Analyse des comportements des clients sur un site marchand en ligne

Mohamad Kanaan

N° d'ordre NNT : 2020LYSE1227

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'Université Claude Bernard Lyon 1

Ecole Doctorale N°512
Informatique et Mathématique (InfoMaths)

Spécialité de doctorat : Informatique

Soutenu publiquement le 15/12/2020, par :
Mohamad KANAAN

Analyse des comportements des clients sur un site marchand en ligne

---

Devant le jury composé de :

Mme. Hanane Azzag, Maître de conférences - HDR, Université Paris 13, Sorbonne Paris Cité	Rapporteure
M. Jean-Luc Baril, Professeur, Université de Bourgogne	Rapporteur
M. Mohand-Saïd Hacid, Professeur, Université Lyon 1	Examinateur
Mme. Lynda Tamine-Lechani, Professeure, Université Paul Sabatier	Examinatrice
M. Hamamache Kheddouci, Professeur, Université Lyon 1	Directeur de thèse
M. Thierry Barreyre, Directeur des opérations, Sistema-Strategy	Invité
M. Khalid Benabdeslem, Maître de conférences - HDR, Université Lyon 1	Invité
M. Rémy Cazabet, Maître de conférences, Université Lyon 1	Invité

Université Claude Bernard – LYON 1

Administrateur provisoire de l'Université	M. Frédéric FLEURY
Président du Conseil Académique	M. Hamda BEN HADID
Vice-Président du Conseil d'Administration	M. Didier REVEL
Vice-Président du Conseil des Etudes et de la Vie Universitaire	M. Philippe CHEVALLIER
Vice-Président de la Commission de Recherche	M. Jean-François MORNEX
Directeur Général des Services	M. Pierre ROLLAND

COMPOSANTES SANTE

Département de Formation et Centre de Recherche en Biologie Humaine	Directrice : Mme Anne-Marie SCHOTT
Faculté d'Odontologie	Doyenne : Mme Dominique SEUX
Faculté de Médecine et Maïeutique Lyon Sud - Charles Mérieux	Doyenne : Mme Carole BURILLON
Faculté de Médecine Lyon-Est	Doyen : M. Gilles RODE
Institut des Sciences et Techniques de la Réadaptation (ISTR)	Directeur : M. Xavier PERROT
Institut des Sciences Pharmaceutiques et Biologiques (ISBP)	Directrice : Mme Christine VINCIGUERRA

COMPOSANTES & DEPARTEMENTS DE SCIENCES & TECHNOLOGIE

Département Génie Electrique et des Procédés (GEP)	Directrice : Mme Rosaria FERRIGNO
Département Informatique	Directeur : M. Behzad SHARIAT
Département Mécanique	Directeur M. Marc BUFFAT
Ecole Supérieure de Chimie, Physique, Electronique (CPE Lyon)	Directeur : Gérard PIGNAULT
Institut de Science Financière et d'Assurances (ISFA)	Directeur : M. Nicolas LEBOISNE
Institut National du Professorat et de l'Education	Administrateur Provisoire : M. Pierre CHAREYRON
Institut Universitaire de Technologie de Lyon 1	Directeur : M. Christophe VITON
Observatoire de Lyon	Directrice : Mme Isabelle DANIEL
Polytechnique Lyon	Directeur : Emmanuel PERRIN
UFR Biosciences	Administratrice provisoire : Mme Kathrin GIESELER
UFR des Sciences et Techniques des Activités Physiques et Sportives (STAPS)	Directeur : M. Yannick VANPOULLE
UFR Faculté des Sciences	Directeur : M. Bruno ANDRIOLETTI

À ma belle famille.

Remerciement

Je tiens tout d'abord à remercier mon directeur de thèse Pr. Hamamache Kheddouci, ainsi que Thierry Barreyre, Directeur des opérations au sein de l'entreprise Sistema-Strategy. Je vous remercie pour votre patience, vos disponibilités et surtout vos conseils qui ont contribué à alimenter ma réflexion.

Je souhaite aussi exprimer ma gratitude à la société Sistema-Strategy en la personne de monsieur Franck Seimann, Président, pour l'intérêt dont il a fait preuve envers ma recherche et les moyens qu'il a mis en œuvre.

Je voudrais aussi remercier M. Benabdeslem et M. Cazabet pour leurs aides, leurs corrections et leurs idées qui m'ont beaucoup inspiré.

Je voudrais aussi remercier toute l'équipe LIRIS. J'ai particulièrement apprécié votre appui et vos conseils durant ma thèse.

Finalement, j'adresse mes remerciements les plus chaleureux à ma famille. À mon père, ma mère, mes frères et ma sœur. Je vous remercie infiniment pour votre support inconditionnel. Vous avez toujours été là quand il le fallait pour m'encourager et me soutenir. Vous avez su trouver les mots juste pour me redonner la force et la détermination indispensables pour réussir. J'espère que vous êtes fiers de moi comme toujours.

Résumé

Au cours des dernières années, le commerce électronique a révolutionné le commerce de détail en modifiant la manière dont les clients achètent. Les choix de l'utilisateur ne sont plus limités par la disponibilité des produits dans les magasins de sa région. Aujourd'hui, il peut chercher et acheter des produits dans n'importe quel magasin dans le monde, n'importe où et n'importe quand, à travers les sites de e-commerce. Cependant, les commerçants en ligne ont quelques difficultés à comprendre le comportement de leurs clients : Pourquoi un client choisit-il un produit plutôt qu'un autre ? Pourquoi préfère-t-il une marque à une autre ? Pourquoi visite-t-il le même produit plus souvent ? Pourquoi l'achat est-il effectué pendant la pause déjeuner ? Quels sont les facteurs qui influencent son choix et ses achats ?

Ces comportements sont très complexes car ils sont influencés par (1) des facteurs internes au site d'e-commerce tels que les prix, les disponibilités des produits, les délais des livraisons, les notations des produits, les opinions des utilisateurs, etc., (2) et des facteurs externes, tels que les événements internationaux (par exemple, le Black Friday), les vacances, le budget du client, etc. Pour en savoir plus sur le comportement du consommateur, une analyse plus approfondie est nécessaire. Une analyse fine du comportement du client consiste à comprendre son parcours sur le site. L'analyse des comportements aident les commerçants en ligne à comprendre et à découvrir les besoins des clients. Par conséquent, les commerçants en ligne peuvent recommander des produits pertinents à leurs clients, prévoir les futurs achats et s'assurer de la disponibilité des produits. Leurs tâches principales seront davantage axées sur l'identification et la compréhension des processus d'abandon et d'achat, depuis le premier stimulus du client jusqu'à sa décision.

Pour atteindre cet objectif, les actions des utilisateurs pourraient être retracées. Ces actions sont connues sous le nom de "flux de clics". Pour les analyser et en extraire des informations utiles, plusieurs outils d'exploitation ont été développés. Ils visent à analyser le parcours de l'utilisateur et à découvrir ses comportements cachés. Certaines techniques d'exploration de données bien connues sont : l'exploration de modèles, la découverte de tendances, le re-

groupement et la classification des clients, le filtrage collaboratif dans les systèmes de recommandation et la prédiction d'achat.

Nous présentons dans cette thèse deux nouveaux modèles de comportements utilisateurs. Le premier modèle, appelé "comportement fréquent", représente les comportements qui sont similaires entre plusieurs sessions d'utilisateurs. Pour les modéliser, nous proposons une nouvelle représentation de motif fréquent et temporel et pour les extraire nous proposons un algorithme de détection de motifs appelé SEPM (*Sequential Event Pattern Mining*) basé sur cette nouvelle représentation. Le deuxième modèle, appelé "comportement intéressant", représente les comportements qui se ressemblent dans l'évolution de leurs caractéristiques dans le temps. Nous les modélisons par des séries temporelles multi-variées et nous proposons un framework appelé GCBag (*Generative time series Clustering with Bagging*) pour les détecter. En plus de ces deux modèles, nous présentons le processus d'analyse complet de ces modèles partant de la préparation des données, passant par l'application des algorithmes et jusqu'à l'analyse des comportements résultants.

Mots clés— e-commerce, comportement des utilisateurs, détection de patterns, séries temporelles, clustering.

Abstract

In recent years, electronic (e)-commerce has revolutionized retail by changing the way customers shop. The user's choices are no longer limited by the availability of products in stores in his or her area. Today, they can search and purchase products from any international store, anywhere, anytime, using e-commerce sites. However, e-merchants have some difficulty understanding the behavior of their customers: Why does a customer choose one product over another? Why do they prefer one brand over another? Why does he visit the same product more often? Why is the purchase made during the lunch break? What factors influence their choice and purchases?

These behaviors are very complex because they are influenced by (1) factors internal to the e-commerce site such as prices, product availability, delivery times, product ratings, user opinions, etc., (2) and external factors, such as international events (e.g. Black Friday), holidays, the customer's budget, etc., (3). To learn more about consumer behavior, a more in-depth analysis is required. A fine analysis of the customer's behavior consists of understanding his or her path on the web site. These behaviors help e-merchants to understand and discover customer needs. As a result, e-merchants can recommend relevant products to their customers, anticipate future purchases and ensure product availability. Their main tasks are more focused on identifying and understanding the abandonment and purchase process, from the customer's first stimulus to their decision.

To achieve this objective, user actions could be tracked. These actions are known as "clickstreams". To analyze them and extract useful information from them, several operating tools have been developed. They aim to analyze the user's path and discover their hidden behaviors. Some well-known data mining and machine learning techniques are pattern mining, trend discovery, customer grouping and classification, collaborative filtering in recommendation systems, and purchase prediction.

In this thesis, we present two new models of user behaviors. The first model, called "frequent behavior", represents behaviors that are similar between several user sessions. To model

them, we propose a new representation of frequent and temporal patterns, and to extract them we propose the pattern detection algorithm called SEPM (*Sequential Event Pattern Mining*) based on this new representation. The second pattern, called "interesting behavior", represents behaviors that are similar in the evolution of their characteristics over time. We model them by multivariate time series and we propose the GCBag framework (*Generative time series Clustering with Bagging*) to detect them. In addition to these two models, we present the complete analysis process of these models starting from the preparation of the data, through the application of the algorithms and up to the analysis of the resulting behaviors.

Keywords— e-commerce, customer behavior, fequent pattern, time series, clustering.

Table des matières

Table des figures

Liste des tableaux

1

Introduction

1.1 Contexte et motivations

Le commerce électronique peut être défini comme l'utilisation d'un média électronique pour effectuer des transactions commerciales afin d'échanger des biens ou des services. Il a révolutionné le commerce de détail en réduisant la distance entre les magasins et les clients. De nos jours, les sites de e-commerce proposent à leurs clients une large gamme de produits et de services. Contrairement aux magasins physiques, les sites de e-commerce permettent aux entreprises de stocker un grand nombre de produit sans tenir compte du coût de l'inventaire. Par conséquent, une entreprise peut offrir aux consommateurs un large choix pour qu'ils puissent

choisir un produit de leur choix. En achetant sur internet, les utilisateurs peuvent économiser de l'argent, du temps, et avoir plus d'informations sur les produits et surtout les retours des autres clients. Cette attractivité a considérablement augmenté le nombre de visiteurs des sites de commerce électronique, incitant les vendeurs en ligne à améliorer leurs sites et à proposer des produits plus personnalisés, en comprenant mieux les comportements de leurs clients. Ces comportements sont très complexes, car ils sont influencés par différents facteurs, par exemple le profil des clients (âge, sexe, pays, emploi ...), les services proposés (mode de paiement, diversité des produits, mode de livraison ...), etc. La compréhension de ces comportements aide les vendeurs en ligne à recommander des produits plus pertinents, à prédire les futurs achats et à assurer la disponibilité des produits. Pour améliorer encore les performances de leur site web, les vendeurs en ligne veulent savoir comment les clients achètent les produits, comment ils naviguent à travers les catalogues de produits ou même pourquoi ils abandonnent certains processus d'achat. Ces informations peuvent les aider à améliorer leurs sites web et à proposer des produits plus pertinents et personnalisés.

Pour atteindre cet objectif, de nombreux outils ont été développés pour tracer les actions des utilisateurs (cliquer, visualiser, ajouter au panier, acheter, etc.). Ces traces sont communément appelées **clickstreams**. Ensuite, en utilisant certaines techniques de data mining et machine learning, les clickstreams sont exploités pour extraire des connaissances utiles. Certaines techniques d'exploration de données bien connues utilisées à cette fin sont : l'exploration de modèles [27, 53], la découverte de tendances [42, 94], le regroupement et la classification des clients [23, 3, 9], le filtrage collaboratif dans les systèmes de recommandation [60, 91, 61] et la prédiction d'achat [40, 98, 56].

Du point de vue applicatif, cette thèse est le fruit d'une collaboration entre l'entreprise Sistema-Strategy et le laboratoire LIRIS. Elle consiste à fournir des modèles mathématiques et algorithmiques pour aider les e-commerçant à analyser les traces de leurs clients, extraire leurs comportements et aider à la décision. L'objectif final de la thèse est de fournir les bases scientifiques d'une plateforme d'aide à la décision et qui sera dédié aux futurs clients de Sistema-

Strategy. À la suite de la thèse, nous développerons au sein du pôle R&D de Sistema-Strategy, une plateforme d'aide à la décision contenant les approches proposées dans cette thèse, et qui sera commercialisée et maintenue tout au long de son utilisation.

1.2 Contribution

L'objectif de ma thèse est donc de mettre à la disposition des e-commerçants, une plateforme d'aide à la décision en s'appuyant sur les comportements de leurs utilisateurs. L'idée est d'utiliser des modèles mathématiques, des algorithmes avancés, et des technologies de Data Mining et de Machine Learning, pour modéliser et analyser d'une façon poussée les traces d'utilisations des sites de e-commerce.

