \|Sig(\alpha,k_2)\|$, or $k_1 \geq k_2$ and $\|Sig(\alpha,k_1)\| > \|Sig(\alpha,k_2)\|$. $Sig(\alpha,k_1)$ dominates $Sig(\alpha,k_2)$ on $\{k, \|Sig(\alpha,k)\|\}$ is noted $Sig(\alpha,k_1) > Sig(\alpha,k_2)$.*

Definition 21. *The* sky-signature *of a sequence α is defined as $skySignature(\alpha) = \{Sig(\alpha,k) \mid k,k' \in [1 \dots n], \nexists Sig(\alpha,k'), s.t\ Sig(\alpha,k') > Sig(\alpha,k)\}$*

A *sky-signature* is therefore a *skyline pattern* computed on the set of measures $\{k, \|Sig(\alpha,k)\|\}$.

The sky-signature computation on such a set of measures will output a set of signatures that are local optima in terms of length or number of segments. Therefore, we have the signature

Database of C_2	
TID	Items
t1	Bread, Milk, Chocolate, Cheese
t2	Bread, Orange juice, Steak
t3	Bread, Milk, Orange juice
t4	Bread, Chocolate, Apples

Filtered database of C_2	
TID	Items
t1	Bread, Milk, Chocolate
t2	Bread, Orange juice
t3	Bread, Milk, Orange juice
t4	Bread, Chocolate

Table 8.1: Products filtering

with the largest number of segments and the longest signature in the sky-signature, along with all intermediate optimal sets of products between these two extremes.

In Section 8.5, we will show that analyzing the sky-signature helps to understand the recurrences in a symbolic sequence, at different time scales.

8.3 Sky-signature Mining Algorithms

From the definition of the sky-signatures, a naive way to compute it would be to compute signatures for values of k ranging from 1 to n and then post-process this set of signatures to only output the ones that satisfy the constraints defined in the sky-signature definition.

Thanks to the use of dynamic programming in Algorithm 1, a straightforward optimization of the naive method is to compute the signature for a number of segments equals to n. Then, the sky-signature can be derived from the DP segmentation table, as it contains the signatures at every step between 1 and n.

The main drawback of this method is that its complexity is in $O(n^3)$. One could notice that this complexity could be reduced in practice, as we do not need to set $k = n$. Indeed, we know that when k is greater than the frequency of the most frequent item, the signature is empty. Hence, k can be set to the largest item frequency in α, which can still be equal to n in the worst case.

Another way to compute sky-signatures is to use the formulation of signatures as episodes. This allows us to use a pattern growth algorithm to mine sky-signatures. While this type of algorithm has a worse theoretical complexity (in $\mathcal{O}(2^{|\mathcal{I}|})$) than the dynamic programming ap-

proach, it can be faster in some practical settings, especially when the number of segments to consider is high.

8.3.1 Pattern Mining Based Algorithm

The basic idea of the *sky-signatures* mining algorithm based on pattern growth is to iterate through all possible itemsets by extending previous ones, and for each set, verify whether it belongs to the *sky-signature*. A set of items belongs to the *sky-signature* if it is not dominated by a set of items contained in the current *sky-signature* on the set of measures $\{k, \|Sig(\alpha, k)\|\}$.

As the pattern growth approach is especially slow when the number of segments is low, one can specify a minimal number of segments that each signature in the sky-signature should have. The result is then a truncated sky-signature. To have the complete sky-signature, the minimal number of segments should be set to 1.

A pseudo code of the algorithm skeleton is presented in Algorithm 5 and its production is illustrated in Diagram 8.1.

Algorithm 5: Sky-signature algorithm

Input: transactions: the client's transactions, $min_{segments}$: the minimal number of segments
Result: Client's sky-signature

```
1 transactions = filter(transactions, min_segments);
2 I = ∪t, t ∈ transactions;
3 skySignature = {∅};
  // Use efficient enumeration strategy
4 forall itemset ⊆ I do
5 |   updateSkySignature(skySignature,itemset, computeMeasure(itemset));
6 end
7 return skySignature
```

In **line 1** of Algorithm 5, the receipts database is filtered so that all items that do not have a support greater or equal than $min_{segments}$ are discarded. These products (or items) can be discarded because they can not be in the sky-signature, since their support is lower than the minimal number of segments.

In **line 2**, the set of all products contained in the filtered database is computed. In **line 4**, the algorithm iterates through all possible sets of products, that is the power set (without the empty set) of $\mathscr{I}$. To iterate through all these sets, different strategies can be deployed. The one that has been used is a pattern-growth method [HPY00], with products ordered in increasing frequency

Algorithm 6: updateSkySignature.

Input: skySignature: the client's current sky-signature, itemset: current itemset, measures: measures for the itemset
Result: Client's new sky-signature

```
1  for skySignatureItemset in skySignature do
2  |  if skySignatureItemset.measures > measures then
3  |  |  return skySignature
4  |  end
5  end
6  skySignature ∪ = {(itemset, measures)};
7  for skySignatureItemset in skySignature do
8  |  if skylineItemset.measures < measures then
9  |  |  skySignature -= skySignatureItemset;
10 |  end
11 end
12 return skySignature
```

order. The idea is to start from the least frequent product, and progressively expand it with the next least frequent product. When no more product can be added, the exploration backtracks to the previous step and try to expand it with the next least frequent product. This exploration strategy is useful to optimize the algorithm.

Then, in **line 5**, the current itemset is compared to the current *sky-signature* content and the *sky-signature* is updated depending on the measures length and number of segments for the current itemset. This updating function is described in Algorithm 6.

In **line 1** of Algorithm 6, the function iterates through all sets of products contained in the current *sky-signature*. In **line 2**, the function computes whether the current set of products is dominated by one in the *sky-signature*. If this is the case, the current set of products does not belong to the *sky-signature* so we do not update it. In **line 6**, the current set of products is added to the *sky-signature* if it is not dominated by any set of products in the current one. In **lines 7 to 10**, sets of products that are dominated by the current sets of products are removed from the *sky-signature*.

It should be noted that this algorithm can be used to mine signatures, by using the set of measures $\{\|Sig(\alpha,k)\|\}$.

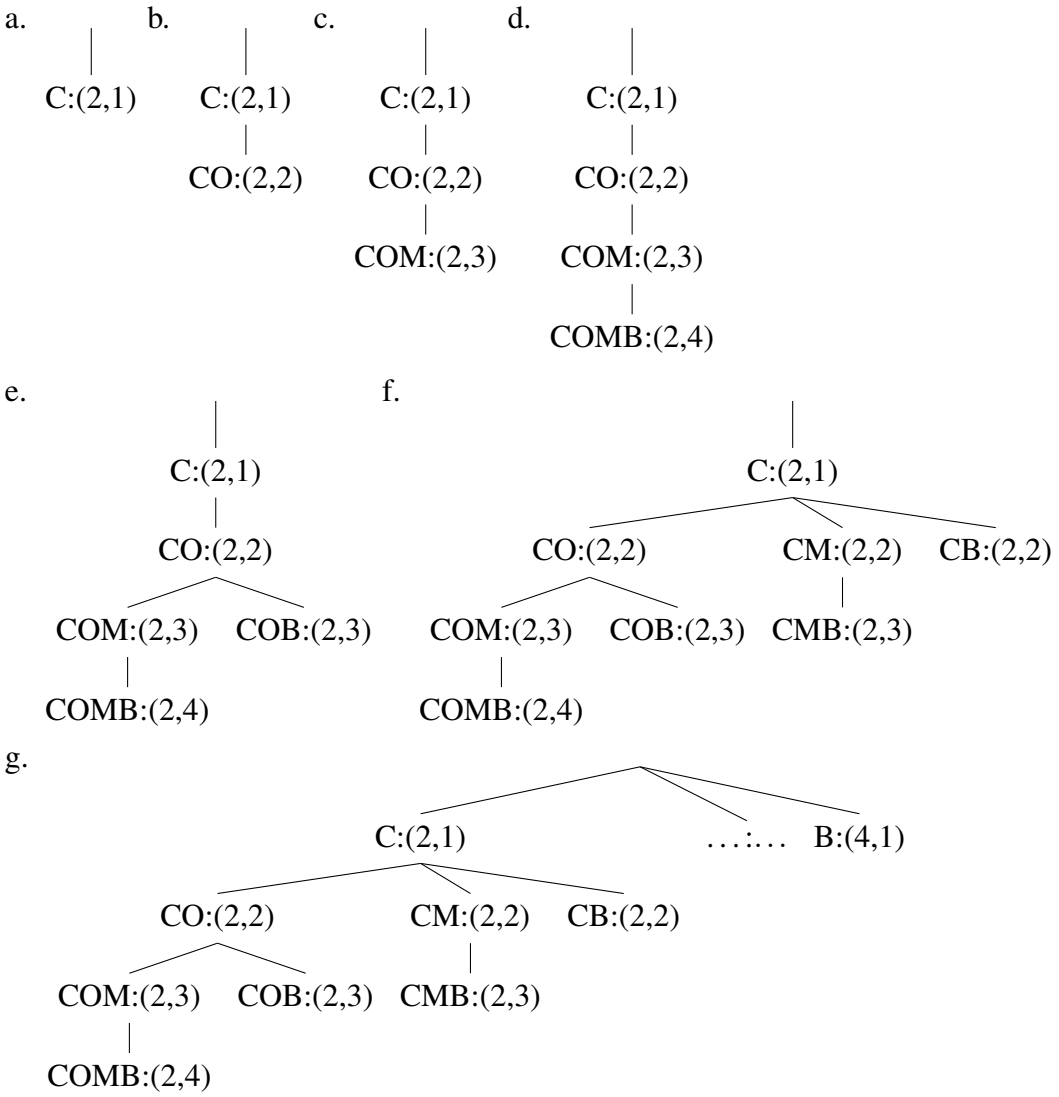

Diagram 8.1: Skyline computation for the filtered database in Table 8.1. Each product is represented by its first letter. Each node has the pair (support,length) attached to it.
a.We start from the least frequent product, chocolate. The skyline is $\{\{Chocolate\}\}$.
b.Adds Orange juice. It has the same support as $\{Chocolate\}$ but is larger so it replaces $\{Chocolate\}$ in the skyline. The skyline is $\{\{Chocolate, Orange\,juice\}\}$.
c.Same as b., but with milk. The skyline is $\{\{Chocolate, Orange\,juice, Milk\}\}$.
d.Same as c., but with bread. The skyline is $\{\{Chocolate, Orange\,juice, Milk, Bread\}\}$.
e. $\{Chocolate, Orange\,juice, Bread\}$ is dominated by $\{Chocolate, Orange\,juice, Milk, Bread\}$ so the skyline remains.
f. Same as e.
g. $\{Bread\}$ is not dominated by $\{Chocolate, Orange\,juice, Milk, Bread\}$ so it is included in the skyline. $\{Bread\}$ does not dominate $\{Chocolate, Orange\,juice, Milk, Bread\}$, so $\{Chocolate, Orange\,juice, Milk, Bread\}$ remains in the skyline. All intermediate nodes (not shown in the figure) are dominated by $\{Chocolate, Orange\,juice, Milk, Bread\}$, so the final skyline is $\{\{Chocolate, Orange\,juice, Milk, Bread\}, \{Bread\}\}$.

8.3.2 Optimization

Iterating through all possible sets of products is in $\mathcal{O}(2^{|\mathcal{I}|})$ where $\mathcal{I}$ contains all products having a support greater than $min_{segments}$. To avoid checking all the possible sets of products, different optimization have been developed.

The first one makes use of the exploration strategy. Indeed, the sets of products are explored through a pattern growth method. One can notice that expanding a set of products can only lead to a lower number of segments for the next set. Hence, if a set of products does not have the minimal number of segments, the expansions of this set will not fulfill the minimal number of segments constraint, so the exploration can backtrack to the previous set. This optimization therefore avoids checking some sets that can not be in the *sky-signature*. This optimization uses the anti-monotony of the number of segments with respect to the set of products inclusion, that derives from the well known anti-monotony of the support of an itemset with respect to the itemset inclusion.

Another optimization has been developed to further reduce the number of sets to explore. This second optimization tries to replace the current set of products by the longest one in the current segmentation. Indeed, each set of products is associated to a given segmentation, the one that maximizes the number of segments that all contain this set. Given this segmentation, one can also compute the longest itemset that is contained in all these segments.

Then, we can replace the current set of products by this longest set. This search of the longest set of products in a given segmentation can be linked to the closure of a set of products. The difference is that this closure uses the number of segments for a set of products instead of its support. This allows the algorithm to skip sets of products that are known to be dominated by another one.

Indeed, expansions of a set of products can only have a lower or equal number of segments. Since the longest set computed has the same number of segments (it has the same segmentation) and is longer than the current set, we know that this longest set dominates the current one. Hence, we can replace the current set by the longest one in the same segmentation and continue the exploration. An example of this optimization is shown in Table 8.2.

8.3.3 Optimized Algorithm

To present the optimized version of the algorithm, I will use a recursive approach. The recursive formulation makes it easier to understand the algorithm. The pseudo code of the algorithm is presented in Algorithm 7.

Database C_1	
Id	Items
1	**Bread**, **Milk**, ***Chocolate***, Cheese
2	**Bread**, Orange juice, Steak
3	Bread, **Milk**
4	Bread, ***Chocolate*** Apples
5	Bread, Fish
6	Milk, Orange juice
7	Bread, Milk

current set = $\{Chocolate\}$
longest set = $\{Chocolate, Milk, Bread\}$

Table 8.2: An example of the principle of the second optimization. Here, $\{Chocolate\}$ can be replaced by $\{Chocolate, Milk, Bread\}$.

Algorithm 7: Sky-signature optimized algorithm.

Input: transactions: the client's transactions, $min_{segments}$: the minimal number of segments
Result: Client's sky-signature

```
1 transactions = filter(transactions, min_segments);
2 I = ∪t, t ∈ transactions;
3 I_sorted = sort_increasing(I);
4 skySignature = {∅};
5 currentSet = ∅;
6 forall product ∈ I_sorted do
7 |   computeSkySignatureRec(transactions, min_segments, skySignature, currentSet,
  |     I_sorted, product);
8 end
9 return skySignature
```

Algorithm 8: computeSkySignatureRec

Input: transactions: the client's transactions, $min_{segments}$: the minimal number of segments, skySignature: the current sky-signature, currentSet: the current set of products, $\mathscr{I}_{sorted}$: the sorted set of all products, product: the product to add to the current set

Result: Client's sky-signature on the product branch.

```
1  currentSet ∪ = {product};
2  𝒮 = getSegments(currentSet);
3  closedSet = common_products(𝒮);
4  addedProducts = closedSet \ currentSet;
5  updateSkySignature(skySignature,closedSet, computeMeasure(closedSet));
6  foreach newProduct ∈ 𝒥_sorted, newProduct > product do
7      if number_of_segments(closedSet) ≥ min_segments then
8          if newProduct ∉ closedSet then
9              if ∀prod ∈ addedProducts, prod > product then
10                 computeSkySignatureRec(transactions, min_segments, skySignature,
                     closedSet, 𝒥_sorted, newProduct);
11             end
12         end
13     end
14 end
```

In **line 2** of Algorithm 7, the set of all products is computed. In **line 3**, this set is sorted in increasing order of frequency. **Lines 6 to 8** iterate through the sorted set of products and **line 7** launches the recursive call to compute the sky-signature. It should be noted that in **line 4**, the variable skySignature is considered to be global.