Le travail présenté dans cette thèse se divise en deux parties : dans la première partie, nous étudions les comportements fréquents via la suite d'exécution d'actions que nous représenterons sous forme de schéma séquentiels et temporels appelés "pattern". Nous avons proposé, dans un premier temps, d'augmenter la représentation des patterns en ajoutant la durée passée par l'utilisateur sur chaque produit. Cette durée est cruciale dans l'analyse du pattern comme elle reflète l'intérêt du client par le produit. Nous avons aussi développé une série d'algorithmes qui permettent d'extraire ce type de patterns, et nous avons proposé ensuite des méthodes d'interprétation de ces patterns. Dans la deuxième partie, nous élargissons notre enquête pour étudier les comportements intéressants des utilisateurs. Ces comportements sont modélisés par des séries temporelles qui représentent l'ensemble du parcours des utilisateurs sur le site d'e-commerce. Ces séries temporelles sont ensuite regroupées dans des clusters pour extraire les comportements intéressants. En fin, nous montrons comment à partir des données brutes, les données sont préparées (nettoyées et transformées), comment les algorithmes sont appliqués et comment les résultats sont analysés.

1.3 Organisation du manuscrit

Cette thèse se scinde en plusieurs chapitres :

- Dans le chapitre 2, nous présentons les différents travaux en relations avec nos problématiques. Ces travaux sont divisés en deux parties. Ceux en relation avec la détection de pattern et les autres en relation avec le clustering des séries temporelles.

- Dans le chapitre 3, nous présentons notre première contribution dans la quelle le comportement fréquent est défini. Ce type de comportement est caractérisé par des patterns fréquents, séquentiels et temporels. Dans un premier temps, nous proposons un format augmenté du pattern pour intégrer la durée passé sur chaque produit. Dans un deuxième temps nous décrivons notre algorithme SEPM (Sequential Event Pattern Mining) qui permet d'extraire ces patterns. Finalement, nous menons une première série d'expériences pour évaluer la capacité de ces patterns à modéliser des comportements fréquents et discriminants, et ensuite une deuxième série pour évaluer les performances de ces algorithmes. Les résultats expérimentaux indiquent que nos algorithmes sont efficaces et évolutifs.

- Dans le chapitre 4, nous élargissons notre étude en proposant un framework GCBag (Generative time series Clustering with Bagging) qui permet d'extraire des comportements intéressants. Le but de cette étude est d'analyser le parcours d'un client dans sa globalité, depuis son arrivée au site d'e-commerce et jusqu'à l'achat ou l'abandon. Nous modélisons les traces par des séries temporelles et nous appliquons notre framework pour extraire les clusters. Les tests réalisés sur des données synthétiques et réels, ont prouvé que GCBag est capable d'améliorer la qualité du clustering des séries temporelles.

- Dans le chapitre 5, nous présentons une application de nos contributions sur des données réelles issues du site marchand de gadget de Google. Cette application représente

une preuve de faisabilité de nos algorithmes. Nous détaillons dans ce chapitre la transformation des données et l'analyse des résultats retournés par nos algorithmes.

- Nous concluons le manuscrit dans le chapitre 6, en résumant les principales contributions et en présentant les perspectives à court et à moyens terme.

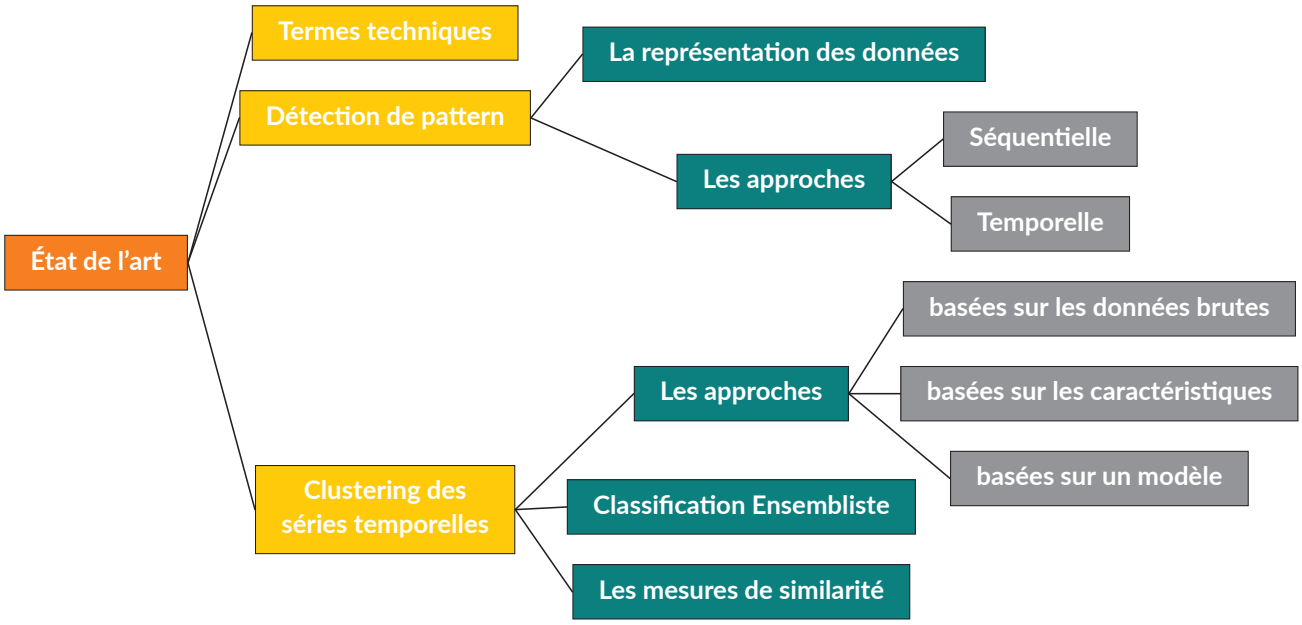
2

État de l'art

Dans ce chapitre, nous nous intéressons à deux domaines d'extraction de l'information : la détection de patterns et le clustering des séries temporelles. Nous présentons ici, les différentes méthodes en relation avec ces domaines et qui peuvent apportées des solutions à notre problème. Nous étudions dans la section 2.2 l'extraction des patterns pour répondre aux questions suivantes : comment les données brutes sont modélisées ? Comment les algorithmes parcourent-ils l'ensemble des données pour identifier les patterns ? Quels sont les types de patterns étudiés ? Ensuite nous nous focalisons dans la section. 2.3 sur le clustering des séries temporelles pour savoir : comment les séries temporelles sont modélisées ? Comment mesurer la similarité entre eux ? Comment les classifier ?

Plan du chapitre

L'étude du comportement de l'utilisateur est une tâche complexe et pluridisciplinaire. Elle fait souvent appel à plusieurs domaines scientifiques tel que l'analyse de l'arbre de décision [41], la prédiction [67], la recommandation [76], etc. Dans cette thèse, nous nous sommes intéressés à deux types de comportements : le fréquent et l'intéressant. Le comportement fréquent est modélisé par des séquences d'actions temporelles et fréquentes, et peut être extrait en utilisant la "Détection de patterns fréquents". Le comportement intéressant est modélisé par un ensemble de séquence de sessions utilisateurs, et peut être étudié à travers le "Clustering des séries temporelles". Dans la suite de ce chapitre, nous présentons et discutons les différents métriques, méthodes et algorithmes présents dans la littérature et qui correspondent à ces deux domaines.



2.1 Termes techniques

Dans cette section, nous détaillons quelques termes et techniques utilisées dans la suite du manuscrit.

2.1.1 Le commerce électronique

Le (e)-commerce électronique, désigne l'achat et la vente de biens ou de services sur Internet. Il existe plusieurs types de transactions e-commerce :

- Business to Consumer (B2C) : une entreprise vend un bien ou un service à un consommateur individuel
- Business to Business (B2B) : une entreprise vend un bien ou un service à une autre entreprise
- Consumer to Consumer (C2C) : un consommateur vend un bien ou un service à un autre consommateur
- Consumer to Business (C2B) : un consommateur vend un bien ou un service à une entreprise

Pour pouvoir faciliter les achats des utilisateurs, mieux recommander les produits ou prédire les achats, les e-commerçants utilisent des outils pour tracer les interactions des utilisateurs sur leur site.

2.1.1.1 Outils de traçage

Un outil de traçage est un script (souvent rédigé en Java-Script) ajouté aux site web et qui permet d'enregistrer les interactions avec le site web. Il enregistre pour chaque visiteur l'ensemble de ses sessions. Il crée pour chaque utilisateur un ID unique. De même pour les sessions.

Ces outils de traçage doivent en même temps protéger les données personnelles des utilisateurs. Ces données personnelles peuvent être : des identifiants directes (nom de la personne ou de l'entreprise, la carte bancaire, le numéro de téléphone, ...), des identifiants indirects (la date de naissance, le sexe, l'addresse, ...).

L'anonymisation rend impossible l'identification d'une personne à partir d'un jeu de données et permet, ainsi, de respecter sa vie privée. L'anonymisation est un traitement qui consiste à utiliser un ensemble de techniques de manière à rendre impossible, en pratique, toute identification de la personne par quelque moyen que ce soit et de manière irréversible. [1]

2.1.1.2 Sessions

Une session est une séquence d'interactions d'un utilisateur avec un site. Elle correspond à une et une seule consultation du site sur une période de temps. Chaque session contient des informations générales et des informations particulières en relation avec chaque interaction. Selon l'outil de traçage utilisé, ces informations peuvent être :

- Des informations générales sur la session :
  - La durée de la session
  - Le pays
  - L'heure de début
  - le support utilisé (ordinateur, téléphone, tablette, ...)
  - ...
- Des informations sur chaque interaction :
  - L'heure de début de l'interaction
  - Le nom, la catégorie, le prix du produit visualisé
  - ...

[1] Source : https://www.cnil.fr/fr/lanonymisation-de-donnees-personnelles

2.1.2 Thésaurus

Nous fournissons, dans le tableau 2.1, la traduction (anglais → français) de quelques mots techniques et qui sont utiles pour la suite du manuscrit.

anglais	**français**
pattern	motif
support	nombre d'apparition
cluster	groupe
framework	structure logicielle

Table 2.1: Thésaurus

2.1.3 Détection de pattern fréquent

La détection de pattern fréquent (appelé en français "motif fréquent") est la reconnaissance automatisée des patterns fréquents dans une base de données. Un pattern est un sous-ensemble composé des éléments de cette base de données. La contrainte de fréquence est imposée pour filtrer les sous-ensembles possibles et en garder les plus récurrents. Nous calculons pour chaque sous-ensemble son "support" qui est égal à son nombre d'apparitions dans l'ensemble des données. Si ce support est supérieur à une limite choisie par l'utilisateur, nous appelons le sous-ensemble un "pattern fréquent".

Un exemple d'une base de données contenant des achats est affiché dans la Table 2.2. Si nous fixons un minimum de support égal à 3, un algorithme de détection de patterns fréquents retourne les patterns suivants : {Café}, {Biscuit}, et {Biscuit, Café}. À partir de ce résultat, nous pouvons analyser les produits qui sont souvent achetés ensemble, comme les produits "Biscuit" et "Café". Grâce à ce résultat, un commerçant peut par exemple réorganiser les rayons pour mettre les biscuits en face du café pour faciliter les déplacements des clients dans un magasin physique, ou réorganiser les pages web dans le cas du e-commerce.

ID	Éléments
1	Banane, Café, Biscuit
2	Café, Biscuit, Croissant
3	Pomme, Banane
4	Biscuit, Café
5	Café, Lait, Croissant
6	Baguette

Table 2.2: Exemple d'une base de données d'achat. La colonne ID correspond à un achat, et la colonne Éléments corresponds aux produits achetés.

2.1.4 Clustering

Une des techniques de machine learning très utilisée pour étudier la répartition des données dans l'espace (comme des points dans un espace 2D) est le clustering (ou la classification non supervisée en français). Elle consiste à diviser un ensemble d'objets en plusieurs sous-ensembles homogènes appelés clusters. Chaque cluster regroupe les points qui sont les plus proches. Un exemple de points dans un espace 2D est affiché dans la Figure 2.1. Après l'application d'un algorithme de clustering (comme K-Means), nous obtenons 5 clusters affichés dans la Figure 2.2.

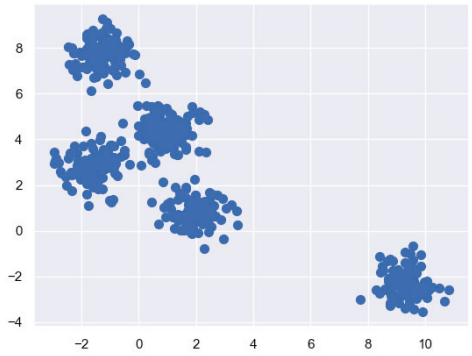


Figure 2.1: Exemple de points dans un espace 2D (avant le clustering).

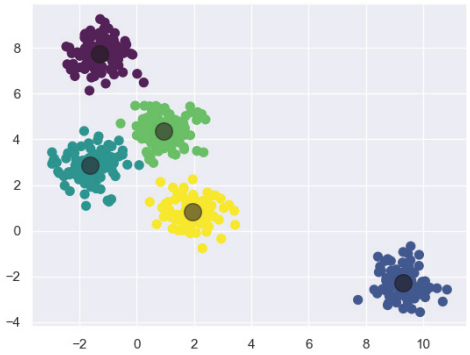


Figure 2.2: Les points de la figure 2.1 après le clustering en utilisant l'algorithme K-Means. Chaque couleur correspond à un cluster identifié.

Une mesure de similarité est utilisée entre les objets pour calculer leurs ressemblances (leurs distances si c'est possible). Ceux qui se ressemblent le plus sont mis ensemble dans le même cluster. Ce qui signifie que dans le même cluster, les objets doivent être le plus semblables que possible les uns aux autres (similitudes internes) et aussi différents que possible des autres dans un autre cluster (similitudes externes).

Souvent, le nombre de cluster est fournis en entrée à l'algorithme. Quand nous ne connaissons pas au préalable le nombre de cluster, des techniques sont appliquées pour trouver le nombre le mieux adapté à nos données comme la méthode d'Elbow.

2.1.5 Séries temporelles.

Une série temporelle est une séquence de variables unie ou multi-variées ordonnées dans le temps, où le temps n'est pas utilisé comme une caractéristique, mais plutôt comme un axe. Elle est un ensemble d'observations sur les valeurs qu'une variable prend à différents moments. Un exemple de séries temporelles uni-variées est affiché dans la Figure 2.3.