The function presented in Algorithm 8 is recursive and does the sky-signature computation. In **line 1**, the new product is added to the current set of products. Then, in **line 2**, the segmentation corresponding to the current set of products is stored in $\mathscr{S}$.

In **line 3**, the longest set of products contained in all segments of $\mathscr{S}$ is computed. **Line 4** stores the products that were added to the current set during the closure operation of **line 3**. In **line 5**, the sky-signature is updated with the new set of products. The update function is the one previously defined in Algorithm 6.

From **lines 6 to 14**, we iterate through all products that are greater than the current product. Since the set of all products is sorted in increasing frequency order, this means that we iterate through all products whose frequency is greater than the frequency of the current product. **Line 7** uses the anti-monotony of the number of segments with respect to the set of products inclusion to discard sets of products that will not satisfy the minimal number of segments constraint.

In **line 8**, the algorithm verifies that new product expansions of the current set of products are not already included in the current set. If so, we do not want to iterate through these products because they are already in the current set. In **line 9**, the algorithm verifies that the current set of products has not been visited before. Indeed, if the current set of products contains products that are smaller than the current product, then this means that this set has already been processed in previous iterations. Therefore, we do not want to process this set once again. **Line 9** therefore guaranties that the algorithm will visit a set of products only once.

8.4 Combining Both Methods

In the previous section, I described in details the mining algorithm based on pattern mining and briefly talked about the mining algorithm based on Dynamic Programming. Both methods are interesting to compute sky-signatures, and they can be combined to efficiently get the whole sky-signature.

The dynamic programming approach is fast when the number of segments k is low, as only few columns of the table have to be computed. On the other hand, the pattern mining approach is fast when the number of segments k is high. Indeed, when the number of segments is high, few items satisfy the minimal segmentation size (frequency) constraint, which in turns reduces

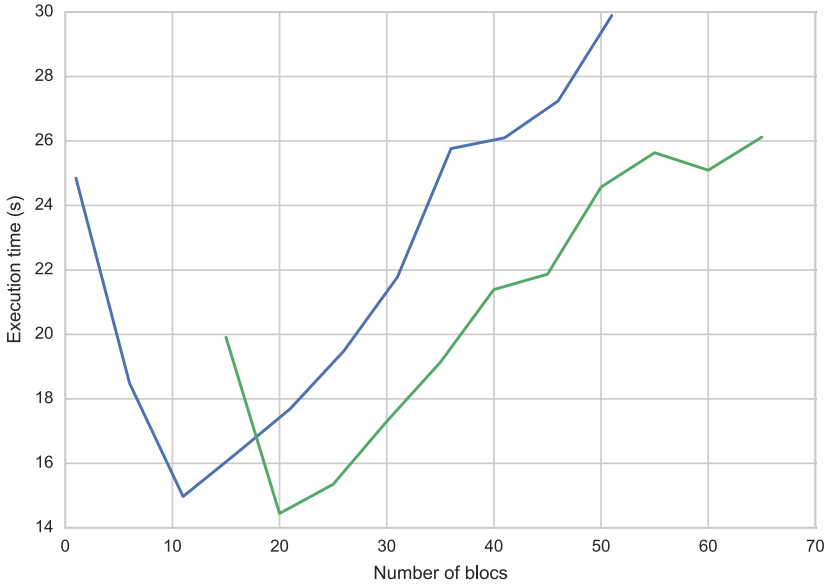

Figure 8.1: The execution times in seconds for the sky-signature computation, for different switch points between both algorithms. The execution times are represented for 2 different customers from the French retail dataset, each represented by a different colour (blue and green).

the size of $|\mathscr{I}|$ and therefore the overall complexity.

A natural idea is to combine both methods to be able to get the whole signature, in a reasonable amount of time. The DP approach will be used to compute the lowest levels of the signature (with the lowest number of segments) and when the DP approach is not worth using anymore, we use the pattern mining approach to compute the remaining parts of the signature, corresponding to a larger number of segments.

The main challenge is now to automatically find the optimal number of segments, where we should switch from the Dynamic Programming approach to the Pattern Mining one. To illustrate the relevance of this combination, we plot the typical execution time of the sky-signature computation, for different values of switch points between both algorithms in Figure 8.1. It is clear that there is a point where switching from one method to another reduces the execution time. It is also clear that this optimal switch depends on the customer at hand, as the optimum point for the blue curve is around 10 segments, while the one for the green curve is about 20 segments.

The optimal moment to switch from one method to another can actually be identified from the analysis of the theoretical complexity of the overall method. Indeed, the DP approach is

in $O(k * n^2)$ while the pattern mining approach is in $O(2^{|\mathscr{I}|})$, with $\mathscr{I}$ depending on k. For each value of k, we can compute the value of $2^{|\mathscr{I}|} + k * n^2$. This corresponds to the theoretical complexity of computing the sky-signature with a number of segments lower than k with the DP approach and then using the pattern mining approach to compute the sky-signature for a number of segments greater than k. The value of the sum can be computed for each value of k using the frequency of each item in the database. This counting step is not expensive and can be reused in the pattern growth algorithm. Hence, computing the theoretical switch point does not add significant computation overhead.

8.4.1 Execution Times

To asses the relevance of the combination of both methods, we compute the whole sky-signature, with a varying change point between both algorithms. A change point at 0 means we use the DP approach to compute the first k levels of the sky-signature, and then switch to the pattern mining approach to compute the sky-signature for number of segments larger than k, where k is the optimal value computed by the method presented above. Then, we shift the change point from the optimal one by some offset and look at the execution time. When the offset is negative, we use the pattern mining approach on lower number of segments than the number recommended by the theoretical evaluation.

The results of this experiment are presented in Figure 8.2. As one can see, when shifting from the theoretical change point, the execution time is going up, especially when the switch occurs too early (negative offset values). However, it should be noted that for offsets close to 0, the execution time is relatively close to the best one. This means that we can reliably estimate the optimal switch points between both algorithms using the method based on the theoretical complexities of both algorithms.

Summary

- Mining sky-signatures with any of the proposed algorithms is costly.
- Combining the algorithm based on sequence segmentation and the one based on episode mining gives an efficient approach to mine sky-signatures.

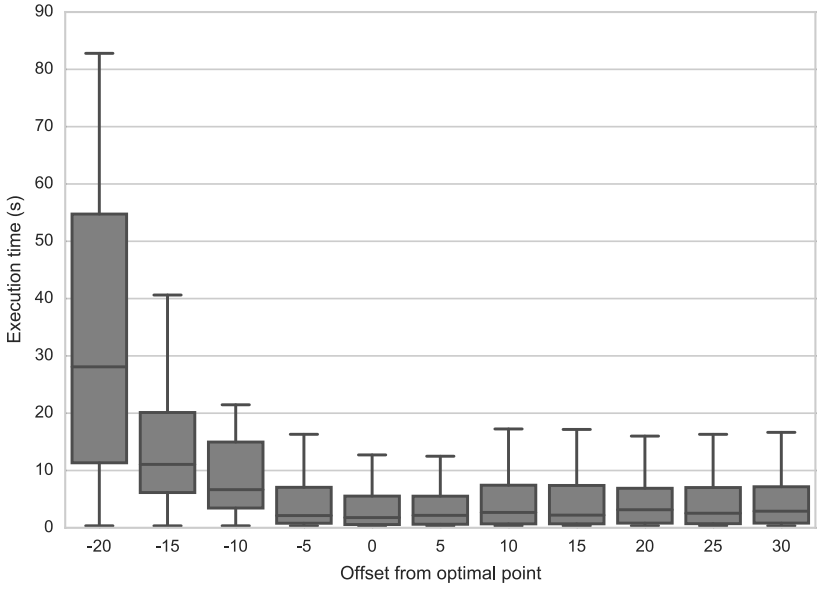

Figure 8.2: The execution times in seconds for the sky-signature computation, for different offsets from the optimal change point. This was computed on 2000 customers from the French retail dataset.

8.5 Experiments

In this part, we will show that sky-signatures are relevant to derive meaningful knowledge and identify coherent sets of events on different time scales, mainly on the TV broadcast and campaign speeches datasets.

8.5.1 Refining the Understanding of Customer Habits

The sky-signature model was introduced to allow the retailer to find recurrent items of a customer on different times scales. To illustrate this possibility, we show a sky-signature of a real customer in Table 8.3. As one can see, the sky-signature indeed contains signatures with different number of segments, giving recurrent purchase behavior of the customer at different time scales.

When using 77 segments, there are two possible signatures: *Ham* or *Yogurt*. The fact that there are two possible signatures usually signals the fact that the customer might have two competing recurrences for a given purchase rhythm.

One can also note that the sky-signature has a Russian doll structure. Indeed, when lowering

the number of segments, we usually add one new item to the signature with a number of segments slightly larger. For example, the signature having 61 segments adds the product *Chocolate* to the signature having 68 segments. The same can be said for the signature with 48 segments, adding *Sugar free jam* to the signature with 61 segments. This redundancy in the sky-signature content can be explained, as in the retail context, purchasing items many times usually implies that these items are frequently bought over time. Hence, items that are in signatures with many segments are likely to occur in signatures with less segments, hence the Russian doll structure.

Because of this redundancy of the sky-signature in the retail context, we decide to instead focus on datasets where the sky-signature shows more diverse behaviors on different time scales.

Table 8.3: Part of a sky-signature of a retail customer

Number of segments	Products
77	Ham
77	Yogurt
68	Ham, Yogurt
61	Ham, Yogurt, Chocolate
48	Ham, Yogurt, Chocolate, Sugar free jam
32	Ham, Yogurt, Chocolate, Sugar free jam, Mineral water
28	Ham, Yogurt, Chocolate, Sugar free jam, Mineral water, Makeup cotton

8.5.2 Regularities at Different Time Scales

To show that the sky-signature is able to detect diverse behaviors on different time scales, we study the TV broadcast dataset. This dataset contains events from the French TV channel *France 2*, during the *Rolland Garros* tennis tournament, presented in Section 6.1.3.

We computed the sky-signature on the whole sequence of TV events. The results are presented in Table 8.4. It should be noted that the sky-signature content can remain almost unchanged for numbers of segments close to one another, as we have seen previously. We therefore decided to only show the most relevant levels of the sky-signature in Table 8.4, that is levels that are significantly different from one another.

The sky-signature levels that were computed on this dataset can be separated into roughly 3 categories. The first category contains TV events that appear many times a day, i.e commercials. This type of event is found when the number of segments is high (equal to 200 in Table 8.4).

The second category of event is defined by events that occur on a daily basis. These events are usually games and series, along with their sponsors, news and commercials. These events

are found by signatures with a low number of segments (15 in Table 8.4). It should be noted that some of these TV shows are not aired during the weekend, hence a number of segments a bit smaller than the number of days in the dataset (15 segments for 23 days).

Finally, the last category of signatures lies between the two first ones. This last category mainly contains temporary events that occur very frequently for a short period of time. In our case the french tennis tournament Roland-Garros started on the twelfth day. The signature computed on a medium number of segments (100 in Table 8.4) contains this event.

The fact that the sky-signature is able to identify Roland-Garros related events in the dataset is actually quite interesting. Because we do not specify any time regularity constraint in the signature definition, the signature is able to identify the most relevant set of events for an approximate time scale. Indeed, if we want to find events occurring in 100 segments during 23 days, this means that we want to find TV shows that are aired about 5 times a day. The events having this precise regularity throughout the whole dataset are Commercials and Weather forecasts. A method based on periodicity (such as periodic patterns) would only find this type of events as the most relevant ones for this time scale.

However, the flexibility of the signature definition allows us to find the set of events that best fit the dataset for this time scale. In our case, the signature finds a larger set of events occurring about 15 times a day (3 times the average periodicity induced by the number of segments) during the afternoon of half of the days in the dataset. Even though these Roland Garros related events are less regular than the daily events (Commercials and Weather forecasts), they provide a more precise description of the dataset.

Overall, this shows that the sky-signature is able to give a complete picture of the recurrent behavior of a sequence, even if these regularities are diverse on different time scales.

Table 8.4: The signature content for different number of segments

Signatures	
Number of segments	Products
200	Commercials, Commercials announcement, Weather Forecast
100	Commercials, Commercials announcement, Sponsor Peugeot, Sponsor GDF, Roland-Garros
15	Today's CD, Traffic news, Millionaire Game, Commercials, Commercials announcement, Weather forecast, News, Motus Games, The Morning show, Sponsor Siemens, Derrick, A day a tree, Sponsor Varilux, A Book, Face Off, The Bold and the Beautiful, The Newlywed Game

8.5.3 Analysis of Temporal Dynamics

In this part, we present how sky-signatures can analyze temporal dynamics. We illustrate this on the 2016 U.S. campaign speeches dataset, that is presented now.

2016 U.S. campaign speeches data

We also applied the signature model to text data. Many text datasets exist, but the signature is mainly tailored to analyze a single sequence of text and extract its recurrent tokens. As such, we focused on using a dataset where recurrent behavior can be analyzed, on a single sequence.

Hence, we processed a dataset containing the transcripts of campaign speeches of both candidates Clinton and Trump, from April, 2015 to November, 2016. The speeches have been extracted from the American Presidency Project (APP).[1] This yielded a total of 164 speeches: 93 for Clinton and 71 for Trump.[2].

Preprocessing The dataset was preprocessed as follows. First, the sentences that did not correspond to a candidate utterance (journalists questions, introduction by another speaker ...) were removed. Next, the sentences were tokenized and the tokens associated with some Part-of-Speech (POS) tags were kept. Precisely, nouns, adjectives and foreign words were kept while verbs and personal nouns were removed. While removing verbs can lead to a loss of semantic information, we found that it resulted in more interpretable topics. This choice of removing verbs has previously been made for topic modeling in political texts [ZS14]. Personal nouns were discarded to remove all references to interviewers or other politicians. We considered keeping some proper nouns (the ones of both campaigners and of some other politicians) but it added noise in the topic modeling step, without providing additional relevant information. Finally, remaining tokens were lemmatized and stop words were removed. We used the WordNet lemmatizer [MF98] and the list of stop words from the nltk library[3] [BKL09]. The final dataset contained 6240 different lemma.