Les séries temporelles représentent un ensemble de variables qui varient dans le temps. Chaque observation dans la série, représente un vecteur contenant les valeurs de ces variables

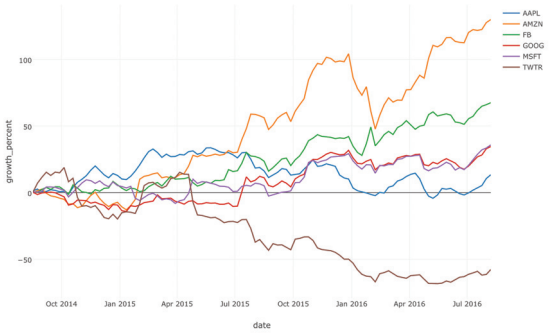


Figure 2.3: Exemple de séries temporelles uni-variées. L'axe des abscisses représente le temps et l'axe des ordonnées représente les valeurs.

Longueur	Largeur	Hauteur
50	94	140
77	85	135
64	92	142
59	97	150

Table 2.3: Exemple d'une série temporelle représentée en matrice.

à un instant t. Nous pouvons modéliser une série par une matrice. Une série correspond à une matrice, avec un nombre de colonnes (features) fixe mais le nombre de lignes (observations) peut changer (ex. la Table 2.3 affiche la modélisation d'une série temporelle par une matrice).

2.1.6 Hidden Markov Model

Le modèle de Markov caché (HMM) décrit la relation entre deux processus stochastiques. Le premier est caché et suit une chaîne de Markov. Le second est observé et modélisé comme une condition dépendante de la séquence des états cachés (voir la Figure. 2.5).

Un processus stochastique ou processus aléatoire représente une évolution, discrète ou à temps continu, de plusieurs variables aléatoires. Il comprend des suites de variables aléatoires qui ne sont pas indépendantes et sont indexées par le temps. Une chaîne de Markov est un processus stochastique qui possède plusieurs états discrets. La transition entre les états se fait

selon certaines règles probabilistes. Les chaînes de Markov ne possède pas de mémoire, ce qui implique que la prédiction du futur ne dépend que de l'état actuel du système. La Figure. 2.4 montre un exemple d'une chaine de Markov.

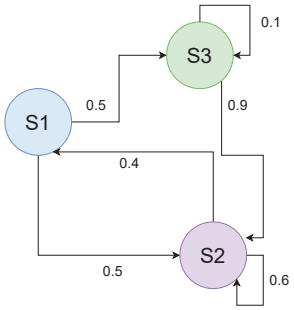


Figure 2.4: Un exemple de chaîne de Markov.

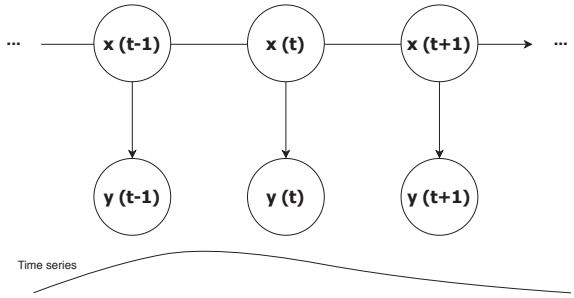


Figure 2.5: Un exemple des deux processus stochastiques d'un HMM. x est la séquence des états cachés et y est la séquence des données observées.

Dans un HMM, le processus caché est composé d'un état non observé $Q = \{q_1, q_2, ..., q_T\}$. Une matrice de transition A contrôle la transition entre ces états et contient la probabilité de rester dans le même état ou de passer à un autre. Chaque état a une distribution de probabilité sur les états observés possibles. L'état observé peut être soit discret (par exemple, généré à partir d'une distribution catégorielle), soit continu (par exemple, généré à partir d'une distribution gaussienne). HMM est composé de trois paramètres principaux :

1. π : le vecteur de distribution de l'état initial

2. A : la matrice de probabilité de transition des états cachés

3. B : les probabilités d'observations

Nous désignons HMM par $\lambda = (\pi, A, B)$.

Trois problèmes fondamentaux peuvent être résolus grâce à HMM. **Le problème de l'évaluation.** Étant donné un modèle λ et une séquence d'observation O, comment trouver la probabilité $P(O|\lambda)$ que O ait été généré par λ ? Ce problème peut être résolu en utilisant l'algorithme de forward. **Le problème du décodage.** Étant donné un modèle λ et une séquence d'observation O, comment trouver la séquence d'état optimale dans le modèle qui a produit O ? Ce problème peut être résolu en utilisant l'algorithme de Viterbi. **Le problème d'apprentissage.** Étant donné une séquence d'observation O, comment trouver les meilleurs paramètres (π, A, B), qui maximisent la probabilité $P(O|\lambda)$? Ce problème peut être résolu en utilisant l'algorithme de Baum-Welch, qui est une variante de l'algorithme bien connu "Espérance-Maximisation" (EM), adapté au modèle HMM.

2.2 Détection de patterns

La détection de pattern consiste à extraire des motifs régulier à partir de l'ensemble de données en entrée. Au cours des années précédentes, plusieurs algorithmes d'extraction de patterns séquentiels [27, 53] ont été proposés. Selon le domaine d'application, les propriétés d'un pattern peuvent changer. Par exemple, dans le domaine des achats en magasin physique, le pattern devient utiles s'il est fréquent. Il est donc utile de savoir quels sont les produits achetés souvent ensemble pour réorganiser le magasin et les mettre ensemble. Dans le domaine de la santé, un pattern devient utile s'il est temporel. Il est intéressant d'extraire et analyser les profils de patients qui ont la même séquence temporelle d'état de santé. Dans le domaine de bioinformatique, un pattern devient intéressant s'il est séquentiel. Il est utile d'étudier la séquence de protéine dans le but d'analyser l'état de santé d'un patient. Dans notre cas, nous supposons

qu'un pattern fréquent est intéressant s'il est séquentiel et temporel. Il sera utile d'étudier la séquence d'actions fréquente en prenant en considération le temps passé sur chaque action.

La **fouille de patterns séquentiels** a été proposée pour la première fois dans [2] comme un problème d'exploitation des transactions de vente des clients afin de découvrir des séquences d'achat fréquentes limitées par un support minimum spécifié par l'utilisateur. Par la suite, elle a été utilisée dans plusieurs domaines d'application : traitements médicaux [8, 84, 37], click-stream [82, 57, 66, 97, 38], appels téléphoniques [99], séquences d'ADN [88, 50], et ainsi de suite. Ces différentes applications ont motivé les chercheurs dans ce domaine à améliorer les performances (temps d'exécution, consommation de mémoire ...) des algorithmes existants et à proposer des méthodes connexes capables de répondre à des questions légèrement différentes, en mettant en évidence différents aspects des données (patterns séquentiels fermés [33, 25], patterns séquentiels maximaux [28, 31, 51] ...). Dans cette thèse, nous nous intéressons au problème de l'extraction de patterns séquentiels.

De nombreux algorithmes d'extraction de séquences [27, 53] ont été développés pour résoudre des problèmes liés à des cas d'application différents. Ces algorithmes ont le même format de données en entrée et en sortie, mais la différence réside dans leur technique d'extraction. Leur objectif principal est d'extraire tous les patterns possibles (le pattern doit exister dans la base de données d'entrée) et valides (le pattern doit respecter la contrainte de fréquence). Étant donné un ensemble de séquences et un seuil de support *minSupp* (support minimum spécifié par l'utilisateur), le but est de trouver les sous-séquences qui apparaissent edans au moins *minSupp* séquences.

Nous listons les formats de représentation des données dans la section 2.2.1 et ensuite nous détaillons les approches importantes dans la section 2.2.2.

2.2.1 La représentation des données

Plusieurs représentations sont proposées dans la littérature [27], dont nous citons les plus importantes : la représentation horizontale, la représentation verticale et la représentation

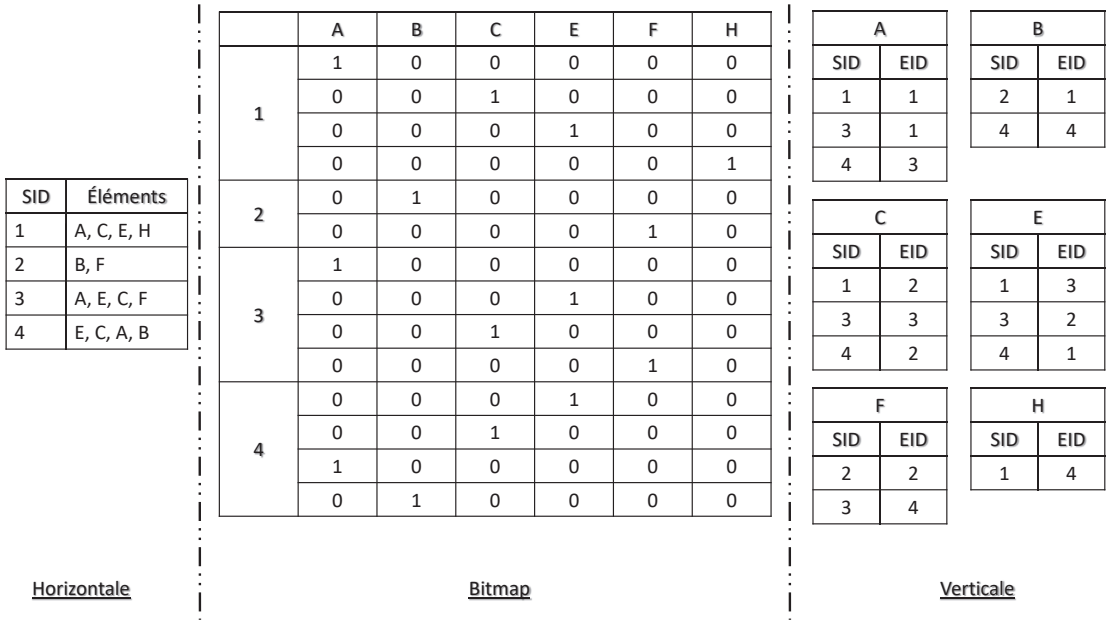
bitmap. Nous trouvons dans la Figure 2.6 un exemple de chaque représentation. Chaque approche utilise une représentation adaptée aux données pour pouvoir optimiser le plus possible les performances de leur algorithme.

Représentation Horizontale. Elle consiste à représenter les données par séquences. Chaque séquence possède un ID (entier) unique qui l'identifie et une séquence d'étiquettes des éléments qui la composent. L'apparition de la même étiquette dans plusieurs séquences est possible. Les étiquettes peuvent être des entiers, les noms des éléments, etc. Souvent, les algorithmes qui utilisent cette représentation font plusieurs parcours des données pour calculer les supports des patterns (le support d'un pattern est le nombre de séquences qui contiennent ce pattern). Ce qui peut avoir un impact direct sur le temps d'exécution de l'algorithme.

Représentation Verticale. Elle consiste à représenter les données par élément. Chaque élément est représenté par une table appelé IDList contenant son étiquette et ses positions dans les données d'entrée. Cette représentation offre deux avantages importants, le premier est le calcul rapide de la fréquence d'apparition d'un pattern. En effet, pour calculer le support d'une séquence il suffit de compter le nombre distinct des Ids des séquences dans son IDList. Ce qui est relativement rapide. Le deuxième avantage est l'élagage plus rapide de l'espace de recherche lors de la génération des candidats. Quand deux patterns se rejoignent pour générer un candidat [2], il est possible de calculer son support à partir de l'intersection de leurs ensembles d'Ids et voir où le candidat est présent dans les données avant même de le construire. Ce qui permet d'éliminer les candidats qui ne sont pas fréquents.

Représentation Bitmap. Elle consiste à représenter les données par une matrice binaire Élément-Séquence. L'ensemble de données est une matrice dont les lignes sont les éléments et les colonnes sont les séquences. Un élément du bitmap est égal à 1 si cet élément apparaît dans la séquence correspondante.

[2] Un pattern potentiel dont les contraintes de validité n'ont pas encore été vérifiées.



Horizontale

SID	Éléments
1	A, C, E, H
2	B, F
3	A, E, C, F
4	E, C, A, B

Bitmap

	A	B	C	E	F	H
1	1	0	0	0	0	0
	0	0	1	0	0	0
	0	0	0	1	0	0
	0	0	0	0	0	1
2	0	1	0	0	0	0
	0	0	0	0	1	0
3	1	0	0	0	0	0
	0	0	0	1	0	0
	0	0	1	0	0	0
	0	0	0	0	1	0
4	0	0	0	1	0	0
	0	0	1	0	0	0
	1	0	0	0	0	0
	0	1	0	0	0	0

Verticale

A	
SID	EID
1	1
3	1
4	3

B	
SID	EID
2	1
4	4

C	
SID	EID
1	2
3	3
4	2

E	
SID	EID
1	3
3	2
4	1

F	
SID	EID
2	2
3	4

H	
SID	EID
1	4

Figure 2.6: Les trois différents formats de représentations des données.

2.2.2 Les approches et les algorithmes

L'un des premiers algorithmes proposés est **GSP** [81], qui utilise la recherche en largeur (breadth-first search) pour explorer l'espace candidat et extraire les patterns niveau par niveau. Il est basé sur l'algorithme AprioriAll [2]. Il commence par scanner la base de données d'entrée pour extraire toutes les séquences fréquentes contenant un seul élément fréquent appelé 1-pattern. Puis, à l'aide de ces patterns, il génère ceux contenant deux éléments fréquents les 2-patterns, et ainsi de suite jusqu'à trouver les patterns les plus longues possibles (contenant le plus d'éléments). Les (k)-patterns trouvés au niveau k sont utilisés pour générer les (k+1)-patterns du niveau suivant (k+1).