Topic Modeling Even though words could be analyzed directly [Sav10], we decided to analyze topics. This choice is mainly guided by the fact that we are looking for recurrent topics, and working directly on words gave uninteresting results, as recurrent words are not directly representative of each candidate ideas. Different topic modeling techniques were tested [Ste+12]

1. `http://www.presidency.ucsb.edu/2016_election.php`
2. Including the 3 presidential debates Speeches of Clinton prior to April 2015 were discarded
3. `http://www.nltk.org/`

Table 8.5: Some topics found by NMF, and their main lemma.

Topic name	**Main topic lemma**
Economic policy	economy, growth, new, business, income, wage
Woman president and voters	woman, election, president, future, young
Illegal immigration	immigration, illegal, law, border, criminal, visa
Climate change	energy, climate, change, clean, future, important

(LDA [BNJ03] and NMF [LS99]) with different parameters, number of topics and settings (with or without verbs for example). As a result, we concluded that using NMF on count vectors with 15 topics produced the most meaningful, diverse, yet non redundant topics. Some of these topics and their top lemma are shown in Table 8.5. However, it should be noted that other topic modeling techniques ([GC17] for example) could be used, and lead to meaningful results. Indeed, as our method is built on top of topics, any technique that provides good enough topics can be used. Any improvements in the topic model can help to draw more precise conclusions (if cleaner topics are available). This remark is also true regarding our choice of removing verbs and personal nouns.

Within NMF, a speech is represented as a numeric weight vector across all topics. However, the signature model works on symbolic data, which means that a set of representative topics for each speech has to be selected. As we want to discriminate the main topics of a speech from the remaining ones, we applied a clustering on the weight vectors of each speech. Two clusters were looked for, the first containing the highest weights i.e. the cluster of the main topics, and the second containing the secondary or absent topics, with lowest weights. We used the spectral clustering technique [SM00]. We did not used techniques based on the Euclidean distance (such as k-means [Mac+67]) as it is not suited to separate main topics from minor ones. Three main topics emerged per speech on average.

Signature Analysis

By computing signature on this dataset, we aim at finding the main topics discussed by each candidate, at different time scales. The sky-signature of both candidates is presented in Table 8.6.

The main topics of each candidate are well separated, showing that each candidate has its own targeted voters. Clinton focuses on topics related to communities, youth, issues for the

next generations, and woman as president. Trump focuses on topics such as new economical policies, illegal immigration, new social policies and criticism of the former government. We are therefore able to discover each candidate main topics, and their importance. Indeed, a set of topics is more important if it appears in the sky-signature for larger values of number of segments (k).

Table 8.6: Sky-signature topics in speeches of Clinton (top) and Trump (bottom).

Clinton

No	k	**Signature topics**
C1	57	Woman as President
C2	30	C1 + Future challenges for President
C3	16	C2 + Communities and police
C4	12	C3 + Childcare and education

Trump

No	k	**Signature topics**
T1	48	Social policy and critics
T2	28	T1 + New economic policy
T3.1	15	T2 + Illegal immigration
T3.2	15	T2 + Education policy
T4.1	9	T3.2 + Illegal immigration (T3.1 + T3.2)
T4.2	9	T3.2 + Money and wall at border

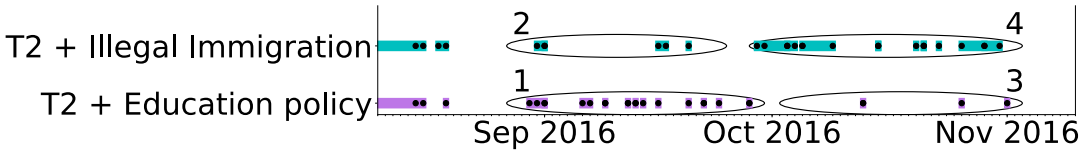

Figure 8.3: Episodes of two Trump signatures. T3.1: Social policy and critics + New economic policy + *Illegal immigration* ; T3.2: Social policy and critics + New economic policy + *Education policy*. Every rectangle in pink or blue represents a segment, and each black dot represents a speech belonging to a segment. Each numbered ellipse represents a group (annotated by hand) of segments.

The signature of Clinton is quite simple, as lowering the minimal number of occurrences only adds new topics to the signature. This is similar to what we observed for retail customers in Table 8.3. This means that Clinton is very stable in her main topics. This observation is also partially true for Trump. Indeed, Trump has sometimes different signatures for a given number of occurrences. For example, with $k = 15$, Trump speeches main topics can include *Illegal*

immigration or *Education policy*, but not both together. This is interesting because it shows that Trump is more diverse in his recurrent topics and that some of them rarely occur together.

To further deepen the analysis of the fact that Trump speeches include either *Education policy* or *Illegal Immigration* but rarely both, let us look at the segments of the related signatures, represented in Figure 8.3.

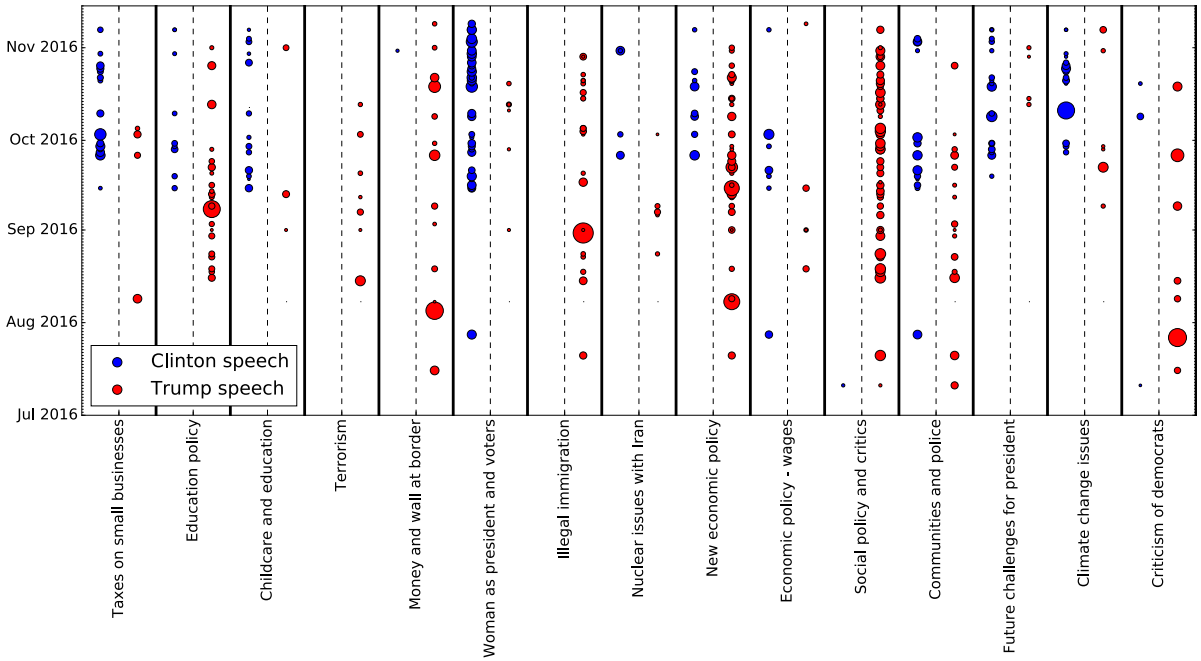

Figure 8.4: Campaign topics through time for each candidate. Each circle represents the presence of a topic in a speech. The larger the topic, the more present in a speech. Trump speeches are depicted in red, Clinton speeches in blue.

First, we note that the difference between both signatures segments began to be apparent by September 2016. Indeed, the signature containing *Illegal immigration* only has three segments (Group 2), whereas the one with *Education policy* has 11 segments (Group 1). This large difference shows that, in September, Trump discussed his main topics a lot (*Criticism of former government*, *New social policies* and *New economic policy*) in association with *Education policies*. In October 2016, he switched to *Illegal immigration* while keeping his main topics, as there are 3 segments for *Education policy* (Group 3) whereas there are 7 segments for *Illegal immigration* (Group 4).

One could argue that this type of information could be derived by solely looking at topics appearances in speeches directly, without using a sky-signature. Hence, we present a visualization of all main topics in Figure 8.4. Only the last months of the campaign are plotted, since

both candidates were particularly active during that period and speeches were sparse earlier. The visualization is especially suited to analyze single topics. First, we can see that most topics are discriminative, they appear often in one candidate's speech while being almost absent in the other's. Some topics, like *Communities and police*, are shared but not used on the same time line. Another example is the use of the *Climate change issues* topic. We can see that it is mainly used at the end of the presidential campaign by Clinton.[4]

While the fact that Trump stopped talking about *Education policy* at the end of September 2016 is visible in Figure 8.4, the segmentation performed by the signature brings additional information. Indeed, the signature is changing only one of its topics, so we know that Trump kept talking about his other main topics (*Social policy and critics* and *New economic policy*) while switching from *Education policy* to *Illegal immigration*.

Another important point is that by the beginning of October, when Trump switched from *Education policy* to *Illegal immigration*, the episodes are longer than the remaining ones (Group 4). This means that Trump's main topics are distributed among more speeches than before, which can reflect a change in his strategy. This information is not easily visible in Figure 8.4, but it is available from a simple analysis of Trump signature.

This case study, based on topic signatures, shows that our method is able to derive each candidate's recurrent topics. Analyzing episodes and related signature topics enables to spot changes in Trump speeches and to explain how some of his recurrent topics are related to each other. This kind of precise analysis is beyond the capabilities of naive regular segmentation techniques.

Summary

- Sky-signatures give the full picture of a sequence recurrences, according to our definition of recurrence.
- On sequences with different event regularities, the sky-signatures correctly identifies each regularity.
- On more regular sequences, the Russian doll structure of the signature gives the relative importance of each recurrent item.
- Anomalies in the sky-signature structure can potentially highlight interesting behaviours.

4. *Climate change issues* became a topic of interest when Clinton attacked Trump on him saying that climate change is a hoax in the first presidential debate (September 26, 2016).

CHAPTER 9

MDL SELECTION OF SIGNATURES

Contents

The previous chapter presented sky-signatures, that are able to capture recurrences at different time scales. However, the sky-signature remains a complex model to analyze, as it contains many signatures. To ease the practical use of signatures, we want to output the signature that best corresponds to the sequence at hand. To this end, we introduce signature selection based on the ideas of compression based pattern mining. This chapter presents this selection technique, as well as experiments showing the practical relevance of the selected signature.

When using signatures to extract the recurrent items in a sequence $\alpha = \langle T_1, T_2, \ldots, T_n \rangle$, one needs to define the number of segments k. Providing meaningful values for k usually requires expert knowledge and/or many tryouts, because the setting of k depends on the dataset at hand and there is no general rule to automatically set k. Our problem is therefore to devise a method that automatically adjusts k, depending on the data at hand. Moreover, as the aim is to output a single signature, this signature should be the one that represents the data best.

Our approach relies on the Minimum Description Length principle (MDL), which enables us to select the best model from a set of candidate models. The idea is that we select the signature that best compresses the data. However, the main challenge consists in defining the length of the data $L(\alpha)$. Because we want to select the signature the best compresses the data, we will use a signature S to define the data length. Hence, $L(\alpha)$ become $L(\alpha, S)$. We say the we encode α using S.

Defining an encoding for α can be seen as defining a transmission protocol between two persons. The first person knows α and wants to send it to the second person. As data is transmitted using bits, the second person needs to know how to decode the bit stream received from the first person. Hence, the two persons have to agree beforehand on how the data will be transmitted. Defining how data is transmitted is similar to defining an encoding.

While the MDL principle for model selection states that the compression ratio is only secondary to the model selection process, defining an encoding that is as short as possible is still of interest. Indeed, to efficiently compress data, one has to identify patterns in the data. Therefore, if the length of the data $L(\alpha)$ is shorter than the length of the data using a signature $L(\alpha, S)$, it means that the signature S is not able to capture patterns in the data. Optimizing the length yielded by the encoding is therefore important to assess how well a signature capture patterns in the data.

9.1 Problem Definition

Problem 1. *Let us denote by* $\mathbb{S}$ *the set of all signatures for all values of k for a sequence* α*, i.e.,* $\mathbb{S} = \{Sig(\alpha, k), k \in [1, |\alpha|]\}$*. The MDL principle states that the best signature* $S \in \mathbb{S}$ *is the one that minimizes encoded length of* α*. That is, the signature we want to find is*

$$S_{MDL} = argmin_{S \in \mathbb{S}} L(\alpha, S)$$

where $L(\alpha, S)$ *is the encoded length that we present in the next section.*

9.2 An Encoding for Signatures

We now explain how $L(\alpha, S)$ is computed, that is how we define our encoding. As typically done in compression-based pattern mining [LV14], we use a two-part MDL encoding to define the encoded length of a database α using a signature S. This leads to decomposing the total

encoded length $L(\alpha, S)$ into two parts: $L(S)$ and $L(\alpha|S)$, with the relation $L(\alpha, S) = L(S) + L(\alpha|S)$. $L(\alpha|S)$ corresponds to the length of the sequence α, knowing the signature S. $L(\alpha|S)$ is described in Section 9.2.2. $L(S)$ is the encoded length of a signature and is presented in the next section. In the remaining of this section, all logarithms are in base 2.

9.2.1 Model Encoding

This section presents the encoded length of the signature model, denoted $L(S)$. The signature is composed of two parts: 1) the signature items $\mathscr{I}$, and 2) the signature segmentation $Seg = \langle S_1, \ldots, S_k \rangle$. The encoding of each part is detailed below.

For the signature encoding, we use a new symbol, denoted *stopcode*, that marks the end of a transaction. The set $\mathscr{I}_{aug} = \mathscr{I} \cup \{stopcode\}$ denotes this new set of items. Let us also recall that the set of signature items is $\mathscr{R}$. We denote by $\mathscr{E}$ the set of items that do not belong to the signature: $\mathscr{E} = \mathscr{I} \setminus \mathscr{R}$

Signature items encoding The encoding of the signature items is in three parts: 1) the number of items, 2) the number of signature items and 3) the signature items. The idea is that we encode the signature items by encoding what items from $\mathscr{I}$ belong to $\mathscr{R}$.

The signature items are a subset of $\mathscr{I}_{aug}$, so we first need to encode the number of items in $\mathscr{I}_{aug}$. A common way to encode non-negative integer numbers is to use the universal code for integers [Grü07; Ris83], denoted $L_{\mathbb{N}}$[1]. In our case, this yields a code of size $L_{\mathbb{N}}(|\mathscr{I}_{aug}|)$. Next, we encode the number of items in the signature, using again the universal code for integers, with length $L_{\mathbb{N}}(|\mathscr{R}|)$.

Finally, we encode the items of the signature. By convention, the *stopcode* symbol is the last symbol of $\mathscr{I}_{aug}$. For the signature items, we know that they are a subset of $\mathscr{I}$. Because we do not care about the order of the signature items, we can therefore use an $|\mathscr{R}|$-combination of $|\mathscr{I}|$ elements without replacement. This yields a length of $\log(\binom{|\mathscr{I}|}{|\mathscr{R}|})$. One interpretation of this code is that the signature items correspond to one $|\mathscr{R}|$-combination of $|\mathscr{I}|$ elements among a total of $\binom{|\mathscr{I}|}{|\mathscr{R}|}$ possible combinations. As we now have both $\mathscr{R}$ and $\mathscr{I}$, we can deduce which items belong to $\mathscr{E}$.

For example, let us consider the sequence depicted in Figure 9.2. We have $\mathscr{I} = \{a, b, c, d, e\}$ and $\mathscr{R} = \{a, b\}$. To understand why we encode the signature items as described above, let us place ourselves in the context where one person knows the signature items and wants to send it to another person. The way the person would send the signature items is equivalent to defining an

1. $L_{\mathbb{N}} = \log^*(n) + \log(2.865064)$, with $\log^*(n) = \log(n) + \log(\log(n)) + \ldots$

encoding. There is a total of 10 2-combinations of 5 elements: $ab, ac, ad, ae, bc, bd, be, cd, ce, de$. The person wishing to send ab will therefore send the index of this combination: 1. However, from the receiver perspective, decoding 1 is meaningless as is. Indeed, the receiver does not know that this index corresponds the combination index of all 2-combinations of 5 elements. For the receiver to know this, we should first send the information that we will encode an index of 2-combinations of 5 elements. We are sending $|\mathscr{I}|$ (5) and $|\mathscr{R}|$ (2) to give this information to the receiver (we are actually sending $|\mathscr{I}_{aug}|$ because we have the additional *stopcode* symbol). With these three information ($|\mathscr{I}|$, $|\mathscr{R}|$ and the combination index), the receiver knows that $\mathscr{R} = \{a, b\}$, hence he knows the signature items. One could notice that both the receiver and the sender have to agree on a common enumeration order for all 2-combinations of 5 elements. This agreement is always assumed.

Segmentation encoding We now present the encoding of the second part of the signature: the signature segmentation. To encode the segmentation, we encode the segment boundaries. These boundaries are indexed on the size of the sequence, hence we first need to encode the number of transactions n. This can be done using again the universal code for integers, which is of size $L_{\mathbb{N}}(n)$. Then, we need to encode the number of segments $|Seg|$, which is of length $L_{\mathbb{N}}(|Seg|)$. To encode the segments, we only have to encode the boundaries between two consecutive segments. As there are $|S|-1$ such boundaries, a naïve encoded length would be $(|Seg|-1)*\log(n)$. An improved encoding takes into account the previous segments. For example, when encoding the second boundary, we know that its value will not be higher than $n-|S_1|$. Hence, we can encode it in $\log(n-|S_1|)$ instead of $\log(n)$ bits. This principle can be applied to encode all boundaries. Another way to further reduce the encoded length is to use the fact that we know that each signature segment contains at least 1 transaction. We can therefore subtract the number of remaining segments to encode the boundary of the segment we are encoding. This yields an encoded length of $\sum_{i=1}^{|Seg|-1} \log(n-(|Seg|-i)-\sum_{j=1}^{i-1}|S_j|)$.

Putting everything together The total cost of the signature encoding is:
$L(S) = L_{\mathbb{N}}(|\mathscr{I}_{aug}|) + L_{\mathbb{N}}(|\mathscr{R}|) + \log(\binom{|\mathscr{I}|}{|\mathscr{R}|}) + L_{\mathbb{N}}(n) + L_{\mathbb{N}}(|Seg|) + \sum_{i=1}^{|Seg|-1} \log(n-(|Seg|-i) - \sum_{j=1}^{i-1}|S_j|)$.

9.2.2 Data Encoding

We now present the encoding length of the sequence given the model: $L(\alpha|S)$. Encoding the sequence α means encoding its items occurrences. For example, if we consider only item a in

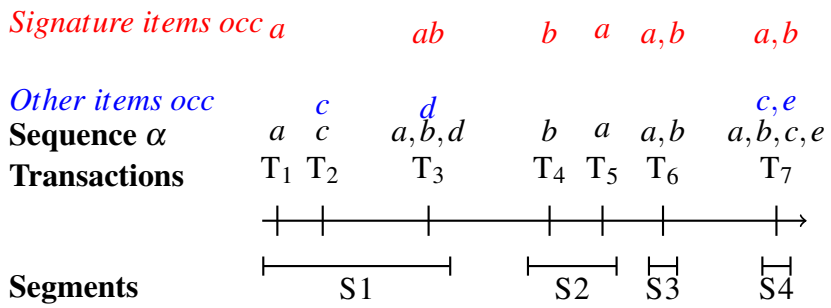

Figure 9.1: A sequence of transactions and its associated encoding scheme. We have $\mathscr{R} = \{a,b\}$, $\mathscr{E} = \{c,d,e\}$ and $\mathscr{I} = \{a,b,c,d,e\}$. The occurrences of each signature item is encoded in the red stream and the items from $\mathscr{E}$ in the blue stream.

Figure 9.1, it means encoding the fact that a occurs in transactions 1, 3, 5, 6 and 7. Because the signature is given, we want to encode α using the information captured by the signature. As in the previous subsection, we exploit the fact that the signature is in two parts.

The first part is the set of signature items. By having a set of items $\mathscr{R}$ for the signature items and a set of items $\mathscr{E}$ for the remaining items, the signature naturally splits $\mathscr{I}$ into two parts. To encode α knowing the signature, we use this natural split by separately encoding the occurrences of $\mathscr{R}$ and the occurrences of $\mathscr{E}$. In practice, this means that we have two streams encoding the items occurrences: one stream for the items in $\mathscr{R}$, and one stream for the items in $\mathscr{E}$. This is shown in Figure 9.1, where the occurrences of the items a and b are encoded using the red stream, and the occurrences of the items c, d and e are encoded using the blue stream.

Then, the signature segmentation tells us that in each segment, there is at least one occurrence of each signature item. We use this information by encoding the first occurrence of each signature item in each segment separately from the rest of the signature items occurrences. In practice, this means that the occurrences of items in $\mathscr{R}$ are split again in two separate streams: one stream for the first occurrence in each segment, and one stream for the remaining occurrences.

To summarize, we have three separate encoding streams: 1) one that encodes the first occurrence of each signature item in each segment, 2) one that encodes the rest of the signature items occurrences, and 3) one that encodes the remaining items occurrences. An example illustrating the three different encoding streams is presented in Figure 9.2.

In the next paragraphs, we present in more detail how each stream is encoded.

Encoding the first occurrence of each signature item in each segment The encoding of this stream is based on the fact that the signature model tells us that in each segment, there is

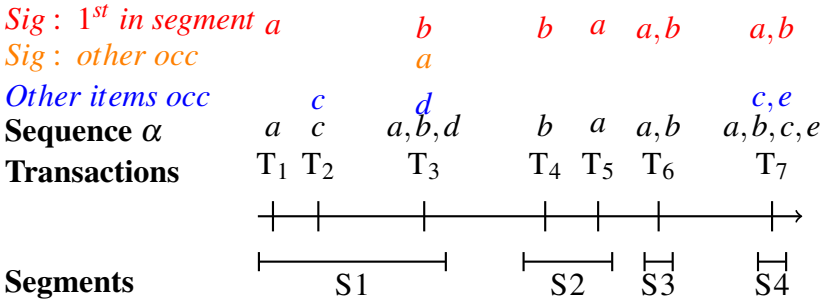

Figure 9.2: A sequence of transactions and its associated encoding scheme. We have $\mathscr{R} = \{a,b\}$, $\mathscr{E} = \{c,d,e\}$ and $\mathscr{I} = \{a,b,c,d,e\}$. The first occurrence of each signature item in each segment is encoded in the red stream, the remaining signature items occurrences in the orange stream, and the items from $\mathscr{E}$ in the blue stream.

at least one occurrence of each signature item. Moreover, we know the size of each individual segment (from the encoding of the model introduced in Section 9.2.1). We therefore encode the first position of each signature item occurrence in segment S_i by encoding the index of that occurrence within segment S_i. Encoding an index over $|S_i|$ possibilities costs $\log(|S_i|)$ bits. For each segment, we do this index encoding $\mathscr{R}$ times: once for each item. We can therefore encode each segment in $|\mathscr{R}| * \log(|S_i|)$ bits.

Encoding the remaining signature items' occurrences Because the signature model does not guarantee that we have a fixed number of occurrences of each signature item in a segment, we cannot use the previous encoding technique relying on segment sizes. In fact, as we have no prior information on these additional signature items occurrences, we use the commonly used uniform code [Grü07], giving an equal penalty to each additional occurrence.

It is important to note that the MDL principle requires lossless compression, i.e., perfect reconstruction of the original sequence should be possible. If we would only send item codes one after the other, we would not know to which transaction each item belongs. To solve this problem, we use the previously introduced *stopcode* symbol to marks the end of each transaction. By using this additional symbol, we can perfectly reconstruct the positions of the signature items' additional occurrences. We denote the new set of 'remaining' items by $\mathscr{R}_{aug} = \mathscr{R} \bigcup \{stopcode\}$. The cost of encoding each occurrence then becomes $\log(|\mathscr{R}_{aug}|)$ bits, as we have one more symbol in our set.

To compute the total encoded length, we simply multiple the total number of symbols in the stream by $\log(|\mathscr{R}_{aug}|)$. The number of stop codes we encode is n. Let us denote by r the count of all occurrences of an item from $\mathscr{R}$ in α. We have $r = \sum_{a \in \mathscr{R}} \sum_{T_i \in \alpha} \sum_{p \in T_i} \mathbf{1}_{a=p}$. The number of additional signature items is equal to $r - |\mathscr{R}| * |Seg|$, as $|\mathscr{R}| * |Seg|$ occurrences are encoded

as the first occurrence in a segment. The total length for encoding additional signature items is therefore $(n+r-|\mathscr{R}|*|Seg|)*\log(|\mathscr{R}_{aug}|)$. One could argue that using this encoding is costly, especially for the encoding of stop codes. However, as our primary goal is to compare models on the same database, the total cost of encoding the stop codes using the uniform code is almost an additive constant for all possible models; it only varies slightly as $|\mathscr{R}|$ varies. Hence, in practice it hardly affects our results.

Remaining items occurrences encoding Finally, we present how to encode the remaining items occurrences, that is, the occurrences of items in $\mathscr{E}$.

The encoding technique is identical to the one used to encode additional signature items occurrences. Indeed, we do not have any prior information on the remaining items occurrences, hence we again use uniform codes over the set $\mathscr{E}$. As explained in the previous section, we need to reconstruct the original data perfectly, hence we need a *stopcode* symbol to mark the end of a transaction. We denote the new set of items by $\mathscr{E}_{aug} = \mathscr{E} \bigcup \{stopcode\}$. Let us denote by e the count of all occurrences of an item from $\mathscr{E}$ in α. We have $e = \sum_{a\in\mathscr{E}} \sum_{T_i\in\alpha} \sum_{p\in T_i} \mathbf{1}_{a=p}$.

Encoding one occurrence then costs $\log(|\mathscr{E}_{aug}|)$ bits. As we have e remaining items occurrences, and n stop code symbols to encode, the cost of encoding this stream is $(e+n)*\log(|\mathscr{E}_{aug}|)$ bits.

Putting everything together Summarizing the cost of encoding the data knowing the model we get:
$L(\alpha|S) = \sum_{S_i\in Seg} |\mathscr{R}|*\log(|S_i|) + (n+r-|\mathscr{R}|*|Seg|)*\log(|\mathscr{R}_{aug}|) + (e+n)*\log(|\mathscr{E}_{aug}|)$

9.2.3 Other Encoding Choices

It should be noted that instead of using uniform codes, we could have used prequential codes [Grü07] to encode the remaining signature items occurrences, or the remaining items occurrences. The upside of prequential codes is that they do not introduce arbitrary choices in the encoding, whereas uniform codes make the arbitrary choice that item occurrences are uniformly distributed. However, prequential codes are less simple than uniform codes. The remaining of this paragraph explains how we can encode remaining items occurrences using prequential codes.

To use prequential codes to encode items occurrences, we need a valid probability distribution over items each time we encode an occurrence. To have a valid probability distribution, we will compute the empirical distribution of items occurrences (or usages).

At the beginning, all usages are initialized to a constant value h. This is equivalent to saying that without any information about item occurrences, we consider the occurrences to be uniformly distributed over items. Typically, we set $h = 0.5$. Then, each time an item is encoded, the items usages are updated to get a new valid probability distribution over items.

For example, let us consider the blue sequence in Figure 9.2. The stream we wish to encode is $c,d,,,,ce$, with "," the *stopcode* symbol, and our alphabet contains 4 symbols. The cost of encoding the first occurrence of c is the logarithm of the inverse probability of c: $-\log(\frac{h}{4*h}) = -\log(\frac{1}{4})$.

Then, the usage of item c is updated and the next item is encoded: ",". Its length is $-log(\frac{h}{4*h+1}))$, and the count for "," is also updated. Then, the first occurrence of d is encoded in $-log(\frac{h}{4*h+2}))$ and its usage is also updated. The next step is to encode "," again. Its cost is $-log(\frac{h+1}{4*h+3}))$ as we recorded that this symbol was used once before. Following this principle, we can encode the whole stream for a cost of $log(\frac{4h}{h}) + log(\frac{4h+1}{h}) + log(\frac{4h+2}{h}) + log(\frac{4h+3}{h+1}) + log(\frac{4h+4}{h+2}) + log(\frac{4h+5}{h+3}) + log(\frac{4h+6}{h+4}) + log(\frac{4h+7}{h+1}) + log(\frac{4h+8}{h}) + log(\frac{4h+9}{h+5})$.

When transforming the sum of logarithms by the logarithm of the product we get a length of $log(\frac{\prod_{i=0}^{9} 4h+i}{(\prod_{j=0}^{5} h+j)*h*(h+1)*h*h})$. As shown in [BV15], the order in which we send items does not matter as we can move multiplication terms as we want. Generalizing this example, and using the work presented in [BV15], we get that the encoded length of the remaining items occurrences is $log(\frac{\prod_{i=0}^{e-1+n}(i+|\mathscr{E}_{aug}|h)}{\prod_{x\in\mathscr{E}_{aug}}\prod_{j=0}^{usage(x)-1}(j+h)})$.

The same principle can be applied to encode the stream of remaining signature items' occurrences (the orange stream in Figure 9.2).