L'algorithme GSP a deux méthodes principales : la génération des candidats et le calcul du support. A chaque étape k, GSP commence par générer tous les (k)-candidats à partir des (k-1)-patterns. Ils sont appelés candidats parce que nous ne connaissons pas encore leurs supports et donc nous ne savons pas s'ils respectent ou non la contrainte de fréquence. Ensuite,

GSP calcule pour chaque candidat son support. Ceux qui ont un support supérieur ou égal à *minSupp* deviennent des (k)-patterns et les autres sont rejetés. Ces deux méthodes sont répétées à chaque niveau jusqu'à ce qu'aucun nouveau pattern ne puisse être trouvé. Pendant le calcul du support, GSP effectue plusieurs passages sur la base de données pour compter le nombre d'apparitions de chaque candidat dans l'ensemble des séquences en vérifiant leurs apparitions dans chaque séquence. GSP est l'une des premières approches efficace pour détecter les patterns fréquents, néanmoins, elle souffre de quelques limitations. Dans GSP, le calcul du support peut prendre un long temps d'exécution (causé par ces passages multiples) et peut aussi consommer beaucoup de mémoire, car nous devons garder la base de données originale en mémoire.

Pour pallier cet inconvénient, les auteurs dans [96] ont proposé un nouvel algorithme appelé SPADE et qui utilise un format vertical des donnés. Nous détaillons les différents formats de représentation des patterns utilisés dans cette thèse dans la section 2.2.1. SPADE est capable d'utiliser la recherche en largeur (breadth-first search) ou la recherche en profondeur (depth-first search) pour extraire les patterns séquentiels. Les positions d'un pattern dans la base de données originale sont enregistrées dans une table verticale appelée IDList. Cette table contient l'identifiant des séquences et les positions du pattern dans ces séquences où il apparaît.

SPADE exécute deux méthodes principales à chaque niveau de l'exploration des patterns : la génération des candidats et le calcul du support. Les objectifs de ces méthodes sont les mêmes que ceux de GSP mais les algorithmes sont différents et plus optimisés. Pour générer un nouveau candidat, deux patterns se rejoignent s'ils ont la même séquence d'éléments sauf le dernier. Par exemple, $p_1 = \{A, B, C\}$ peut joindre $p_2 = \{A, B, D\}$ comme ils ont le même préfixe $\{A, B\}$ et cette opération de jointure produira les candidats suivants : $c_1 = \{A, B, C, D\}$ et $c_2 = \{A, B, D, C\}$. Enfin, SPADE calcule le support de chaque candidat et conserve ceux qui respectent la contrainte de fréquence. SPADE continue à répéter la génération de nouveaux candidats jusqu'à ce qu'aucun nouveau pattern ne soit trouvé. L'un des points forts de SPADE réside dans le calcul des supports des patterns. Cette opération peut être réalisée en

vérifiant simplement la table verticale du pattern. Comme nous l'avons déjà mentionné, cette table contient l'identifiant des séquences où se trouve le pattern. Par conséquent, le support est égal au nombre d'ID de séquences distinctes. Contrairement à GSP, SPADE n'a plus besoin de ré analyser la base de données originale pour calculer le support des candidats. Son point fort est donc son temps d'exécution relativement rapide et sa consommation de mémoire relativement petite. Néanmoins, SPADE souffre de la génération des faux candidats. Bien que ces faux candidats seront ensuite éliminer quand leurs supports sont calculés, mais leurs générations et le calcul de leurs support peuvent augmenter le temps d'exécution. L'architecture de l'algorithme que nous proposons par la suite est inspirée de cet algorithme.

D'autres algorithmes tels que PrefixSpan [68] et SPAM [11] ont été proposés ultérieurement. PrefixSpan est un algorithme de croissance par pattern dans lequel la base de données de séquences est projetée de manière récursive dans un ensemble de bases de données de séquences plus petites et chacune d'entre elles se développera en explorant des fragments localement fréquents. PrefixSpan commence par générer un arbre de recherche à partir d'un nœud racine vide. Chaque nœud enfant de l'arbre de recherche étend la séquence de son nœud parent avec des éléments en plus, puis PrefixSpan explore l'espace de recherche de l'arbre en utilisant la recherche en profondeur. Un ordre total sur les éléments est utilisé pour ne jamais générer de séquences en double dans l'arbre de recherche. PrefixSpan a permit d'éviter de vérifier toutes les combinaisons possibles d'une séquence de candidats potentiels, en utilisant la projection basée sur les préfixes, mais son temps d'exécution reste inférieure à celui de SPADE. L'algorithme SPAM représente les séquences en utilisant des vecteurs binaires verticaux. Il explore également l'espace de recherche selon la stratégie "depth-first" avec certains mécanismes d'élagage. Il est efficace surtout avec des séquences très longues. En utilisant les vecteurs binaires, la consommation de mémoire est largement réduite quand l'ensemble de données est denses et de nombreux bits sont généralement mis à 1. Ce qui n'est pas forcément le cas, lorsque l'ensemble de données n'est pas denses, et les vecteurs binaires sont creux.

Depuis quelques années, de nombreux algorithmes de détection de patterns séquentiels

temporels sont proposés. L'un d'entre eux est IEMiner [65] (Interval-based Event Miner), qui extrait des patterns séquentiels, fréquents et temporels. Ses patterns sont modélisés par des séquences d'événements reliés par des relations temporelles. Chaque événement dans les patterns découverts a une relation temporelle (définie par l'algèbre d'intervalle d'Allen [5]) avec tous les autres. Nous listons dans la Figure 2.7 les 13 relations définîtes dans l'algèbre des intervalles d'Allen.

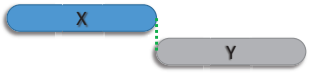
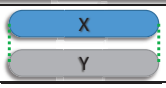
Relation	Interprétation		Illustration
Se déroule avant ou après	X < Y	Y > X	
Rencontre (meets)	XmY	YmiX	
Chevauche (overlaps)	XoY	YoiX	
Démarre (starts)	XsY	YsiX	
Se déroule pendant (during)	XdY	YdiX	
Termine (finishes)	XfY	YfiX	
Égal	X = Y		

Figure 2.7: Les 13 relations définîtes dans l'algèbre des intervalles d'Allen.

Dans l'algorithme IEMIner, deux séquences d'événements ne sont égales que si elles ont la même séquence d'étiquettes et la même séquence de relations temporelles. La durée n'est pas un facteur déterminant dans le processus d'extraction proposé. Par exemple, dans la Figure 2.8, IEMiner considère que les 4 séquences qui ont les mêmes relations temporelles sont égales alors qu'ils ont des durées différentes. IEMiner est capable de prendre en considération les relations temporelles entre les évènementss, mais n'est pas capable de faire la différence entre deux patterns temporels qui ont la même relation temporelle entre leurs évènements et que

ces derniers ont des durées différentes.

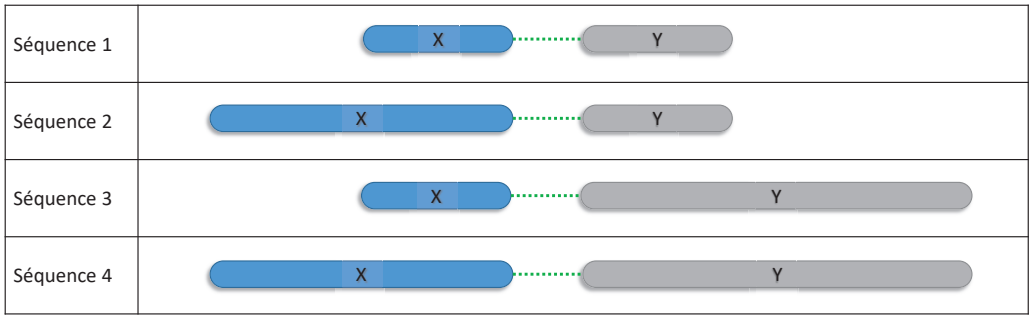


Figure 2.8: 4 séquences considérées comme égales avec l'algorithme IEMiner [65]. La durée de l'évènement est représentée par sa longueur dans cet exemple.

Nous pensons que la séquence des durées est également importante pour représenter un pattern, car l'information temporelle enrichi le pattern (en ajoutant la durée comme nouvelle dimension dans la représentation du pattern) et permet de détecter les patterns les plus discriminants. Pour cela, dans notre approche proposée dans le chapitre 3, nous prenons en considération la durée ainsi que sa variation dans le pattern.

2.3 Clustering des séries temporelles

Au cours de la dernière décennie, une quantité considérable de données de séries temporelles a été générée dans plusieurs domaines, tels que le traitement du signal, la finance, la santé, l'internet des objets, etc. Une série temporelle est une séquence de variables unies ou multi variées ordonnées dans le temps, où le temps n'est pas utilisé comme une caractéristique, mais plutôt comme un axe. Les variables collectées prennent différentes formes. Elles peuvent être discrètes si elles sont générées à partir d'une distribution catégorielle (par exemple, l'état de santé d'un patient : sain, malade), ou continues si elles changent de valeur dans un intervalle donné (par exemple, la température d'une ville). L'analyse de ces données se concentre sur

leur évolution dans le temps, et tente de répondre à plusieurs questions : comment évoluent-elles ? Sont-elles périodiques ? Y a-t-il des modèles cachés ? Comment prédire les prochaines valeurs ?

Ces données dynamiques ont motivé les chercheurs à les exploiter et les analyser pour en extraire plus de connaissances. Ainsi, plusieurs techniques d'apprentissage (machine learning) sont proposées pour extraire de la connaissance. Nous citons par exemple, la détection d'anomalies [14], le clustering [43], la prédiction [90], l'analyse [6], etc. Dans cette thèse, nous nous intéressons au problème de clustering. Notre objectif est de regrouper les comportements similaires de clients de sites marchands afin de les analyser et ensuite extraire des comportements intéressants.

2.3.1 Les approches et les algorithmes

Le clustering des séries temporelles n'est pas une tâche facile. Plusieurs défis doivent être surmontés durant le clustering. Les séries peuvent être de longueur différente, peuvent contenir des valeurs manquantes, bruitées, aberrantes, etc. Aussi la mesure de similarité entre deux séries n'est pas évidente, deux séries peuvent se différencier en amplitude, en longueur, en décalage temporel, etc.

En effet, plusieurs paramètres importants entrent en jeu quand il faut choisir ou proposer un algorithme de clustering : (1) la nature des données : les séries sont-elles courtes ou longues ? Catégorielles ou continues ? (2) les valeurs des données : contiennent-elles beaucoup de valeurs bruitées ? Aberrantes ? (3) la similarité entre deux séries : quelles métriques faut-il utiliser pour calculer la distance entre deux séries temporelles similaires ? (4) la représentation des séries temporelles : quel est le modèle le plus efficace à utiliser pour représenter les séries et les parcourir ?

Selon [49], les approches de clustering des séries temporelles peuvent être regroupées dans trois catégories :

1. des approches basées sur les données brutes : le clustering se fait directement sur les

données

2. des approches basées sur les caractéristiques : le clustering se fait indirectement, à travers les caractéristiques des données

3. des approches basées sur un modèle : le clustering se fait indirectement, à travers des modèles construits à partir des séries

Nous détaillons ces trois catégories dans les sections suivantes.

2.3.1.1 Les approches basées sur les données brutes

Ces algorithmes utilisent directement les données brutes et sont souvent composés de trois méthodes principales : (1) la mesure de similarité, (2) le calcul du représentant du cluster, et (3) la répartition des séries dans les clusters.

La mesure de similarité consiste à calculer la distance entre deux séries temporelles. Les différentes métriques et algorithmes de mesure de similarité sont détaillés dans la section 2.3.3. La plupart de ces approches considèrent que la mesure de similarité est plus importante que l'algorithme lui-même. Pour cela la plupart des travaux dans cette catégorie se sont concentrés sur les mesures de similarité plus que les algorithmes.

Le calcul du représentant du cluster permet de retrouver la série qui peut le plus représenter son cluster. Deux types de représentants du cluster existent : le centre (aussi appelé barycentre) et le médoïde. Le centre d'un cluster n'est pas forcément un point/objet appartenant aux données d'entrées. Ce qui n'est pas le cas d'un médoïde qui doit forcément l'être. La Figure 2.9 montre la différence entre le centre et le médoïde.

Finalement, la **répartition des séries** dans les clusters qui est le cœur de l'algorithme de clustering et utilise la mesure de similarité et le représentant des clusters pour affecter les séries aux clusters.

Ces approches se basent souvent sur des algorithmes de clustering qui existent déjà comme K-Means, Hierarchical Clustering, Partitionning Clustering, Density-Based Clustering ou Pat-

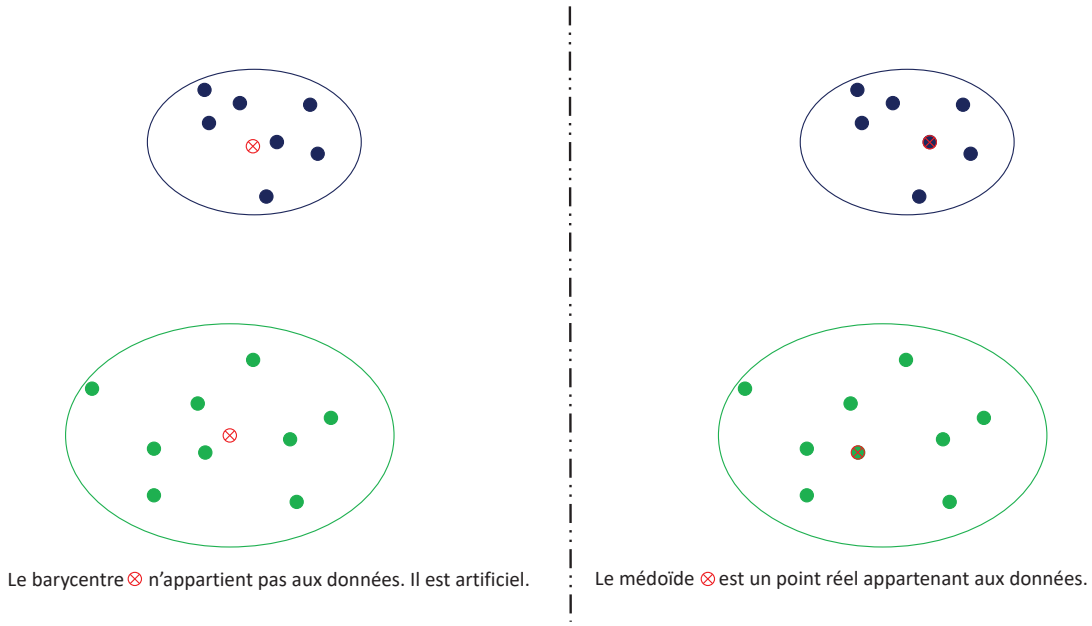


Figure 2.9: La différence entre le centre (ou barycentre) et le médoïde.

tern discovery, et remplacent leurs mesures de similarité par ceux conçus pour les séries temporelles. Malgré leur grande utilisation, ces approches souffrent de quelques faiblesses. Elles ne sont pas capables de traiter une grande quantité de données, puisqu'elles utilisent des mesures de similarité qui ont un coût de calcul élevé et aussi elles sont sensibles aux données bruitées ou aux valeurs aberrantes.