However, experiments using prequential codes did not show any improvement over the results of the experiments done in Section 9.4. Hence, we decided to use the simpler uniform codes, as they suffice for our purposes and simplify the computation of encoded length. However, we present this alternative encoding, as it might yield more interesting results than the uniform encoding in other applications.

9.3 Algorithms

The previous section presented how a sequence is encoded using the MDL principle, completing our problem formalization. Now, the remaining problem is to find the signature that minimizes the total compressed size, that is, finding S_{MDL} such that $S_{MDL} = \text{argmin}_{S\in\mathbb{S}} L(\alpha, S)$.

One naïve approach would be to directly mine the whole set of signatures $\mathbb{S}$ and then find the signature that minimizes the code length. However, mining a signature with k segments has

time complexity $O(n^2k)$. Mining the whole set of signatures requires k to vary from 1 to n, resulting in a total complexity of $O(n^4)$. The quadratic complexity does not allow us to mine the complete set of possible signatures, hence we have to rely on heuristic approaches such as the following greedy hillclimber.

9.3.1 Greedy Algorithm

To search for the signature in $\mathbb{S}$ that minimizes the code length, we initially rely on a greedy algorithm that allows for quickly mining good signatures. More precisely, we use a top-down hill climbing algorithm that recursively adds new segments by greedily minimizing $L(\alpha, S)$. We start with one segment containing the whole database, and then search for a single segment boundary that minimizes the encoded length. Then, we recursively search for a new single segment boundary that minimizes the encoded length. We stop when no segment can be added, that is, when the number of segments is equal to the number of transactions. During this whole iterative process, we record the signature leading to the best encoded length.

Algorithm 9: Widening algorithm for signature code length minimization.

Input: $\alpha = \langle T_1, \ldots, T_n \rangle$: sequence of length n, β: the diversity parameter, w: the beam width

Result: *Sig*: the best signature found

```
BestSignatures = ∅;
for k = 1, n do
    BestKSignatures = ∅;
    AllKSignatures = Split1Segment(BestKSignatures);
    S_opt = argmin_{S∈AllKSignatures} L(α, S);
    BestSignatures = BestSignatures ⋃ S_opt;
    BestKSignatures = BestKSignatures ⋃ S_opt;
    θ = t(β, w, AllKSignatures);
    while S_opt ≠ ∅ and |BestKSignatures| < w do
        S_opt = argmin_{S∈AllKSignatures} L(α, S), ∄S_i ∈ BestKSignatures, d(S_i, S) ≤ θ;
        BestKSignatures = BestKSignatures ⋃ S_opt;
    end
end
return argmin_{S∈BestSignatures} L(α, S);
```

9.3.2 Diversity Widening

One of the drawbacks of the greedy algorithm, however, is that it can perform early segment splits that seem promising initially, but that are actually detrimental to the search for the overall best signature.

To mitigate this drawback, one solution is to keep the top-w signatures (with respect to encoded length) at each step of the greedy algorithm instead of keeping only the best one. This technique is called beam search and is commonly used in pattern mining [LK12]. The number of best solutions to keep at each step of the algorithm is called the beam width, which we denote by w. Nevertheless, the beam search technique still suffers from the fact that among the best w signatures, many of them tend to be similar and correspond to slight modifications of one signature. In our case, this means that most signatures in the beam would have segmentations that are very similar.

To prevent from having similar solutions in the beam, the widening technique [SRB94] consists in adding a diversity constraint into the beam. This can be done in different ways [IB13; LK12; SRB94], but a common solution is to add a distance constraint between each pair of elements in the beam. Basically, all pairwise distances between the signatures in the beam have to be greater than a given threshold θ. One requirement of this approach is that one needs to define a distance between each pair of elements, as well as a distance threshold θ. Both will be presented in the following Section.

Widening algorithm

Pseudo code of the proposed widening algorithm searching for the signature with the lowest code length is presented in Algorithm 9. **Line 2** iterates over all the possible values for the number of segments. **Line 4** computes all signatures having k segments that are considered to enter the beam. More specifically, function *Split1Segment* takes as input a set of signatures and returns as output the direct refinements of each of these signatures. A direct refinement of a signature corresponds to splitting one segment in the segmentation associated with that signature. If a signature has k segments, a refinement of this signature has $k+1$ segments and both have $k-1$ segments in common. **Line** 5 selects the best refinement: the one having the smallest code length.

Lines 8 to 12 perform the widening step by adding new signatures to the beam while respecting the pairwise distance constraint. **Line 8** computes the distance threshold (θ) depending on the diversity parameter (β), the beam width (w), and the current refinements. The details of

the threshold computation are presented in Algorithm 10. Once we know this threshold, we recursively add a new element in the beam, until either the beam is full or no new element can be added (**Line 9**). **Line 10** defines which element is added to the beam: we add the signature having the smallest code length and being at a distance of at least θ to any current element of the beam. **Line 14** returns the best overall signature we have encountered.

It should be noted that the greedy algorithm presented in Subsection 9.3.1 is a specific instance of Algorithm 9, with $\beta = 0$ and $w = 1$.

Distance between signatures

In the previous section, we presented the widening algorithm for mining the optimal signature. In this algorithm, a distance measure for signatures has to be defined (used in line 10 in Algorithm 9). As the signature is composed of two parts, the signature items and the signature segmentation, we can in principle use either part to define the distance between two signatures. However, in Algorithm 9 we use the segmentation to explore our search space and each segmentation uniquely corresponds to a signature. As the purpose of the signature distance is to bring diversity in the beam, we will use the segmentation to define the distance between two elements of the beam, i.e., between two signatures.

Terzi et al. [Ter06] presented several distance measures for segmentations. Among the different distances presented, the *disagreement distance* is particularly appealing as it compares how transactions belonging to the same segment in one segmentation are allocated to the other segmentation. Let $S_a = \langle S_{a1} \dots S_{ak} \rangle$ and $S_b = \langle S_{b1} \dots S_{bk} \rangle$ be two k-segmentations of a sequence α. Let $seg(S, T_i) = S_j, s.t.\ S_j \in S, T_i \in S_j, T_i \in \alpha$ be a labeling function that returns the segment of S where T_i appears.

We denote by $d(S_a, S_b)$ the disagreement distance between segmentation a and segmentation b. The disagreement distance corresponds to the number of transaction pairs that belong to the same segment in one segmentation, but that are not in the same segment in the other segmentation. Formally, $d(S_a, S_b) = \sum_{T_i, T_j \in \alpha, i \neq j} \mathbb{1}((seg(S_a, T_i) == seg(S_a, T_j)) \neq (seg(S_b, T_i) == seg(S_b, T_j)))$. Techniques on how to efficiently compute this distance are presented in [Ter06].

Defining a distance threshold

In Algorithm 9, the distance between two signatures has to be compared to a distance threshold θ. This distance threshold controls the diversity constraint within the beam. If θ is equal to 0, then there is no diversity constraint, as any pairwise distance between two different signatures

Algorithm 10: Distance threshold computation.

Input: β: the diversity parameter, w: the beam width, $AllSignatures$: the set of signatures that can enter the beam

Result: θ: the computed distance threshold

1 KBest = $\beta * |AllSignatures|$;
2 BestSignatures = GetBestSignatures(AllSignatures, KBest);
3 BestSignature = GetBestSignatures(AllSignatures, 1);
4 **return** $\min_\theta |\{S \in BestSignatures | d(S, BestSignature) < \theta\}| \geq |BestSignatures|/k$;

will be greater than 0. Higher values of θ will enforce more diversity in the beam, as some good signatures will not be included in the beam as they are too close to signatures that already are in the beam.

However, directly setting the θ threshold is not easy. For example, the beam width w has to be taken into account when setting θ. Indeed, when we have a large beam width, the distance threshold should be low enough to allow many good signatures to enter the beam. But if the distance threshold is too high, many signatures having a high code length will be added to the beam.

To this end, we introduce a method that automatically sets the θ parameter, depending on the beam width and on a new parameter β that is easier to interpret. The β parameter ranges from 0 to 1 and controls the strength of the diversity constraint. The intuition behind β is that its value will approximately correspond to the relative rank of the worse signature in the beam. For example, if β is set to 0.2, it means that the signatures in the beam will be in the top-20% in ascending order of code length. We detail in Algorithm 10 how θ is derived from β and w; this algorithm is called by the t function in line 8 of Algorithm 9.

Knowing the set of all candidate signatures that are considered to enter the beam, we can retain only the β best signatures (**Line 2** of Algorithm 10). Then, in **Line 3** we extract the best signature. Finally, we look for the distance threshold θ such that the number of signatures within a distance of θ from the best signature is equal to the number of considered signatures divided by the beam width w (**Line 4**). The rationale behind this threshold is that since we are adding w signatures to the beam and we want to use the β best signatures, the distance threshold should approximately discard $1/w$ of the β best signatures around each signature of the beam.

Sky-signature based algorithm

A different approach consists in using the *sky-signature* model presented in Section 8.2. This approach is a less naïve alternative to the naïve direct mining of $\mathbb{S}$. Indeed, the sky-signature

can be used as an efficient way to quickly mine signatures that well represents the set of all signatures. Because we do not look at all signatures, but only at signatures in the sky-signature, this approach is also heuristic. However, it enables the use of the efficient sky-signature mining algorithms presented in Section 8.3. More specifically, we will use the *hybrid* algorithm defined in Section 8.4, as it is efficient to mine the whole sky-signature.

9.4 Experiments

This section first studies the different algorithms mining the best signature according to our encoding. Then, we show the relevance of the problem formalization and corresponding signature encoding defined in Section 9.2 on synthetic datasets. Finally, we analyze signatures computed on real customers to illustrate the practical use of signatures minimizing the encoding length.

9.4.1 Runtime Experiments

This section presents best MDL code length, as well as the runtimes of the algorithms presented in Section 9.3. These experiments are performed on 1000 Instacart customers, uniformly sampled from the 6076 Instacart customers having more than 60 baskets. Several settings of Algorithm 9 are tested, with β ranging from 0 to 0.6 and k ranging from 1 to 20. The not so naïve approach based on sky-signature is also evaluated in 2 settings: one where the Dynamic Programming algorithm is the optimal one, and one where the Dynamic Programming algorithm is the approximate one. The MDL code length results are presented in Figure 9.3, and the runtimes in Table 9.1.

The best mean MDL encoded length is obtained when using the optimal sky-signature miner. This is reasonable, as the sky-signature acts as a way to quickly retain the most interesting signatures. One interesting thing to note is that using the non optimal sky-signature miner leads to a degradation of the MDL encoded length. For the different beam search instances, the results meet our expectations. Indeed, the larger k, the better the resulting MDL encoded length is. Adding diversity, by increasing β, leads to better encoded length. Adding diversity can be seen as a way to simulate a larger beam size, without degrading the MDL encoded length too much. For example, with a beam size of 5 and a diversity of 0.2, we can obtain an MDL encoded length that is similar to the one obtained with a beam size of 10 without diversity. As the runtime mainly depends on the beam size, the diversity allows us to quickly find good results.

Runtimes are also meeting our expectations. Indeed, the beam search runtimes increase

proportionally with k, and adding diversity has a minor impact on the runtime. The sky-signature mining algorithms are about 5 times faster than the beam search instance finding similar MDL encoded lengths. One thing to note is that the largest execution times were observed for the optimal sky-signature algorithm. This means that while this algorithm is generally faster than its beam search alternative, it can be significantly slower in a few cases.

A more practical thing to consider is that the sky-signature mining algorithm was developed at the beginning of the thesis, while the beam search was developed in the last months of the thesis. This means that the sky-signature mining algorithm has been continuously optimized, while there might be some room for improvement for the beam search implementation.

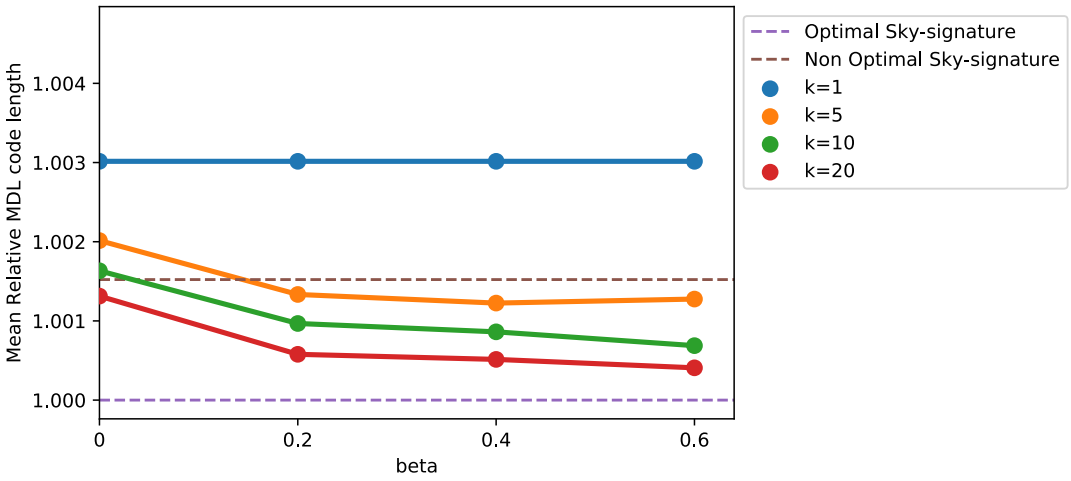

Figure 9.3: Relative mean MDL code length for different algorithms. The MDL code length found by the algorithm based on the optimal Sky-signature mining algorithm (mean MDL code length of 4488.03 bits).

9.4.2 Signature Encoding Relevance

First, to assess the relevance of the proposed encoding, the best encoded length found by the different algorithms on 1000 Instacart customers is compared with the encoded length of the sequence, without using the signature model. If the encoded length using the signature is smaller than the encoded length without the signature, it shows that the encoding is appropriate as it accurately captures regularity leading to data compression. This encoded length comparison is presented in Figure 9.4.

We can see that the encoded length using the signature is always shorter than the encoded length of the raw sequence, without using signatures. This is a desirable property, as it shows

Table 9.1: Mean runtimes (in seconds) for 1000 customers from the Instacart dataset, depending on different algorithms instances.

Algorithm	k	β	Optimal miner	Runtime (s)
Beam search	1	0		4.23
		0.2		4.23
		0.4		4.24
		0.6		4.23
	5	0		19.45
		0.2		19.63
		0.4		19.78
		0.6		19.94
	10	0		38.16
		0.2		38.76
		0.4		39.22
		0.6		39.53
	20	0		74.28
		0.2		76.38
		0.4		77.51
		0.6		78.73
Sky-signature			False	1.75
			True	16.63

that the signature model captures a regularity that allows for compression. The typical gain is about 25% of the encoded length without signatures.

To assess the relevance of our proposed MDL encoding, we now show that it is able to retrieve a planted signature from synthetic datasets.

We generate our synthetic dataset as follows. First, we pick a number of segments uniformly at random between 5 and 20, and for each segment a length uniformly at random between 2 and 20 transactions. Then, we choose a set of 30 items that will be in each segment. This set of items corresponds to the signature items. We augment each segment with additional data to assess the ability of both the encoding and the algorithm to identify the signature we originally planted.

To augment the synthetic dataset, we add new "noise" items to each segment. These items are randomly selected from a set of items different from the signature items set. The noise items set size is 600. The extend of the augmentation is expressed as a percentage of the overall number of signature items. For example, an augmentation rate of 0.1 means that we add a number of extra items equal to 10 percent of the total number of signature items in each segment.

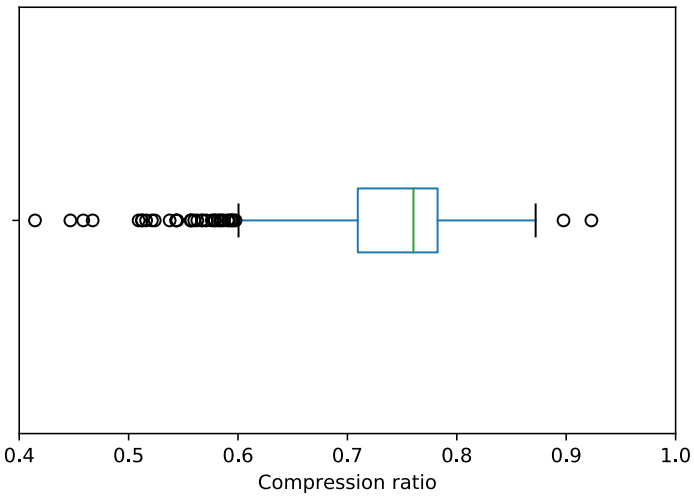

Figure 9.4: Boxplot of the compression ratio between the MDL code length of the best signature and the MDL code length of the sequence without using a signature. Values below 1 mean that the MDL code length using the signature is shorter. This was computed on 1000 Instacart customers.

The larger this rate, the harder it is to recover the original signature.

We generated 100 customers for each noise items rate. For each generated database, we compute the best signature according to our MDL encoding with the optimal sky-signature based algorithm.

To evaluate the relevance of the encoding (and of the algorithm), we compute the Jaccard similarity between the items of the planted signature and the items of the signature yielded by the sky-signature based algorithm. The results are presented in Figure 9.5.

The retrieved signature exactly corresponds to the planted signature for levels of noise up to 0.6. Then, the Jaccard similarity quickly drops when the amount of noise exceeds 1. The swarm plots details how the Jaccard similarity is dropping. Indeed, there is a clear separation between signatures having a Jaccard similarity of 1 and the other signatures, having a Jaccard similarity below 0.3. When analyzing signatures having a Jaccard similarity lower than 1, we noticed that all these signatures have 1 segment only. This means that the signature items correspond to the set of all items. Looking at the encoding of the data: $L(\alpha|S) = \sum_{S_i \in Seg} |\mathscr{R}| * \log(|S_i|) + (n + r - |\mathscr{R}| * |Seg|) * \log(|\mathscr{R}_{aug}|) + (e+n) * \log(|\mathscr{E}_{aug}|)$, one hypothesis is that the MDL encoding favors putting items from the noise in the signature if the number of baskets is significantly lower than the number of noise items. Indeed, when there is only 1

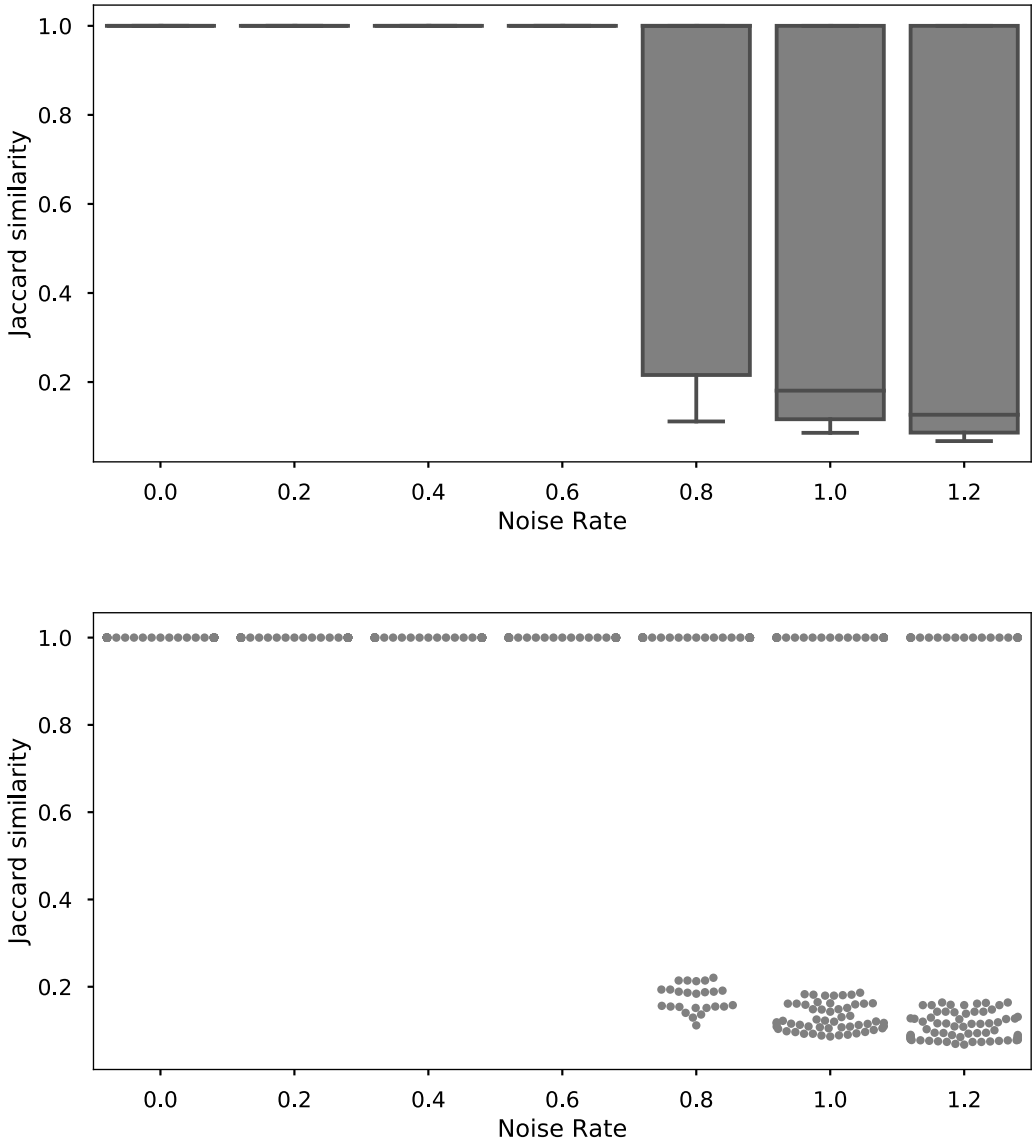

Figure 9.5: Boxplots (top) and swarm plot (bottom) of the Jaccard similarity between the planted signature items and the retrieved signature items, depending on the amount of noise. Each boxplot corresponds to 100 datasets.

segment, the first occurrence of each item will be encoded using $\log(|S_1|) = \log(n)$ bits. When computing the cost of encoding these items using the planted signature, the first occurrence for noise items will be encoded using $\log(|\mathscr{E}_{aug}|)$ bits. If the number of baskets n is lower than the number of noise items $|\mathscr{E}_{aug}|$, then the higher the noise rate, the more occurrences will be efficiently encoded using only 1 segment. From this analysis, it should follow that generated

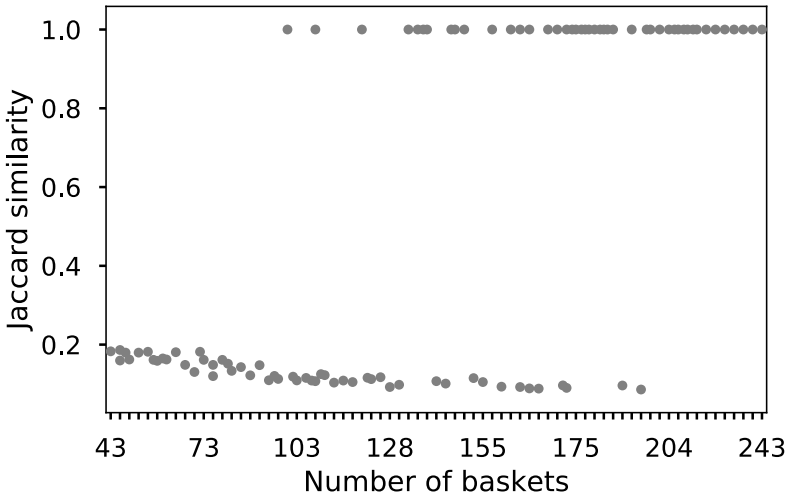

Figure 9.6: Swarm plot of the Jaccard similarity between the planted signature items and the retrieved signature items, depending on the number of baskets. This was computed on customers with a noise rate of 1.

customers with few baskets should have a lower Jaccard similarity. Figure 9.6 shows that this hypothesis is indeed verified. This result is actually reasonable, as sequences containing few baskets contain less information to discriminate noise items from signature items. Indeed, it is more difficult to identify noise if only a few samples are given. Overall, we can say that our encoding is robust to a reasonable amount of noise, but has difficulties to identify recurrences if the noise is large compared to the number of baskets.

9.4.3 Qualitative Analysis

Now that we have seen that the MDL encoding allows us to retrieve a signature even if it is hidden in (a reasonable amount of) synthetic noise, we analyze how the encoding is behaving on real customers. More specifically, we confirm that the simple greedy algorithm is sometimes not appropriate for model selection on real data. Again, we use 1000 customers from the Instacart dataset, having more than 60 baskets.

Among these 1000 customers, we first look at a specific customer exhibiting a typical result we get from running different instances of the widening algorithm and the algorithm based on sky-signature. We use the widening algorithm to get a variety of good signatures according to our MDL encoding. Good signatures are presented in Table 9.2. The first thing to notice is that all these signatures mostly contain the same products. All signatures can be seen as a variation of

Table 9.2: Signatures found by the different widening instances. The first signature is found by the algorithm based on optimal Sky-signature mining. The second signature is found with a beam size of 10 and $\beta \geq 0.2$. The third signature is found with a beam size of 5, and the last signature is found with a beam size of 1 (greedy algorithm). Common segmentation parts are highlighted in red and blue.

Encoding length	Signature products	segmentation
1703.86	Skim Milk, Soy Free Buttery Spread, Spring Water, Lavender Floral and Mint Natural Dish Liquid, Large White Eggs, Organic Gala Apples, Whole Almonds	1,11|12,16|17,18|19,26| 27,33|34,36|37,41| 42,44|45,47|48,51|52,55|56,60
1714.42	Same signature as above + Organic Fuji Apple	1,26| 27,33|34,36|37,41| 42,44|45,47|48,51|52,55|56,60
1717.33	First signature - Large White Eggs	1,4|5,8|9,13| 14,17|18,22|23,24| 25,28|29,32|33,36|37,41| 42,44|45,47|48,51|52,55|56,60
1738.39	First signature - Whole Almonds + Organic Fuji Apple	1,29|30,36|37,37|38,40|41,41| 42,44|45,47|48,51|52,55|56,60

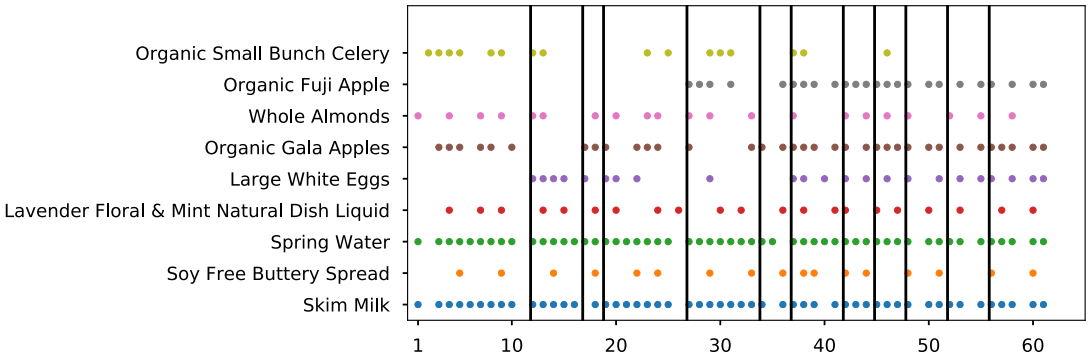

Figure 9.7: Timeline of products purchases of the best signature in Table 9.2. Each black line represent a segment boundary of the best signature in Table 9.2. *Organic Fuji Apple* (part of other signatures) and *Organic Small Bunch Celery* (most frequent item not included in a signature) are depicted as well.

the best one. Indeed, to get the second best signature, we just need to add *Organic Fuji Apple* to the best signature. Removing *Large White Eggs* from the best signature gives the third best one. The same can be observed for the segmentation. Indeed, all signatures share common segments,

in this case all segments from transaction 42 to transaction 60. This shows that even though a signature is good but not optimal according to our MDL encoding, the resulting signature still captures many regularities of the customer.

Figure 9.7 plots the purchases of this customer. All products except the first two (*Organic Small Bunch Celery* and *Organic Fuji Apple*) belong to the best signature. The last 4 products are clear favorite products, as they are bought over the whole sequence. *Large White Eggs* are bought a bit less regularly, so it makes sense that the third best signature considers that this product does not belong to the signature. The fact that the second signature adds *Organic Fuji Apple* in the signature can be understood, as it seems to be a favorite product starting from transaction 26. Finally, *Organic Small Bunch Celery* does not belong to any signature found by the widening instances, despite the fact that it is bought as many times as *Soy Free Buttery Spread*. This is understandable as Celery stops being bought from transaction 45, and is irregularly bought before. The only signature that makes less sense is the last one, that removes *Whole Almonds* from the signature. Indeed, this product seems to be bought regularly, with some inconsistencies. However, as this signature is the one with the worse encoding length, it is actually desirable that its quality seems worse than the better signatures. Overall, this analysis shows that the MDL encoding favors signatures that seem to correctly capture the favorite products of a customer. While this validation is qualitative, it still shows the relevance of the encoding, as well as the relevance of the signature model to analyze retail customer habits.