Dans [35], les auteurs appliquent des filtres non linéaires d'alignement et de moyennage. Ils proposent une méthode pairwise où chaque point de la séquence de moyenne est calculé comme un centre du mappage produit par DTW. Cette méthode est appliquée à chaque paire jusqu'à ce qu'une seule séquence reste.

La DTW (Dynamic Time Warping) est l'une des mesures les plus utilisées de la similarité entre deux séries temporelles. Elle ne compare pas les points des deux séries point-à-point mais plutôt point-à-plusieurs ou point-à-aucun. La DTW cherche l'appariement le plus optimale entre les deux séries tout en ignorant le décalage et la vitesse entre eux. La DTW est

utilisée dans notre proposition et sera plus détaillé dans la section 2.3.3.1.

Une méthode hiérarchique pour moyenner les séquences est proposée dans [63]. Les coordonnées d'une séquence moyenne sont calculées comme le centre pondéré des coordonnées de deux séquences de séries temporelles qui ont été couplées par DTW. Initialement, toutes les séquences ont un poids égal à 1, et chaque séquence moyenne produite aura un poids qui correspond au nombre de séquence qu'elle moyenne.

Dans [69], les auteurs adaptent l'algorithme de clustering K-Means pour pouvoir l'utiliser avec DTW, en proposant une nouvelle méthode pour calculer le représentant du cluster. Ils proposent Dynamic Time Warping Barycenter Averaging (DBA) qui affine itérativement les coordonnées d'une série initialement choisie parmi les données. Chaque coordonnée de la série moyenne est mise à jour en utilisant le barycentre d'une ou plusieurs coordonnées des autres séries qui ont été associées à l'utilisation de DTW. Toutefois, sa précision dépend de la longueur de la série moyenne initiale et des valeurs de ses coordonnées.

Les auteurs de [92] proposent une technique de moyennage basée sur la décomposition matricielle dans le cadre du K-Spectral Centroid Clustering (KSC).

Dans [64], les auteurs proposent deux nouveaux algorithmes k-shape et k-MS, basés sur une procédure de raffinement itératif et évolutif similaire à celle de K-Means mais avec une mesure de similarité et une méthode de calcul du médoïde différentes. Ils adaptent la mesure de corrélation croisée pour mesurer la similarité et proposent deux méthodes pour calculer les médoïdes : ShapeExtension (SE) et MultiShapes Extraction (MSE). K-Shape est basé sur SE pour calculer un seul médoïde par cluster, en utilisant les séries de ce cluster. K-MS est basé sur MSE pour calculer plusieurs centres par cluster à fin de prendre en considération la proximité et la distribution spatiale des séries temporelles de chaque cluster.

Un nouveau modèle de clustering en trois phases est proposé dans[1]. La première phase consiste à pré-clusteriser approximativement les séries temporelles. Dans cette phase, le clustering se fait sur une résolution faible des données et non pas sur les données. Une approximation des données est utilisée, suivie par un algorithme de clustering (basée sur le K-Means)

pour trouver un regroupement approximative. Ensuite, dans la deuxième phase, les clusters sont divisés en quelques sous-clusters purs. Dans cette phase, les clusters trouvés sont purifiés en utilisant les données originales. Les auteurs proposent un algorithme qui, à partir d'une matrice de distance entre les séries d'un cluster, décompose les pré-clusters en sous-clusters. L'algorithme divise les pré-clusters dont les membres ne sont pas dispersés solidement, en plusieurs sous-clusters. Enfin, dans la troisième phase, des sous-clusters sont fusionnés. Les distances entre les sous-clusters sont calculées pour former les clusters finaux. La distance entre deux sous-clusters est calculée en utilisant la DTW, et les sous-clusters qui sont les plus proches sont fusionnés.

2.3.1.2 Les approches basées sur les caractéristiques

Pour éviter de travailler directement sur les données brutes qui peuvent contenir des données bruitées ou valeurs aberrantes, des approches sont proposées pour travailler plutôt sur les caractéristiques des séries. Les méthodes d'extraction des caractéristiques sont souvent génériques, mais les caractéristiques résultantes dépendent de chaque cas d'utilisation. Certaines études appliquent une sélection de caractéristiques après leurs extractions des séries, pour en garder que les essentielles et réduire leurs nombres si possible.

Dans [87], les auteurs proposent une approche pour améliorer le clustering incrémental en utilisant la transformée en ondelettes de Haar. Tout d'abord, la décomposition en ondelettes de Haar est calculée pour toutes les séries temporelles. Ensuite l'algorithme K-Means est appliqué en commençant par le niveau grossier et en progressant vers des niveaux plus fins. Les centres finaux à la fin de chaque résolution sont réutilisés comme centres initiaux pour le niveau de résolution suivant. Une telle approche est résistante au bruit, et capable aussi de filtrer certaines informations non pertinentes et réduit par la suite la dimension des données, ce qui permet d'améliorer la qualité du clustering.

Une nouvelle approche est proposée dans [70] et est capable de regrouper les séries en utilisant seulement l'ensemble des dérivées des caractéristiques statistiques. Cette méthode est

moins sensible aux données manquantes, et peut traiter des séries de différentes tailles. Les auteurs proposent d'abord d'extraire les caractéristiques des données qui leurs semblent les plus importantes. Ensuite, ils appliquent l'algorithme K-means à l'ensemble des vecteurs de caractéristiques.

Les auteurs dans [52] ont proposé un nouveau modèle d'apprentissage de la représentation temporelle non supervisée, appelée Deep Temporal Clustering Representation (DTCR), qui intègre la reconstruction temporelle et K-means dans le modèle seq2seq. Les modèles Seq2Seq (Sequence-To-Sequence) sont des modèles d'apprentissage profond contenant deux composants principaux : un encodeur et un décodeur. Ils apprennent, et par la suite peuvent transformer, des séquences d'un domaine (par exemple, des phrase en français) en séquences d'un autre domaine (par exemple, les mêmes phrases traduites en anglais).

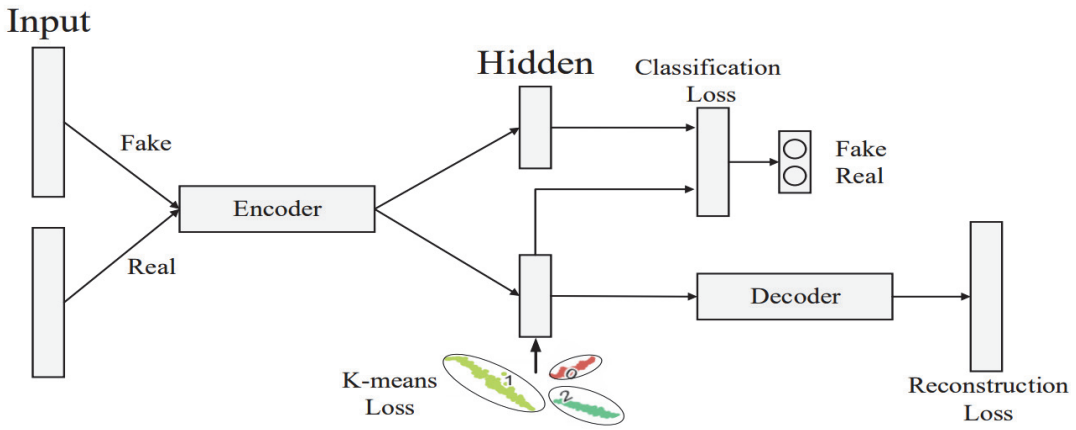


Figure 2.10: L'architecture générale de Deep Temporal Clustering Representation (DTCR)

L'architecture générale du DTCR est affichée dans la Figure 2.10. L'encodeur mappe les séries temporelles originales dans un espace latent de représentations. Ensuite, ces représentations sont utilisées pour reconstruire les données d'entrée avec le décodeur. En même temps, K-means est intégré au modèle pour guider l'apprentissage des représentations. Les auteurs proposent en plus, une stratégie de génération de faux échantillons et une tâche de classifica-

tion auxiliaire pour améliorer la capacité du codeur.

Deep Temporal Clustering (DTC) est proposé dans [54]. Il utilise un auto-encodeur [3] et une couche de clustering pour apprendre une représentation non linéaire des clusters. La couche de clustering est conçue en mesurant la divergence de Kullback-Leibler (KL) entre la distribution prévue et la distribution cible. Lors de l'apprentissage, la distribution cible est calculée par la distribution prévue et mise à jour à chaque itération. De plus, la performance du DTC dépend fortement de la capacité de l'encodeur puisque la distribution prédite est calculée sur les représentations apprises. L'architecture de DTC est présentée dans la Figure 2.11.

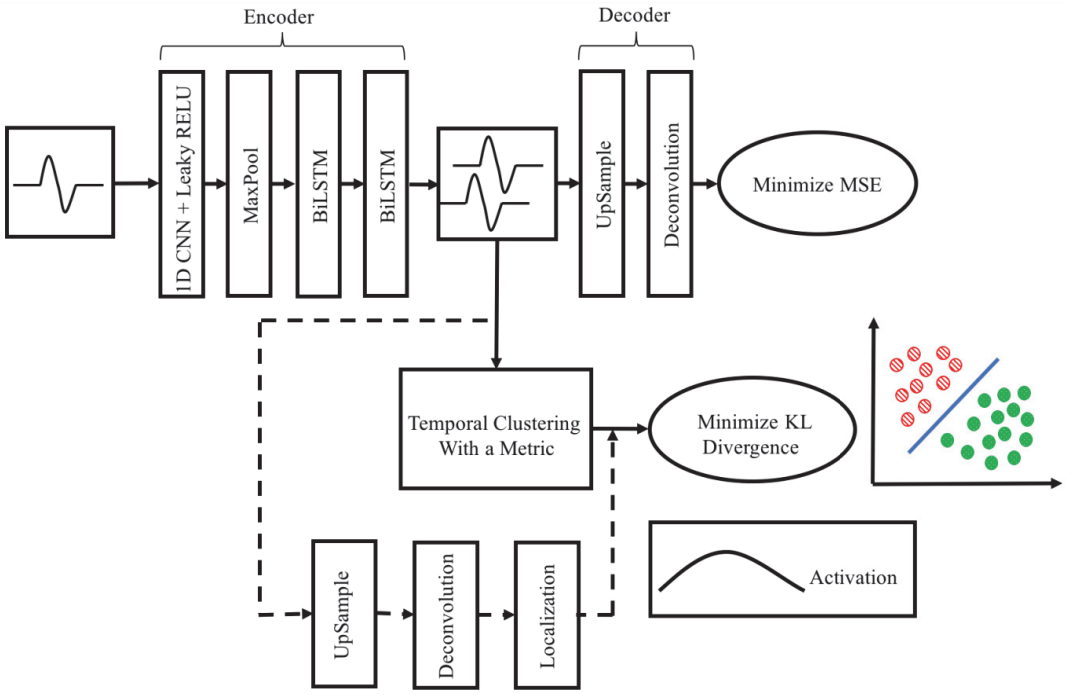


Figure 2.11: L'architecture générale de Deep Temporal Clustering (DTC)

Dans [95], les auteurs proposent un algorithme qui extrait dans un premier temps, les sous-séquences appelées « shapelets », qui sont des patterns locaux dans une série temporelle et

[3] Les auto-encodeurs sont des cas particuliers de modèles d'encodeurs-décodeurs dans lesquels l'entrée et la sortie sont les mêmes

qui sont hautement prédictives d'un groupe. Une shapelet S est un petit modèle dans une série temporelle T pour lequel la distance entre S et une partie de la série temporelle est beaucoup plus petite que la distance entre S et le reste de la série temporelle. La sélection de ces shapelets permet de séparer et supprimer une sous-séquence de la série temporelle du reste de l'ensemble de données. Ensuite, tout algorithme de clustering tel que K-Means peut être utilisé pour regrouper le reste des séries temporelles.

De plus, dans cette catégories, nous retrouvons des approches [79, 78, 44, 75, 17] qui se basent sur l'analyse en composantes principales PCA [4]. La CPA est utilisée pour la réduction de la dimensionnalité et la représentation des caractéristiques des données complexes. Son rôle est d'obtenir les K premiers composantes principales (ensemble de variables dé-corrélées les unes des autres), pour représenter les données originales.

Les auteurs dans [46] proposent MC_2 PCA une approche qui utilise la CPAC [5] pour réduire la dimensionnalité, et prend en considération les valeurs et les relations entre les variables. La CPCA [48, 45] étend l'idée de PCA pour pouvoir l'appliquer à plusieurs groupes. Contrairement aux autres approche de clustering basées sur la PCA qui examinent uniquement la relation entre les variables, MC_2 PCA examine la relation entre les différentes variables et aussi la distribution des valeurs initiales. La relation peut être reflétée par la matrice de covariance par les axes de projection communs. En outre, l'erreur de reconstruction basée sur les axes de projection communs est utilisée pour évaluer la qualité de l'affectation des séries temporelles au groupe correspondant.

2.3.1.3 Les approches basées sur un modèle

Ces approches consistent à trouver le modèle qui génère les séries de chaque cluster. Il considère que deux séries sont différentes si elles sont générées à partir de deux modèles différents. Les algorithmes regroupent ensemble les séries temporelles qui ont une probabilité

[4] Principal Component Analysis
[5] Common Principal Component Analysis

d'être générées par le même modèle. Le modèle HMM (Hidden Markov Model) est le modèle le plus utilisé par ces approches.

Hidden Markov Models (HMM)

Dans le clustering basé sur HMM, on suppose qu'une séquence d'observation O est générée à partir d'une distribution de mélange de M composants.

Dans [80], l'auteur cherche à trouver un modèle descriptif des données et non pas un modèle prédictif. Le modèle proposé est un modèle de mélange de HMMs :

$$f_K(S) = \sum_{j=1}^{K} f_j\left(S|\theta_j\right) p_j \tag{2.1}$$

où S est une séquence, p_j est le poids du $j^{\text{ième}}$ modèle, et $f_j\left(S|\theta_j\right)$ est la fonction de densité pour les données de S étant donné le modèle f_j avec les paramètres θ_j.

Ce mélange de HMMs peut être lui-même vu comme un seul HMM à plusieurs composants où la matrice de transition est bloc-diagonale. Les données sont donc générées à partir d'un seul composant HMM, et la transition entre les composants est interdite. Le chevauchement n'est pas possible entre les clusters. Pour estimer les paramètres de ce composant HMM, la maximisation du Likelihood est fixée comme objectif en utilisant l'algorithme de Baum-Welch (qui est une extension de l'algorithme EM (Expectation-Maximization) dédié aux modèles HMMs).

Comme l'algorithme EM fait beaucoup de saut dans l'espace de Likelihood, la solution finale va dépendre du point de départ. Pour cela une étape d'initialisation est importante:

Initialisation. Supposons qu'on dispose de N séquences et qu'on cherche K cluster (HMM) avec m états cachés :

1. Un HMM de m-state est affecté à chaque séquence. La matrice de transition peut être initialisée uniformément, et les vecteurs de covariances peuvent être initialisés en utilisant K-Means avec m cluster

2. La matrice de log-likelihood N*N est construite. Elle contient le log-likelihood de chaque série par rapport à chaque modèle

3. Cette matrice est utilisée pour regrouper les séries en K groups en utilisant K-Means

4. Un HMM est assigné à chaque groupe de séries trouvé. Ensuite, un modèle HMM est construit pour l'ensemble des modèles HMMs trouvés :

   (a) Sa matrice de transition A est initialisée comme une matrice bloc-diagonale. Elle est une matrice mk*mk de blocs où Aj est m*m

   (b) Le vecteur de probabilité des états initiaux P est calculé où $P_j = N_j/N$ avec N_j le nombre de séquence dans le cluster j

Apprentissage. Après l'initialisation, l'apprentissage se fait directement sur le HMM général avec la matrice A en utilisant l'algorithme de Baum-Welch et en utilisant toutes les séquences.

La faiblesse de cette approche se réside dans l'initialisation qui consiste à créer un HMM par série. Alors que le modèle HMM a besoin d'une quantité importante de données pour connaître ses paramètres. Aussi, si la série contient des bruits, le modèle va apprendre ce bruit.

Dans [7], les auteurs proposent une nouvelle approche pour découvrir les groupes d'objets similaires dans une collection de vidéos. Il s'agit d'une approche soft-clustering où le chevauchement entre les clusters est pris en considération. L'objectif est de maximiser la fonction likelihood :

$$\mathcal{L}(\theta) = \prod_{l=1}^{L} \sum_{m=1}^{M} p^{(m)} P\left(O^{(l)} | \lambda^{(m)}\right) \tag{2.2}$$

où L est le nombre de séries, M le nombre de cluster et $p(m)$ la probabilité a priori que les séries temporelles se trouvent dans le $k^{\text{ième}}$ cluster.

L'approche EM est utilisée pour maximiser le likelihood, avec :

L'étape Espérance (E-step) :

$$w_{lm} = \frac{p^{(m)} P\left(O^{(l)} | \lambda^{(m)}\right)}{\sum_{m=1}^{M} p^{(m)} P\left(O^{(l)} | \lambda(m)\right)} \tag{2.3}$$

où w_{lm} est la la probabilité postérieure qu'un $O(l)$ observé appartienne à un cluster k.

L'étape Maximisation (M-Step) :

$$\bar{p}^{(m)} = \frac{\sum_{l=1}^{L} w_{lm}}{\sum_{m=1}^{M} \sum_{l=1}^{L} w_{lm}} = \frac{\sum_{l=1}^{L} w_{lm}}{L} \tag{2.4}$$

Dans [36], les auteurs proposent une nouvelle approche utilisant une combinaison entre HMM et DTW (pour mesurer la similarité entre les clusters). L'approche proposée est « top-down ». Elle commence d'abord par la taille minimale, tant pour le modèle que pour la partition, et elle les incrémente selon une procédure de type EM jusqu'à ce qu'une certaine mesure de qualité soit atteinte. Il commence par un seul grand cluster, et ensuite augmente progressivement le nombre de cluster jusqu'à ce que la mesure BIC atteigne un maximum. Pour mesurer la qualité du modèle la mesure BIC (Bayesian Information Criterion) est utilisée.

Le likelihood des séquences a tendance à être plus faible pour les séquences plus longues, dues à sa nature cumulative. Pour corriger ce biais , les auteurs normalisent la mesure BIC en divisant le premier terme par la longueur de la séquence, et en ajoutant un facteur de régularisation α (à peu près l'inverse de la longueur moyenne de la séquence) au terme de pénalité, ce qui donne une mesure BIC normalisée.

$$\text{BIC}\left(X_k, \lambda_k\right) = \sum_{j=1}^{N_k} \frac{\log P\left(x_{kj} | \lambda_k, \hat{\theta}_k\right)}{\left|x_{kj}\right|} - \alpha \times \frac{d_k}{2} \log N_k \tag{2.5}$$

où x_{kj} est la $j^{\text{ième}}$ séquence dans le cluster X_k, $|x_{kj}|$ sa taille, et $P\left(x_{kj} | \lambda_k, \hat{\theta}_k\right)$ est le likelihood de la séquence.

Les auteurs dans [32] proposent une méthode pour regrouper les séries temporelles multi-

variées contenant des variables continues et catégorielles appliquées au domaine de la santé. L'objectif est de modéliser une série (continue et/ou catégorielle) par un HMM et ensuite regrouper les modèles au lieu de regrouper les séries selon les étapes suivantes :

1. Chaque série est modélisée par un HMM et nous notons $P_{\lambda_i} = P(T|\lambda_i)$ la probabilité que la série T soit générée par le modèle λ_i

2. $D\left(P_{\lambda_i}; P_{\lambda_j}\right)$ est définie comme la distance entre les densités de probabilité P_{λ_i} et P_{λ_j}. La distance utilisée est une adaptation de la divergence symétrique de Kullback-Leibler (KL), avec
$$D_{ij} \equiv D\left(T_i; T_j\right) \equiv \frac{1}{2}\left(D_{KL}\left(\lambda_i; \lambda_j\right) + D_{KL}\left(\lambda_j; \lambda_i\right)\right) \tag{2.6}$$

$$D_{KL}\left(\lambda_i; \lambda_j\right) \equiv D_{KL}\left(\tilde{P}_{\lambda_i}; \tilde{P}_{\lambda_j}\right) = \sum_{i=1}^{N} \tilde{P}(T_i|\lambda_i) \log \frac{\tilde{P}(T_i|\lambda_i)}{\tilde{P}\left(T_i|\lambda_j\right)} \tag{2.7}$$

$$Z_\lambda = \sum_{i=1}^{N} P(T_i|\lambda)) \text{ et } \tilde{P}(T_i|\lambda) \equiv Z^{-1} P(T_i|\lambda) \tag{2.8}$$

3. Une fois la matrice $D_{ij} \equiv D\left(T_i; T_j\right)$ est construite, le clustering est fait en appliquant n'importe quel algorithme de clustering qui prend en entrée une matrice de distance. Les auteurs ont utilisé l'algorithme PAM (Partition Around Medoids)

Dans [93], les auteurs proposent une méthode utilisant la DTW le HMM appliqués pour étudier le comportement des conducteurs. La DTW et le clustering hiérarchique sont utilisés pour obtenir une première version des clusters. Ensuite chaque cluster sera modélisé par un HMM. Une méthode d'optimisation itérative est utilisée pour affiner les résultats. Elle consiste à calculer pour chaque série les probabilités qu'elle soit générée par les autres modèles HMMs. Ensuite, la série rejoint le modèle ayant la probabilité la plus élevée et les paramètres du modèle HMM du cluster seront ré-estimés. Cette procédure est répétée jusqu'à la convergence quand aucune série ne quitte son cluster pendant 50 itérations.

Un framework de clustering des données temporelles est proposé dans [10]. En premier, le framework un HMM est entraîné par série temporelle. Ensuite les séries temporelles sont réparties aléatoirement dans les clusters. Pour chaque cluster, un HMM est entraîné à partir des séries temporelles qu'il contient. La distance entre les séries et les HMM est ensuite calculée. Finalement chaque série rejoint le cluster le plus proche d'elle. L'entraînement des clusters et la répartition des séries dans les clusters sont répétés jusqu'à la convergence.

2.3.2 Classification ensembliste

La classification ensembliste est une branche de l'apprentissage ensembliste qui consiste à utiliser plusieurs algorithmes d'apprentissage pour obtenir de meilleures prédictions. Nous citons parmi ces algorithmes : le bagging [15] et le boosting [77].

Le bagging. Appelé aussi "bootstrap aggregating", le bagging est conçu pour améliorer la stabilité et la précision des algorithmes d'apprentissage automatique. Il est surtout utile pour éviter le surapprentissage. Il consiste à extraire plusieurs échantillons de données à partir de l'ensemble original, et ensuite entraîner un modèle par échantillon. L'entraînement peut se faire donc en parallèle et le tirage des données de l'échantillon est aléatoire et avec remise. Selon le domaine d'application, le résultat final peut être la moyenne de la sortie (pour la régression) ou peut être obtenu par vote majoritaire (pour la classification). L'exemple d'algorithme le plus connu est le Random Forest [16].

Le boosting. Il consiste à améliorer les compétences d'un faible classifieur. Il entraîne plusieurs modèles (relativement faible) l'un après l'autre. Chaque modèle va essayer de corriger les erreurs de son prédécesseur. De cette manière, les faiblesses d'un modèle sont compensées par les forces de l'autre. Le boosting concentre donc ses efforts sur les objets difficiles à ajuster. L'entraînement avec boosting se fait consécutivement et d'une façon complémentaire. Les deux exemples d'algorithmes les plus connus sont AdaBoost [71] et Gradient Boosting [29].

2.3.3 Les mesures de similarité

Les mesures de similarité sont des fonctions qui consistent à calculer la distance (exacte) ou à quantifier la similarité (approximative) entre deux objets (points, images, séries temporelles, etc.). Dans le cas des séries temporelles, ces mesures doivent surmonter plusieurs difficultés liées à la nature complexe de ces séries. Souvent, les séries temporelles contiennent des données bruitées ou des valeurs aberrantes. En plus, contrairement aux données statiques, les séries temporelles peuvent se différencier par leurs amplitudes, longueurs, discontinuités, décalages, etc.

Plusieurs mesures de similarité sont proposées pour surmonter ces difficultés, et pour prendre en compte des cas spécifiques de séries temporelles. Le choix de la mesure de similarité doit être par rapport à la nature des données (ses caractéristiques, longueur, la méthode de représentation, etc.), d'où l'utilité de dresser un état de l'art de ces mesures pour pouvoir choisir la mesure la plus adaptée à notre besoin.

Plusieurs façons de catégoriser les mesures sont proposées.

Les auteurs dans [59] proposent de grouper les mesures dans quatre catégories (1) mesures sans modèle (2) mesures basées sur des modèles (3) mesures basées sur la complexité et (4) mesures basées sur la prédiction.

Quatre catégories de mesures sont proposées dans [89] et qui sont : (1) mesures de verrouillage (2) mesures élastiques (3) mesures basées sur des seuils et (4) mesures basées sur des patterns.

Dans [24], les auteurs proposent les quatre catégories suivantes : (1) mesures basées sur la forme (2) mesures basées sur l'édition (3) mesures basées sur les caractéristiques et (4) mesures basées sur les structures. Cette catégorisation correspond le plus à notre besoin de présentation et nous l'utilisons pour présenter les mesures de similarité les plus populaires dans le domaine des séries temporelles. Pour une meilleure organisation, nous fournissons les formules les plus importantes en supposant que nous disposant de deux séries temporelles $X = (x_0, x_1, \ldots, x_{N-1})$ et $Y = (y_0, y_1, \ldots, y_{M-1})$ si la longueur de X et Y est différente, et

$X = (x_0, x_1, \ldots, x_{T-1})$ et $Y = (y_0, y_1, \ldots, y_{T-1})$ sinon. Plus de détails sur les mesures sont disponibles dans [13].

2.3.3.1 Les mesures basées sur la forme (shape-based)

Cette catégorie regroupe les mesures qui s'appliquent directement sur les valeurs brutes des séries et qui s'intéressent à la forme des séries.

Distance type L_p

Nous retrouvons en premier les mesures de distances type L_p. Pour comparer deux séries, ces mesures appliquent une comparaison point par point sur l'axe de temps, comme le montre la Figure 2.12. La forme et le temps sont importants pour faire la différence entre deux séries. Malgré leurs utilisations très répandues, leurs facilités à comprendre et à calculer, ces mesures souffrent de quelques faiblesses. En effet, elles ne traitent que des séries de la même longueur, ce qui n'est pas souvent le cas des données réelles. En plus, elles ne prennent pas en considération les bruits, les valeurs aberrantes, le décalage temporel, etc., même si la quantité de ces derniers est parfois réduite par des prétraitements. Les différentes distances de type L_p sont présentées dans la Table 2.4.