Table 9.3: Signatures found by the different widening instances. The first signature is found by the algorithm based on optimal Sky-signature mining. The second signature is found with a beam size of 10 and $\beta = 0.2$ or $\beta = 0.4$ (it is also found by all variants with a beam size of 20). The last signature is found with a beam size of 1 or 5 and any β value in $[0, 0.2, 0.4, 0.6]$.

Encoding length	Signature products	segmentation
4020.37	Organic Granola, Plain Bagels, Goo Berry Pie,Organic Whole Milk, Organic Orange Juice, Fettuccine,Cream Cheese Spread, Bananas,Vanilla 0% Milkfat Yogurt, Vanilla Skyr Yogurt, Whole Wheat Bread	1,24|25,31|32,36| 37,38|39,48|49,75
4025.64	Same signature as above + Organic Leaf Tea Bags	1,24|25,31|32,37| 38,48|49,75
4039.02	All products (146 in total)	1,75

We now illustrate the relevance of optimizing the MDL encoded length on a particular customer. This customer is interesting, as he exhibits an unusual behavior, that retailers are eager to find. Let us look at this customer having 75 purchases. Signatures found by the different

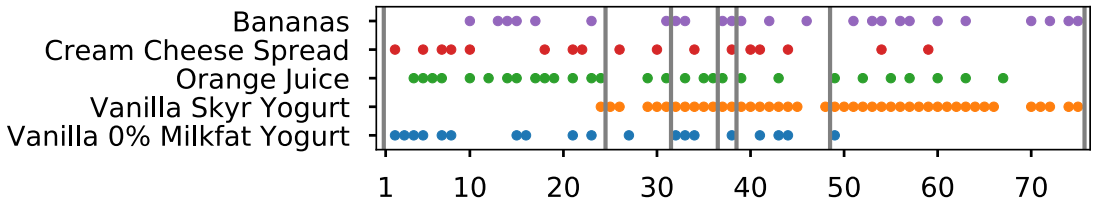

Figure 9.8: Timeline of the Vanilla Yogurt purchases. Each grey line represent a segment boundary of the best signature in Table 9.3. Some other signature products are depicted as well for illustration.

algorithms are presented in Table 9.3.

A first thing to note is that the encoded length of these three signatures are close, but their content can largely differ. For example, the signature having only one segment, with an encoding length of 4039.02 bits gives almost no information about the purchase rhythm of the customer. On the other hand, the best signature, with an encoding length of 4020.37 bits gives a more concise view of the regular products of the customer, as well as a more sophisticated rhythm analysis. Nevertheless, both encoding lengths only differ by 0.4%. This shows that small improvements in the encoding length can have a large impact on the resulting best signature.

The segmentation associated to the best signature contains 6 segments, with large first and last segments (containing 24 and 27 transactions respectively) and small remaining segments (containing between 2 and 10 transactions each). This type of segmentation could be a sign of an undesirable signature, as we noticed that Instacart customers tend to have signatures with similar segment sizes.

A deeper analysis of the purchases of this customer, however, shows that his purchase behavior is indeed different between the first segment, the last segment, and the remaining ones. As illustrated in Figure 9.8, this customer has the habit of purchasing two types of Vanilla Yogurt. We also plot purchases of some other products from the signature, to illustrate that the signature mainly captures products that are regularly purchased. During the first purchases, this customer only buys *Vanilla 0% Milkfat Yogurt*. Starting from transaction 24, however, this customer starts buying the *Vanilla Skyr Yogurt*. Then, until transaction 49, the customer buys both yogurts. After transaction 49, the customer solely buys *Vanilla Skyr Yogurt*. This type of behavior is called a *product switch*, as this customer gradually switched from one yogurt to another. This behavior is especially interesting for retailers, as they want to detect when customers are changing their habits.

Because the purchase data of this particular customer mainly contains the period where the

customer buys both yogurts, the signature items contain both products. However, the signature segmentation reflects the fact that the behavior of this customer is different in the first and last segments. Indeed, except for the first and last segments, the customer replenishes his signature items in a few transactions (between 2 and 10 transactions). The first and last segments, on the other hand, contain respectively 24 and 27 transactions. When looking at the purchases of the two yogurts, the segment length difference is clearly linked to the fact that the customer has a different purchase behavior in the first and last segments.

To summarize, the optimal signature found by our method allows the retailer to quickly identify unusual behavior thanks to the segmentation. Then by looking at the signature items purchases, the retailer can quickly identify the product switch operated by the customer.

Summary

- The optimal signature can be identified by optimizing the MDL encoded length
- Mining the optimal signature is difficult, so we rely on heuristics
- One efficient heuristic is based on sky-signatures, other efficient heuristics are based on a combination of beam search and diversity widening
- The best signature according to our encoding can highlight interesting behaviors for the retailer

CHAPTER 10

CONCLUSION

The main contribution of this thesis is the definition of a new type of pattern called a *signature*. The goal of the *signature* is to find recurrences in a sequence of events. The initial motivation for introducing the signature stems from the grocery retail context. In that context, the signature identifies the favorite products of a customer, by identifying the products that this customer buys regularly.

Hence, the signature has been defined to cope with erratic purchase behavior, while yielding an understandable output for the retailers. Having the flexibility to adapt to a sequence rhythm, while producing a simple output is what mainly differentiates the signature from existing methods (such as itemsets [AS+94], episodes [MTV97] or periodic patterns [MH01]), that usually produce complex results to cope with erratic behavior.

The initial definition of the signature is expressed as a pattern mining problem. One of the contribution of this thesis is to define the signature mining problem as a sequence segmentation problem. Having definitions rooted in two different fields allows the resolution of the signature mining problem to use solutions from both fields. Another contribution of this thesis is therefore to define efficient algorithms, stemming from both the pattern mining and the sequence segmentation fields, to solve the signature mining problem.

While the signature is useful to capture recurrences in a sequence given a number of occurrences, one can also be interested in mining all the recurrences. To this end, this thesis introduces a first extension of the signature model: the *sky-signature*. The *sky-signature* is a model that summarizes sequence recurrences at all time scales, whereas the signature gives the recurrences at a single time scale. Indeed, the sky-signature can be seen as an efficient method to summarize the signatures that can be computed when using all the possible number of occurrences. The sky-signature is therefore a more complex model, but enables practitioners to get the full picture of sequence recurrences. The sky-signature mining problem is efficiently solved by using an approach that combines pattern mining and sequence segmentation based approaches.

If one is interested in finding the single signature that best represents a sequence, this thesis

introduces a selection method that outputs the signature compressing the data the most. This selection technique is based on the Minimum Description Length principle, a well-known model selection technique in the pattern mining community. Apart from selecting the best signature, this selection process also automatically chooses the time scale fitting the data best.

Finally, the practical use of signatures and their extensions has been illustrated on 2 main use cases. The first use case is the initial motivation for the signature model: grocery retail data analysis. This thesis presents the analysis of a large set of retail customers (about 150 000 customers) from a French retailer. Thanks to the signature, we are able to analyze the purchase habits of a customer: her/his temporal purchase behavior, and her/his favorite products. We also illustrated that signatures can identify groups of customers that have similar purchase habits. This thesis also presents how signature can be used to identify and characterize specific retail behaviors, such as attrition or product switch.

The use of signatures is also illustrated on a Natural Language Processing task. Indeed, campaign speeches of D. Trump and H. Clinton during the 2016 presidential campaign were analyzed with the signature. The signature for each candidate contained the main topics of their speeches, as well as some of their campaign dynamics. Signatures were able to detect a subtle change of topics during the campaign speeches of D. Trump.

10.1 Perspectives

10.1.1 Study New Retail Use Cases

While the use of signatures has been illustrated on different use cases, signatures could be further processed to derive knowledge from customer habits. One first possibility would be to use the signature to group customers. Even though Section 6.5.3 compared the purchase habits of customers having different purchase rhythms, this thesis mainly focused on analyzing the signature of a single sequence. One first step would be to cluster signatures based on the recurrent items, their occurrences or both types of information. This step could be done as a post-processing to convey knowledge about large groups of customers. However, an even more interesting step would be to perform both the signature mining and the clustering at the same time. This means that this method would find signatures that fit groups of customers instead of fitting only one customer. This second step is more complex but would allow practitioners to capture signatures that represent groups of customers.

Using signatures on other use cases could also be interesting. Chapter 7 presented how

signatures can be used to detect and characterize attrition of retail customers. Signatures could also be used for other use cases, such as product recommendation, targeted marketing or stock optimization.

10.1.2 Study New Datasets

Signatures have mostly been studied on retail data. While this thesis studies signatures in a Natural Language Processing (NLP) setting as well, it would be interesting to apply signatures to other types of data. For example, one could apply signatures to analyze users of online music platforms. In that case, the signature would identify favorite songs of a user, as well as the moment she/he listens to these songs. While this setting seems similar to the retail one, the studied sequences are likely to have different dynamics. Indeed, for active users, listening to many songs a day should be quite common, whereas buying as many products everyday is less likely for a retail customer. Therefore, the frequency of events should be higher in sequences of music platform users. These users should also be more prone to intensive listening of a specific band an a short period of time, and then switching to another band. This behavior is less likely to happen in the retail context, where common products, such as milk, cheese, eggs or chocolate, are regularly bought by customers.

Using signatures in other NLP context could also be interesting. For example, analyzing news articles with signatures could identify perennial subjects or favorite subjects of a newspaper. Studying specific types of newspaper (sports, business, etc.) is also another possibility to identify recurrent events (such as sport tournaments for example).

All the types of dataset presented so far share the fact that they are sequences of events. Adding a new dimension in the dataset would be more challenging. For example, adding spatial information would enable the use of signatures in wider contexts. Datasets, such as taxi trips or in-game sport events both have a temporal and a spatial dimension. A first step to integrate the spatial dimension into the signature would be to discretize the spatial information. For example, most in-game sport events happen on a well defined pitch, that can be discretized using expert knowledge. For example, a football pitch has well defined zones such as the goal area or the penalty area. Using these game specific discretizations of space is usually meaningful to perform analysis on these types of data.

However, integrating spatial information directly within the signature mining is more attractive, but also more challenging. Indeed, as the signature is performing a segmentation on the temporal dimension, it seems desirable to also perform a segmentation on the spatial dimension. This would allow the signature to output spatial segmentations that depend on the sequence at

hand.

10.1.3 Improving the Signature Model

Finally, improving the signature model itself is an interesting perspective. Indeed, the current signature model has some drawbacks. For example, it does not handle the fact that within a segment, a product can be bought several times. We therefore loose the information about the quantity of each signature item within a segment. The sky-signature can partially tackle this issue by outputting signatures with different purchase rates, but it yields a complex output. Moreover, if a product occurs in many segments but not all of them, it is discarded from the signature. To tackle both challenges, one idea is to introduce a signature that is more resilient to variations in product quantities. A first solution is to post-process existing signatures and provide the practitioner with the information about products that have several occurrences within a segment, or products that occur in many segments but not all. A more challenging solution is to define a model that takes into account product quantity, as well as the possibility for products to occur in many segments but not all. For example, one could change the representative of the segment: instead of using the union of the products that belong to a segment, one could use the empirical distribution of products. Then, the reconstruction error of the segmentation could be expressed as a distance between distributions. In that case, the segmentation will return the probability density function that repeats itself overtime, as well as the occurrences of this probability distribution. This model would allow for some products to be absent in some segments. This model should also be able to capture the fact that products appear many times within a segment. Finding this optimal segmentation would however be more difficult, as distances between probability distributions, such as the Jensen-Shannon divergence [WY85], are more difficult to optimize than a simple set intersection.

BIBLIOGRAPHY

[AGM04] Foto Afrati, Aristides Gionis, and Heikki Mannila, "Approximating a collection of frequent sets", *in*: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2004, pp. 12–19.

[AIS93] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami, "Mining Association Rules Between Sets of Items in Large Databases", *in*: *Proc. 17th Int. Conf. on Management of Data*, 1993, pp. 207–216.

[AS+94] Rakesh Agrawal, Ramakrishnan Srikant, et al., "Fast algorithms for mining association rules", *in*: *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, 1994, pp. 487–499.

[AS95] Rakesh Agrawal and Ramakrishnan Srikant, "Mining sequential patterns", *in*: *Proc. 11th Int. Conf. on Data Engineering*, 1995, pp. 3–14.

[Ayr+02] Jay Ayres et al., "Sequential pattern mining using a bitmap representation", *in*: *Proc. 8th Conf. on Knowledge Discovery and Data mining*, 2002, pp. 429–435.

[Bel13] Richard Bellman, *Dynamic programming*, Courier Corporation, 2013.

[Bel61] Richard Bellman, "On the approximation of curves by line segments using dynamic programming", *in*: *Communications of the ACM* 4.*6* (1961), p. 284.

[Bin10] Ella Bingham, "Finding Segmentations of Sequences", *in*: *Inductive Databases and Constraint-Based Data Mining*, 2010, pp. 177–197.

[BKL09] Steven Bird, Ewan Klein, and Edward Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*, " O'Reilly Media, Inc.", 2009.

[BNJ03] David M Blei, Andrew Y Ng, and Michael I Jordan, "Latent dirichlet allocation", *in*: *Journal of machine Learning research* 3.*Jan* (2003), pp. 993–1022.

[Bos+18] Guillaume Bosc et al., "Anytime discovery of a diverse set of patterns with Monte Carlo tree search", *in*: *Data Mining and Knowledge Discovery* 32.*3* (2018), pp. 604–650.

[BP05] Wouter Buckinx and Dirk Van den Poel, "Customer base analysis: partial defection of behaviourally loyal clients in a non-contractual FMCG retail setting", *in*: *European Journal of Operational Research* 164.*1* (2005), pp. 252–268.

[BV15] Kailash Budhathoki and Jilles Vreeken, "The Difference and the Norm — Characterising Similarities and Differences Between Databases", *in*: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part II*, ed. by Annalisa Appice et al., Cham: Springer International Publishing, 2015, pp. 206–223, ISBN: 978-3-319-23525-7.

[BV17] Apratim Bhattacharyya and Jilles Vreeken, "Efficiently Summarising Event Sequences with Rich Interleaving Patterns", *in*: *Proc. of the SIAM International Conference on Data Mining*, 2017, pp. 795–803.

[CG03] Gemma Casas-Garriga, "Discovering unbounded episodes in sequential data", *in*: *European Conference on Principles of Data Mining and Knowledge Discovery*, Springer, 2003, pp. 83–94.

[Cue+12] Patricia López Cueva et al., "Debugging embedded multimedia application traces through periodic pattern mining", *in*: *Proc. 12th Int. Conf. on Embedded Software*, 2012, pp. 13–22.

[DRZ07] L. De Raedt and A. Zimmermann, "Constraint-Based Pattern Set Mining", *in*: *Proceedings of the Seventh SIAM International Conference on Data Mining*, Minneapolis, Minnesota, USA: SIAM, 2007, pp. 237–248.