La distance Short Time Series

Proposée par [58] pour étudier les données des puces à ADN, la distance Short Time Series (STS) est proposée pour inclure l'information temporelle dans le calcul de la similarité. Elle permet de mesurer la similarité des formes qui sont formées par le changement relatif d'amplitude et les informations temporelles correspondantes. Elle est définie par :

$$STS\left(\mathbf{X}_T, \mathbf{Y}_T\right) = \sqrt{\sum_{i=0}^{T-1} \left(\frac{y_{i+1} - y_i}{t_{i+1} - t_i} - \frac{x_{i+1} - x_i}{t'_{i+1} - t'_i} \right)} \tag{2.9}$$

[6] nommé aussi "Maximum value distance" ou "infinite norm" ou "Chebyshev"

Distance	p	Formule	Commentaire
Manhattan	1	$\sum_{i=0}^{T-1} \lvert x_i - y_i \rvert$	utile pour calculer la distance entre deux points dans un trajet en forme de grille
Euclidienne	2	$\sqrt{\sum_{i=0}^{T-1} (x_i - y_i)^2}$	calcule la distance directe entre deux points
Minkowski	$1 < p < \infty$	$\sqrt[p]{\sum_{i=0}^{T-1} \lvert x_i - y_i \rvert^p}$	elle s'agit d'une généralisation des distances de Manhattan et Euclidienne
Tchebychev [6]	∞	$\lim_{p\to\infty} \sqrt[p]{\sum_{i=0}^{T-1} \lvert x_i - y_i \rvert^p} = \max_{0\leq i\leq T-1} \lvert x_i - y_i \rvert$	défini comme le maximum des valeurs absolues des différences entre leurs coordonnées

Table 2.4: Les différentes distance type L_p

où t, t' représentent les indices temporels de X et Y, respectivement, et leur incrémentation doit être identique : $t_{i+1} - t_i = t'_{i+1} - t'_i, \forall i = 0, ..., T-1$.

Distance DTW

Pour surmonter les faiblesses des distances type L_p, des mesures élastiques sont proposées. En plus de la comparaison point-à-point entre les séries, ces mesures appliquent une comparaison point-à-plusieurs ou point-à-aucun, ce qui donne plus de souplesse temporelle au calcul. Cela permet une prise en considération des décalages, des longueurs différentes, etc. La distance élastique la plus populaire est la Dynamic Time Warping (DTW) [85].

La DTW est une mesure très populaire conçue pour mesurer la similarité entre deux séries temporelles. Son objectif est de trouver un alignement global et optimal entre deux séries temporelles. Cela revient à trouver un appariement entre les points des deux séries tout en minimisant les coûts d'associations.

Au lieu de comparer les points qui se trouvent sur le même axe temporel (comme c'est le cas avec les distances de type L_p), la DTW compare chaque point à ses voisins de l'autre série

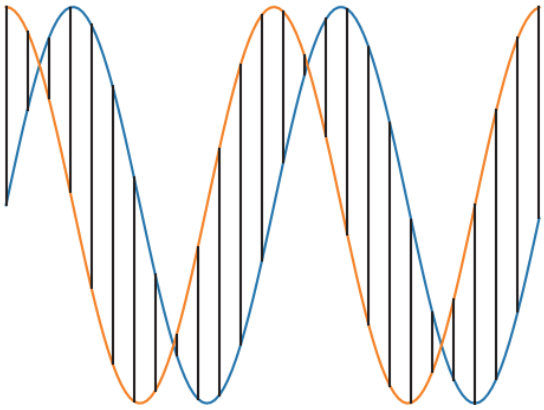
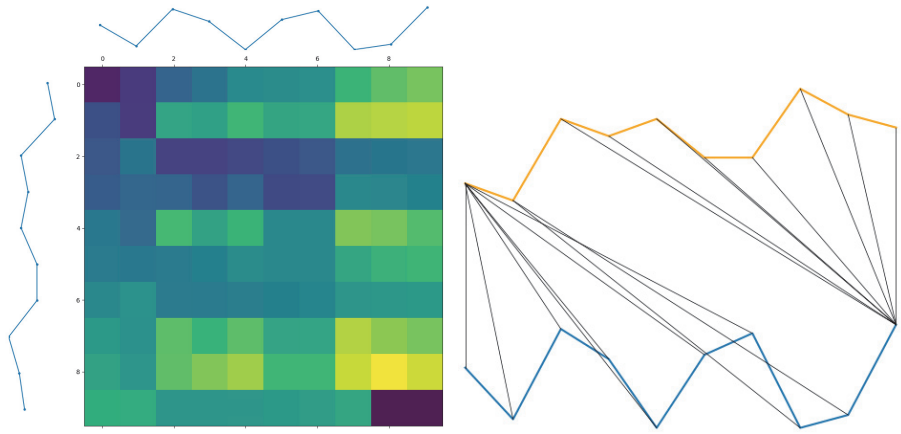


Figure 2.12: Comparaison de deux séries temporelles égales et décalées en utilisant la distance euclidienne. Les lignes noires verticales relient les paires de points comparés utilisés pour mesurer la similarité.

et choisit celui qui offre le meilleur appariement. Elle collecte ce qu'elle compare dynamiquement et cherche un appariement optimal entre les points (voir la Figure 2.13). L'appariement optimal est celui qui a le coût global le plus faible parmi tous les autres possibles.



(a) Local Cost Matrix. **(b)** Comparaison des points.

Figure 2.13: Comparaison de deux séries temporelles à l'aide de DTW.

La DTW commence par construire la matrice des distances par paires appelée "Local Cost

Matrix"[7]. Une fois cette matrice est construite, DTW fait un backtrack[8] pour trouver le chemin d'alignement optimal appelé le "Warping Path"[9]. Nous formalisons cette mesure comme suit.

Définition 1 *Étant donné deux séquences X_N avec N points et Y_M avec M points, la "Local Cost Matrix" D est une matrice N*M, où la distance $D_{i,j}$ entre le $i^{ième}$ point de X et le $j^{ième}$ point de Y est calculé récursivement :*

$$D_{i,j} = d(X_i, Y_j) + min \left\{ \begin{array}{l} D_{i-1,j} \\ D_{i-1,j-1} \\ D_{i,j-1} \end{array} \right\} \tag{2.10}$$

où d(.,.) peut être la distance euclidienne (classiquement de type L_p).

Définition 2 *Le "Warping Path" peut être chaque chemin de la matrice D reliant le coin de départ au dernier coin qui lui est opposé sur la diagonale. Le nombre de chemins possibles peut être très important et le chemin optimal doit satisfaire certaines contraintes comme :*

1. *Contrainte de frontière (Boundary-constraint) : le début et la fin du chemin doivent être dans les coins opposés de la diagonale de la matrice (pour traiter les séquences entières)*
2. *Contrainte de monotonie (Monotonicity-constraint) : force les points du chemin à être espacés de façon monotone dans le temps (on ne remonte pas dans le temps)*
3. *Contrainte de continuité (Continuity-constraint) : limite les étapes autorisées aux cellules adjacentes (pas de saut dans le temps)*

Il est à noter que d'autres contraintes [74] peuvent également être ajoutées, telles que la fenêtre de déformation, la contrainte de pente, etc.

Sauf mention du contraire, c'est cette mesure qui sera utilisée comme référence tout au long de cette thèse.

[7] "matrice des coûts locaux" en français
[8] "marche arrière" en français
[9] "le chemin d'alignement" en français

D'autres variantes et alternatives de la DTW sont proposées pour accélérer son exécution ou pour améliorer ses performances. Derivative Dynamic Time Warping (DDTW) est proposée dans [34], et est basée sur la dérivée pour obtenir une distance de similarité plus nuancée. Elle considère la forme générale d'une série temporelle plutôt que la comparaison par point. La DDTW essaie d'éviter les singularités (où un seul point peut correspondre à une grande sous-section d'une autre série temporelle) en obtenant les caractéristiques du niveau supérieur de la forme de la courbe. Time Warp Edit Distance (TWED), introduite dans [55], mesure la similarité entre deux séries temporelles en comptant le nombre de transformations à effectuer pour les faire correspondre. Dans [39], les auteurs proposent le Weighted Dynamic Time Warping (WDTW) qui est une DTW basée sur les pénalités. La WDTW pénalise les distances plus élevées entre les points et empêche une distorsion minimale de la distance causée par des valeurs aberrantes. Les auteurs de [83] proposent le Move-Split-Merge (MSM) où la similarité est calculée en utilisant un ensemble d'opérations pour transformer une série donnée en une série cible.

2.3.3.2 Les mesures basées sur l'édition (Edit-based)

Les mesures dans cette catégorie, calcule la distance entre deux séries temporelles en comptant le nombre d'opérations (ajout, suppression, remplacement) nécessaires pour transformer une série en une autre. Nous retrouvons ici la distance de Levenshtein, qui est très utilisée pour calculer la distance entre deux chaînes de caractères.

Longest Common Sub-Sequence (LCSS)

Cette mesure [86] cherche l'alignement entre deux séquences qui maximise la longueur des sous-séquences communes. Elle permet l'écart et le saut durant la comparaison de deux séries.

Elle était conçue pour traiter les valeurs aberrantes ou bruyantes.

$$LCSS(X,Y) = \begin{cases} 0 & \text{si } N-1=0 \text{ ou si } M-1=0 \\ LCSS(tail(X), tail(Y)) + 1 & \text{si } |x_0 - y_0| \leq \varepsilon \\ \max(LCSS(tail(X), Y), LCSS(X, tail(Y))) & \text{sinon.} \end{cases} \tag{2.11}$$

où ε est certain seuil pour contrôler la distance minimale de différence et *tail* est une fonction qui renvoie toute la série sauf le premier.

Edit distance pour les séquences réelles (EDR)

Cette mesure proposée par [20] autorise, comme LCSS, des sauts durant la comparaison mais elle pénalise ces sauts par une valeur égale à sa taille. Son calcule peut être converti en itération en utilisant la programmation dynamique.

$$\text{EDR}(\text{X}, \text{Y}) = \begin{cases} N & \text{si } M-1=0 \\ M & \text{si } N-1=0 \\ \Pi & \text{sinon.} \end{cases} \tag{2.12}$$

où

$$\Pi = d(x_0, y_0) + \min \begin{cases} EDR(\text{tail}(X), \text{tail}(Y)) + d_{edr}(x_0, y_0, \varepsilon) \\ EDR(\text{tail}(X), Y) + 1 \\ EDR(X, \text{tail}(Y)) + 1 \end{cases} \tag{2.13}$$

et

$$d_{erd}(x_i, y_i, \varepsilon) = \begin{cases} 0 & \text{if } |x_i - y_i| \leq \varepsilon \\ 1 & \text{sinon} \end{cases} \tag{2.14}$$

Edit distance with Real Penalty (ERP)

Cette mesure [19] est une combinaison de DTW et EDT. Comme EDT, elle autorise et pénalise les sauts mais différemment. La pénalisation s'effectue en fixant une constante g et en ajoutant à g la distance euclidienne des points non appariés.

$$ERP(X,Y) = \begin{cases} \sum_{i=0}^{N-1} |y_i - g| & \text{si } M-1=0 \\ \sum_{i=0}^{M-1} |x_i - g| & \text{si } N-1=0 \\ \kappa & \text{sinon.} \end{cases} \tag{2.15}$$

avec

$$\kappa = \min \begin{cases} ERP(\text{tail}(X), \text{tail}(Y)) + d(x_0, y_0) \\ ERP(\text{tail}(X), Y) + d(x_0, g) \\ ERP(X, \text{tail}(Y)) + d(y_0, g) \end{cases} \tag{2.16}$$

2.3.3.3 Les mesures basées sur les caractéristiques (Feature-based)

L'idée de ces mesures est de calculer la distance en se basant sur les caractéristiques des séries temporelles qui reflètent leurs aspects. La comparaison ne se fait plus directement sur les valeurs brutes, mais plutôt sur une abstraction plus élevée.

Distance basée sur le coefficient de corrélation de Pearson

La corrélation de Pearson mesure comment deux séries temporelles continues co-varient dans le temps et retourne leur relation linéaire sous forme d'un nombre entre -1 (corrélé négativement) à 0 (non corrélé) à 1 (parfaitement corrélé).

$$PC(X,Y) = \frac{\sum_{i=0}^{T-1} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=0}^{T-1} (x_i - \bar{x})^2} \sqrt{\sum_{i=0}^{T-1} (y_i - \bar{y})^2}} \tag{2.17}$$

avec $\bar{x}$ et $\bar{y}$ les moyennes des séries X et Y respectivement. Deux mesures sont basées sur

cette corrélation :

$$d_{PC_1} = \left(\frac{1 - PC}{1 + PC}\right)^{\beta} \tag{2.18}$$

$$d_{PC_2} = 2(1 - PC) \tag{2.19}$$

avec b un paramètre positif défini par l'utilisateur.

Basée sur la corrélation croisée entre deux séries

La corrélation croisée est la comparaison de deux séries temporelles différentes pour détecter s'il existe une corrélation entre des mesures ayant les mêmes valeurs minimales et maximales.

$$CC_k(X, Y) = \frac{\sum_{i=0}^{T-1-k} (x_i - \bar{x})\,(y_{i+k} - \bar{y})}{\sqrt{\sum_{i=0}^{T-1} (x_i - \bar{x})^2}\sqrt{\sum_{i=0}^{T-1-k} (y_{i+k} - \bar{y})^2}} \tag{2.20}$$

où k est un pas de décalage. Une mesure est proposée en se basant sur cette corrélation :

$$d_{\mathrm{CC}}(X, Y) = \sqrt{\frac{(1 - CC_0(X, Y))}{\sum_{k=1}^{\max} CC_k(X, Y)}} \tag{2.21}$$

2.3.3.4 Les mesures basées sur la structure (structure-based)

Comme les mesures basées sur les caractéristiques, les mesures de distance entre deux séries temporelles ne se font plus en utilisant les valeurs brutes, mais plutôt en comparant la structure des deux séries temporelles. Ces mesures sont très efficaces pour les longues séries temporelles, comme elles ne cherchent pas des similarités locales entre les patterns. La comparaison se fait à une échelle plus globale. Dans cette catégorie, nous retrouvons deux sous-catégories : la première est basée sur le modèle et la deuxième est basée sur la compression.

2.3.3.5 Les mesures basées sur le modèle (model-based)

Pour mesurer la similarité, l'idée ici est de comparer les paramètres des processus générant les séries temporelles. Pour comparer deux séries, on va trouver le modèle générant la première et ensuite calculer la probabilité que la deuxième série soit générée par le même modèle. Plusieurs modèles peuvent être utilisés (selon la nature des données).

Modèle HMM : modèle probabiliste, il généralise le modèle de Markov observable avec en plus une couche d'états cachés.

Modèle ARMA (Autoregressive Moving Average model) : Il est utilisé pour décrire des séries temporelles stochastiques faiblement stationnaires en termes de deux polynômes. Le premier de ces polynômes est pour l'autorégression, le second pour la moyenne mobile.