[GC17] Derek Greene and James P. Cross, "Exploring the Political Agenda of the European Parliament Using a Dynamic Topic Modeling Approach", *in*: *Political Analysis* 25.*1* (2017), 77–94, DOI: 10.1017/pan.2016.7.

[GNDR13] Tias Guns, Siegfried Nijssen, and Luc De Raedt, "k-Pattern set mining under constraints", *in*: *Knowledge and Data Engineering, IEEE Transactions on* 25.*2* (2013), pp. 402–418.

[GNP06] Fosca Giannotti, Mirco Nanni, and Dino Pedreschi, "Efficient mining of temporally annotated sequences", *in*: *Proceedings of the 2006 SIAM International Conference on Data Mining*, SIAM, 2006, pp. 348–359.

[Grü07] Peter D Grünwald, *The minimum description length principle*, MIT press, 2007.

[Han+02] Jiawei Han et al., "Mining top-k frequent closed patterns without minimum support", *in*: *Proc. Int. Conf. on Data Mining (ICDM)*, 2002, pp. 211–218.

[HDY99] Jiawei Han, Guozhu Dong, and Yiwen Yin, "Efficient mining of partial periodic patterns in time series database", *in*: *Data Engineering, 1999. Proceedings., 15th International Conference on*, IEEE, 1999, pp. 106–115.

[HG04] Niina Haiminen and Aristides Gionis, "Unimodal segmentation of sequences", *in*: *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*, IEEE, 2004, pp. 106–113.

[HPY00] Jiawei Han, Jian Pei, and Yiwen Yin, "Mining frequent patterns without candidate generation", *in*: *ACM SIGMOD Record*, vol. 29, 2, ACM, 2000, pp. 1–12.

[IB13] Violeta N Ivanova and Michael R Berthold, "Diversity-driven widening", *in*: *International Symposium on Intelligent Data Analysis*, Springer, 2013, pp. 223–236.

[JMB00] Michael A Jones, David L Mothersbaugh, and Sharon E Beatty, "Switching barriers and repurchase intentions in services", *in*: *Journal of retailing* 76.*2* (2000), pp. 259–274.

[Lam+12] Hoang Thanh Lam et al., "Mining Compressing Sequential Patterns", *in*: *Proceedings of the Twelfth SIAM International Conference on Data Mining, Anaheim, California, USA, April 26-28, 2012*. SIAM / Omnipress, 2012, pp. 319–330.

[Li+06] Zhenmin Li et al., "CP-Miner: Finding copy-paste and related bugs in large-scale software code", *in*: *IEEE Transactions on software Engineering* 32.*3* (2006), pp. 176–192.

[Li+08] Haoyuan Li et al., "Pfp: parallel fp-growth for query recommendation", *in*: *Proceedings of the 2008 ACM conference on Recommender systems*, ACM, 2008, pp. 107–114.

[LK12] Matthijs van Leeuwen and Arno Knobbe, "Diverse subgroup set discovery", *in*: *Data Mining and Knowledge Discovery* 25.*2* (2012), pp. 208–242.

[Lo+09] David Lo et al., "Classification of software behaviors for failure detection: a discriminative pattern mining approach", *in*: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2009, pp. 557–566.

[LS99] Daniel D Lee and H Sebastian Seung, "Learning the parts of objects by non-negative matrix factorization", *in*: *Nature* 401.*6755* (1999), pp. 788–791.

[LSW97] Brian Lent, Arun Swami, and Jennifer Widom, "Clustering association rules", *in*: *Data Engineering, 1997. Proceedings. 13th International Conference on*, IEEE, 1997, pp. 220–231.

[LV14] Matthijs van Leeuwen and Jilles Vreeken, "Mining and using sets of patterns through compression", *in*: *Frequent Pattern Mining*, Springer, 2014, pp. 165–198.

[LXM08] Christopher LaRosa, Li Xiong, and Ken Mandelberg, "Frequent pattern mining for kernel trace data", *in*: *Proceedings of the 2008 ACM symposium on Applied computing*, ACM, 2008, pp. 880–885.

[Mac+67] James MacQueen et al., "Some methods for classification and analysis of multivariate observations", *in*: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, 14, Oakland, CA, USA., 1967, pp. 281–297.

[MF98] George Miller and Christiane Fellbaum, *Wordnet: An electronic lexical database*, 1998.

[MH01] Sheng Ma and Joseph L Hellerstein, "Mining partially periodic event patterns with unknown periods", *in*: *Data Engineering, 2001. Proceedings. 17th International Conference on*, IEEE, 2001, pp. 205–214.

[MTV97] Heikki Mannila, Hannu Toivonen, and A Inkeri Verkamo, "Discovery of frequent episodes in event sequences", *in*: *Data mining and knowledge discovery* 1.*3* (1997), pp. 259–289.

[NG08] Xavier Naturel and Patrick Gros, "Detecting repeats for video structuring", *in*: *Multimedia Tools and Applications* 38.*2* (2008), pp. 233–252.

[ORS98] Banu Ozden, Sridhar Ramaswamy, and Abraham Silberschatz, "Cyclic association rules", *in*: *Data Engineering, 1998. Proceedings., 14th International Conference on*, IEEE, 1998, pp. 412–421.

[PK03] Iko Pramudiono and Masaru Kitsuregawa, "Parallel FP-growth on PC cluster", *in*: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2003, pp. 467–473.

[Ris78] Jorma Rissanen, "Modeling by shortest data description", *in*: *Automatica* 14.*5* (1978), pp. 465–471.

[Ris83] Jorma Rissanen, "A universal prior for integers and estimation by minimum description length", *in*: *The Annals of statistics* (1983), pp. 416–431.

[Sav10] Jacques Savoy, "Lexical analysis of US political speeches", *in*: *Journal of Quantitative Linguistics* 17.*2* (2010), pp. 123–141.

[SM00] Jianbo Shi and Jitendra Malik, "Normalized cuts and image segmentation", *in*: *IEEE Transactions on pattern analysis and machine intelligence* 22.*8* (2000), pp. 888–905.

[Sou+11] Arnaud Soulet et al., "Mining dominant patterns in the sky", *in*: *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, IEEE, 2011, pp. 655–664.

[SRB94] Peter Shell, Juan Antonio Hernandez Rubio, and Gonzalo Quiroga Barro, "Improving search through diversity", *in*: *Proc. of the AAAI National Conf. on Artificial Intelligence*, AAAI Press, 1994, pp. 1323–1328.

[Ste+12] Keith Stevens et al., "Exploring topic coherence over many models and many topics", *in*: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Association for Computational Linguistics, 2012, pp. 952–961.

[SV12] Koen Smets and Jilles Vreeken, "Slim: Directly Mining Descriptive Patterns", *in*: *Proc. of the Twelfth SIAM International Conference on Data Mining*, SIAM / Omnipress, 2012, pp. 236–247.

[Ter06] Evimaria Terzi, "Problems and algorithms for sequence segmentations", PhD thesis, 2006.

[TT06] Evimaria Terzi and Panayiotis Tsaparas, "Efficient Algorithms for Sequence Segmentation", *in*: *Proc. SIAM Conference on Data Mining*, 2006, pp. 314–325.

[TV12] Nikolaj Tatti and Jilles Vreeken, "The Long and the Short of It: Summarising Event Sequences with Serial Episodes", *in*: *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, ACM, 2012, pp. 462–470, ISBN: 978-1-4503-1462-6.

[TWH01] Robert Tibshirani, Guenther Walther, and Trevor Hastie, "Estimating the number of clusters in a data set via the gap statistic", *in*: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63.*2* (2001), pp. 411–423.

[UKA04] Takeaki Uno, Masashi Kiyomi, and Hiroki Arimura, "LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets", *in*: *FIMI*, vol. 126, 2004.

[VHL12] Bay Vo, Tzung-Pei Hong, and Bac Le, "DBV-Miner: A Dynamic Bit-Vector approach for fast mining frequent closed itemsets", *in*: *Expert Systems with Applications* 39.*8* (2012), pp. 7196–7206.

[VLS11] Jilles Vreeken, Matthijs van Leeuwen, and Arno Siebes, "Krimp: mining itemsets that compress", *in*: *Data Mining and Knowledge Discovery* 23.*1* (2011), pp. 169–214, ISSN: 1384-5810.

[VT14] Jilles Vreeken and Nikolaj Tatti, "Interesting patterns", *in*: *Frequent pattern mining*, Springer, 2014, pp. 105–134.

[WY85] Andrew KC Wong and Manlai You, "Entropy and distance of random graphs with application to structural pattern recognition", *in*: *IEEE Transactions on Pattern Analysis and Machine Intelligence 5* (1985), pp. 599–609.

[ZH02] Mohammed Javeed Zaki and Ching-Jiu Hsiao, "CHARM: An Efficient Algorithm for Closed Itemset Mining.", *in*: *SDM*, vol. 2, SIAM, 2002, pp. 457–473.

[ZS14] Cäcilia Zirn and Heiner Stuckenschmidt, "Multidimensional topic analysis in political texts", *in*: *Data & Knowledge Engineering* 90 (2014), pp. 38–53.

Appendices

APPENDIX A

PUBLICATIONS

---

[1] B. Doux, C. Gautrais, and B. Negrevergne. Detecting strategic moves in hearthstone matches. In *Machine Learning and Data Mining for Sports Analytics Workshop of ECML/PKDD*, 2016.

[2] C. Gautrais, P. Cellier, T. Guyet, R. Quiniou, and A. Termier. Understanding customer attrition at an individual level: a new model in grocery retail context. In *International Conference on Extending Database Technology (EDBT)*, pages 686–687, 2016.

[3] C. Gautrais, P. Cellier, R. Quiniou, and A. Termier. Topic signatures in political campaign speeches. In *EMNLP 2017-Conference on Empirical Methods in Natural Language Processing*, pages 2342–2347, 2017.

[4] C. Gautrais, Y. Dauxais, and M. Guilleme. Multi-plant photovoltaic energy forecasting challenge: Second place solution. In *Discovery Challenges co-located with European Conference on Machine Learning-Principle and Practice of Knowledge Discovery in Database*, 2017.

[5] C. Gautrais, R. Quiniou, P. Cellier, T. Guyet, and A. Termier. Purchase signatures of retail customers. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 110–121. Springer, 2017.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre: Signatures : Détection et Caractérisation de comportements complexes récurrents dans des données séquentielles

Mot clés : Exploration de données, Analyse des données symboliques, Bases de données temporelles

Resumé : Cette thèse introduit un nouveau type de motif appelé *signature*. La signature segmente une séquence d'itemsets, afin de maximiser la taille de l'ensemble d'items qui apparait dans tous les segments. La signature a été initialement introduite pour identifier les produits favoris d'un consommateur de supermarché à partir de son historique d'achat. L'originalité de la signature vient du fait qu'elle identifie les items récurrents qui 1) peuvent apparaitre à différentes échelles temporelles, 2) peuvent avoir des occurrences irrégulières et 3) peuvent être rapidement compris par des analystes. Etant donné que les approches existantes en fouille de motifs n'ont pas ces 3 propriétés, nous avons introduit la signature.

En comparant la signature avec les méthodes de l'état de l'art, nous avons montré que la signature est capable d'identifier de nouvelles régularités dans les données, tout en identifiant les régularités détectées par les méthodes existantes. Bien qu'initialement liée au domaine de la fouille de motifs, nous avons également lié le problème de la fouille de signatures au domaine de la segmentation de séquences. Nous avons ensuite défini différents algorithmes, utilisant des méthodes liées à la fouille de motifs et à la segmentation de séquences. Les signatures ont été utilisées pour analyser un large jeu de données issu d'un supermarché français. Une analyse qualitative des signatures calculées sur ces consommateurs réels a montré que les signatures sont capables d'identifier les produits favoris d'un consommateur. Les signatures ont également été capables de détecter et de caractériser l'attrition de consommateurs.

Cette thèse définit également 2 extensions de la signature. La première extension est appelée la *sky-signature*. La sky-signature permet de présenter les items récurrents d'une séquence à différentes échelles de temps. La sky-signature peut être vue comme une manière efficace de résumer les signatures calculées à toutes les échelles de temps possibles. Les sky-signatures ont été utilisées pour analyser les discours de campagne des candidats à la présidentielle américaine de 2016. Les sky-signatures ont identifié les principaux

thèmes de campagne de chaque candidat, ainsi que leur rythme de campagne. Cette analyse a également montré que les signatures peuvent être utilisées sur d'autres types de jeux de données.

Cette thèse introduit également une deuxième extension de la signature, qui permet de calculer la signature qui correspond le plus aux données. Cette extension utilise une technique de sélection de modèle basée sur le principe de longueur de description minimale, communément utilisée en fouille de motifs. Cette extension a également été utilisée pour analyser des consommateurs de supermarché.

Title: Signatures: Detecting and Characterizing Complex Recurrent Behavior in Sequential Data

Keywords : Data Mining, Pattern Mining, Sequential Data

Abstract : This thesis introduces a new type of pattern called a *signature*. The signature segments a sequence of itemsets, so that the set of items that occurs in every segment is the largest. The signature was initially introduced to find the favorite products of a retail customer from her/his purchase sequence. The particularity of the signature is that these recurrent items can 1) occur at different time scales, 2) have non regular occurrences and 3) be easily understood by practitioners. As previous approaches from the pattern mining field are not able to tackle these 3 challenges, we defined the signature.

Experiments comparing the signature with state-of-the-art methods from the pattern mining field shows that the signature is able to capture new regularities in the data, while identifying regularities found by existing methods. While initially defined within the pattern mining field, we also express the signature mining problem as a sequence segmentation problem. We then defined algorithms mining signatures based on pattern mining and sequence segmentation ideas. The practical use of signatures has been illustrated on a large dataset from a French retailer. A qualitative analysis of signatures computed on real customers shows that signatures are indeed able to identify favorite products of retail customers. Signatures have also been used to detect and characterize attrition of retail customers.

This thesis also defines 2 extensions of the signature model. The first extension is called a *sky-signature*. The sky-signature is a model that summarizes sequence recurrences at all time scales, whereas the signature gives the recurrences at a single time scale. In-

deed, the sky-signature can be seen as an efficient method to summarize the signatures that can be computed when using all the possible number of occurrences. Sky-signatures have been used to analyze campaign speeches of candidates during the 2016 U.S. presidential campaign. Sky-signatures identified candidates main topics as well as their campaign dynamics. This analysis of campaign speeches also shows that the signature can be used to analyze different types of datasets.

This thesis introduces a second extension of the signature that outputs the signature that best fits the data. It uses a model selection technique based on the Minimum Description Length principle, a well-known selection technique in the pattern mining community. Apart from selecting the best signature, this selection process also automatically chooses the time scale fitting the data best. This extension is also successfully used to analyze retail customers.