2.3.3.6 Les mesures basées sur la compression (compression-based)

Le calcul de similarité repose ici sur la complexité Kolmogorov. La complexité Kolmogorov K(x) d'un objet x (texte, chemin, image, chaîne de caractère ...) est la taille du plus petit programme/algorithme capable de produire x. K(x) est la quantité minimale d'information nécessaire pour générer l'objet x par un algorithme. Par conséquent, le niveau de complexité de x est lié à K(x). La complexité conditionnelle de Kolmogorov K(x|y) de x étant donné y est définie comme la longueur du plus court programme produisant x lorsque y est donnée comme entrée auxiliaire au programme.

Normalized information distance (NID)

NID(x, y), proposée par [47], prend une valeur entre [0, 1]. Comme la complexité de Kolmogorov n'est pas calculable, elle est remplacée dans NID par la taille de l'objet compressé obtenue par des compresseurs (gzip, bzip2, etc). NID est donc approximativement calculé.

$$NID(X, Y) = \frac{\max\{K(X|Y), K(Y|X)\}}{\max\{K(X), K(Y)\}} \tag{2.22}$$

Normalized compressed information (NCD)

Proposé par [47] pour pouvoir estimer plus facilement le numérateur qui implique des complexités Kolmogorov conditionnelles rendant plus difficile l'obtention de l'approximation.

K(x) – K(x|y) mesure la quantité d'information qu' y produit sur x.

$$NCD\left(X,Y\right)=\frac{C\left(XY\right)-\min\left\{C\left(X\right),C\left(Y\right)\right\}}{\max\left\{C\left(X\right),C\left(Y\right)\right\}} \tag{2.23}$$

Plus la valeur de NCD est petite, plus les séries sont proches.

2.4 Résumé

Dans ce chapitre, nous avons présenté différentes approches en relation avec la Détection de patterns fréquents et avec le Clustering des séries temporelles.

Détection de patterns. Nous avons présenté dans la section. 2.2, plusieurs approches en relation avec la détection de patterns fréquents, séquentiels, et temporels. En ce qui concerne l'aspect temporel du pattern, nous constatons que seules les relations temporelles entre les éléments du pattern sont prises en compte. Dans notre cas, ces relations n'existent pas, car il n'existe pas de chevauchement entre les actions sur le site. Par contre, une information importante est à prendre en considération, c'est la durée. Cette durée reflète la longueur de l'élément dans sa séquence (ex. dans le domaine de la santé, en étudiant les états de santé des patients, il sera intéressant d'étudier la durée de la maladie d'un patient pour savoir si elle a une influence sur son état de santé générale). Il serait donc intéressant de proposer une approche capable d'inclure cette durée/longueur dans l'extraction des patterns comme nous le verrons au chapitre 3.

Clustering des séries temporelles. Nous avons présenté dans la section. 2.3, plusieurs approches en relation avec le clustering des séries temporelles. Ces approches se divisent en trois catégories : celles qui s'appliquent directement sur les données brutes, celles qui travaillent sur les caractéristiques des données, et celles qui travaillent sur les modèles qui génèrent

les données. Parmi ces catégories, nous nous sommes intéressés par la dernière catégorie qui est capable de fournir en plus des clusters, les modèles qui ont généré les données de ces clusters. Nous parlons ici d'un clustering descriptif. Mais malgré la richesse présente dans l'état de l'art, nous avons constaté qu'il n'existe pas d'approches permettant d'inclure le Bagging pour surmonter les problèmes de démarrage bien connu et présent dans le clustering des séries temporelles. Nous nous sommes donc intéressé au clustering avec Bagging des séries temporelles comme nous le verrons au chapitre 4.

3

Modéliser un comportement fréquent

Le comportement fréquent est primordial pour la découverte des besoins du client. Son analyse peut être la première étape menant à comprendre les comportement généraux des clients. Il peut être représenté par une suite séquentielle et fréquente d'action similaire sur le même site d'e-commerce. Il est très important de valoriser ce type de comportement comme il reflète des tendances récurrentes sur le site d'e-commerce. Dans ce chapitre, nous présentons un nouveau modèle de comportement appelé "comportement fréquent". Nous proposons une nouvelle représentation de motif temporel pour modéliser ce comportement et un nouvel algorithme de détection de motifs appelé ***SEPM*** *(****S****equential* ***E****vent* ***P****attern* ***M****ining) basé sur la nouvelle représentation pour détecter les comportement fréquents. SEPM vise à extraire des patterns d'action séquentiels et fréquents à partir d'un ensemble de cliques. La particularité de SEPM est la prise en compte de la durée de chaque action durant l'extraction des patterns. Cette durée est utilisée comme un indice qui reflète l'intérêt d'un utilisateur par la page*

qu'il visualise. SEPM utilise une représentation verticale des données et des patterns, et il les exploite pour miner les patterns séquentiels fréquents. Pour une interprétation avancée des résultats, nous catégorisons les patterns résultants de SEPM et nous prouvons qu'ils sont plus discriminants que les patterns sans durée. Notre approche est testée sur des données réelles, et les résultats sont analysés pour extraire des comportements discriminants des utilisateurs. Elle est aussi testée sur des données synthétiques pour évaluer ses performances. Les résultats expérimentaux indiquent que SEPM est efficace et évolutif.

Plan du chapitre

3.1 Introduction

Les vendeurs en ligne savent que leurs clients sont tous différents, mais ils savent aussi qu'il y a des régularités dans leurs recherches. La compréhension de ces subtilités peut les aider à personnaliser leur site web en fonction des actions de leurs clients, ou à proposer des nouvelles offres commerciales.

Dans ce chapitre, notre principal objectif est de fournir aux vendeurs en ligne des techniques appropriées qui leur permettent d'acquérir des connaissances sur leurs données. Cela en concevant les outils nécessaires pour extraire les comportements fréquents et discriminants de leurs clients. Parmi ces comportements, nous trouvons par exemple :

- *le comportement des déterminés* : les clients consultent la même séquence de produits et y consacrent à peu près le même temps pour chaque produit
- *le comportement des observateurs* : les clients, avant de se concentrer sur leurs principaux produits, consultent certains accessoires connexes (par exemple, les accessoires de téléphone portable avant les téléphones portables, les cartouches d'imprimante avant les imprimantes, etc.)
- *le comportement des comparateurs* : les clients consultent plusieurs fois les mêmes produits pour comparer leurs caractéristiques
- *le comportement des chercheurs* : les clients consultent le catalogue de produits au hasard sans intention d'achat

Chaque comportement fréquent indique un parcours commun des visiteurs durant leurs parcours sur le site web. Nous le modélisons sous forme de pattern séquentiel fréquent dans les clickstreams. Un pattern séquentiel est une séquence ordonnée d'actions (clics) qui conduisent à une action spécifique (un achat ou un abandon). L'analyse des patterns séquentiels peut nous aider à déduire des règles d'actions par exemple :

Si un client **achète** une *imprimante*

↓

il sera probablement intéressé par **l'achat** des *cartouches d'encre* juste après

Au cours des années précédentes, plusieurs algorithmes d'extraction de patterns séquentiels [27, 53] ont été proposés. Ils sont conçus pour extraire des patterns séquentiels de produits en utilisant leurs étiquettes (nom ou url du produit). Cependant, les clickstreams contiennent une information très importante sur le parcours d'un visiteur. C'est la **durée** de temps consacrée à la consultation de chaque produit. Cette durée représente le temps passé par un utilisateur sur une page, et elle peut être utilisée comme un facteur pour mesurer l'intérêt d'un utilisateur par le contenu de la page. Un utilisateur est plus susceptible d'être intéressé par un produit qu'il a consulté pendant 5 minutes qu'un autre produit qu'il a consulté pendant 1 minute. Sans cette durée, nous ne faisons aucune différence entre les pages consultés rapidement (non intéressante pour l'utilisateur) et les autres pages auxquelles le visiteur consacre plus de temps à ces produits, ce qui peut entraîner une mauvaise interprétation des patterns.

Nous décidons donc de prendre en considération cette durée, et de l'intégrer dans le processus d'extraction des patterns, afin de rendre les patterns extraits plus significatifs. Deux aspects sont donc pris en considération : l'ordre des produits consultés et le temps passé par les visiteurs sur ces produits. L'ordre est utilisé pour conserver le caractère séquentiel des données, et la durée pour rendre les patterns plus significatifs. L'objectif est de trouver la **séquence** des produits consultés **fréquemment** avec la **durée moyenne** du temps passé sur chaque produit.

3.2 Motivations

Comme décrit précédemment, notre objectif est d'extraire et d'analyser les comportements, discriminants et fréquents, des utilisateurs sur les sites de commerce électronique. Nous modélisons ces comportements par des patterns **temporels**, **séquentiels** et **fréquents**. Ces trois

propriétés nous aident à extraire les patterns les plus discriminants.

Un pattern est temporel. Le temps de consultation des produits est crucial, il peut indiquer si le visiteur est susceptible d'être intéressé par le produit ou non. Pour cela, nous proposons d'inclure cette durée dans la représentation des données et le prendre en considération lors de l'extraction des patterns.

Un pattern est séquentiel. L'ordre de consultation d'un catalogue de produits est très important dans l'analyse du comportement du visiteur. Cet ordre peut nous donner des indices sur les besoins cherchés. Nous pouvons savoir, quand le visiteur a un objectif précis (si tous les produits visités sont de la même catégorie) et quand le visiteur parcourt aléatoirement les catalogues sans objectif à l'avance.

Un pattern est fréquent. Le nombre d'apparitions du pattern est aussi important. Il indique si un comportement est souvent présent et doit être étudié ou s'il est rare et on peut l'ignorer.

3.3 Reformulation du problème

Pour modéliser un pattern fréquent, nous commençons d'abord par définir ses éléments.

Définition 3 ***(évènement).*** *Un évènement* ***E*** *est représenté par un couple* ***étiquette*** *et* ***durée****, où l'étiquette est le nom du produit et la durée est le temps écoulé depuis le début de l'évènement et jusqu'à sa fin (ce qui correspond au temps passé sur la page du produit).*

Dans notre étude, l'étiquette de l'évènement est une chaîne de caractère contenant le nom du produit, mais elle peut bien être remplacée par d'autre information comme : l'url, la catégorie, les prix, etc. Nous ignorons l'heure de début et l'heure de fin de l'événement, considérées comme moins instructives que la durée dans notre cas. Dans ce travail, nous ne ferons pas de différence entre un utilisateur qui se connecte le matin ou un utilisateur qui se connecte le soir. Nous souhaitons le prendre en compte dans de futurs travaux.

SID	Events (Label, Duration)	Event list ($\text{Label}_{\text{eid}}$)
1	E_1 (A, 3), E_2 (B, 6) E_3 (C, 7), E_4 (E, 9)	$EL_1 = A_1 \rightarrow B_2 \rightarrow C_3 \rightarrow E_4$
2	E_1 (A, 5), E_2 (B, 8) E_3 (C, 9)	$EL_2 = A_1 \rightarrow B_2 \rightarrow C_3$
3	E_1 (A, 3), E_2 (B, 4) E_3 (D, 1), E_4 (E, 3)	$EL_3 = A_1 \rightarrow B_2 \rightarrow D_3 \rightarrow E_4$
4	E_1 (A, 7), E_2 (B, 8) E_3 (D, 3)	$EL_4 = A_1 \rightarrow B_2 \rightarrow D_3$

Table 3.1: Exemple de base de données de séquence d'événements et de patterns détectés.

Définition 4 ***(liste d'évènements).*** *Une liste d'évènements séquentiels* $\boldsymbol{SE} = \{E_1, E_2, ..., E_n\}$ *est une liste d'évènement trié par les heures de début de ses événements. La longueur de SE, désignée par* $|SE|$*, est égale au nombre de ses événements.*

Nous supposons qu'il existe un ordre sur les évènements d'une liste, désigné par $\prec$, et qui définit leur ordre dans la liste. Nous notons $E_1 \prec E_2$, si (1) $\{E_1, E_2\} \in SE_i$, et (2) $E_1.(\text{heure de début}) < E_2.(\text{heure de début})$. Ce qui signifie que E_1 et E_2 se trouvent dans la même liste d'évènement et que E_1 commence avant E_2.

Définition 5 ***(sous-séquence).*** $SE' = \{E'_1, E'_2, ..., E'_n\}$ *est appelée une sous-séquence de* $SE = \{E_1, E_2, ..., E_m\}$*, et désignée par* $SE' \subseteq SE$*, si et seulement si il existe une fonction injective* f *:* $SE'.$*évènements* $\xrightarrow{f}$ $SE.$*évènements, tel que (1)* $\forall E'_i \in SE' \rightarrow f(E'_i) \in SE$*, (2)* $E'_i.$*étiquette* $= f(E'_i).$*étiquette, et (3)* $\forall \{E'_i, E'_j\} \in SE'$*, if* $E'_i \prec E'_j \rightarrow f(E'_i) \prec f(E'_j)$*.*

Par exemple, $\{B, E\} \subset EL_3$, $EL_2 \subset EL_1$ et $EL_2 \nsubseteq EL_3$ peuvent être déduits de la Table 3.1.

3.4 Problématique

Soit ***D*** une base de données séquentielle, composée d'un ensemble de séquences d'évènements. Chaque séquence d'évènements ***SE*** est une liste d'évènements ordonnés par le temps, de longueur

A		
SID	EID	IDuration
1	1	3
2	1	5
3	1	3
4	1	7
INeighbors: { B, C, D, E }		

B		
SID	EID	IDuration
1	2	6
2	2	8
3	2	4
4	2	8
INeighbors: { C, D, E }		

C		
SID	EID	IDuration
1	3	7
2	3	9
INeighbors: $\{\emptyset\}$		

D		
SID	EID	IDuration
3	3	1
4	3	3
INeighbors: $\{\emptyset\}$		

E		
SID	EID	IDuration
1	4	9
3	4	3
INeighbors: $\{\emptyset\}$		

Table 3.2: Exemple de IDListExt construit à partir de la table 3.1 avec minSupp = 2 (durée en minutes).

différente et est identifiée par un entier unique dans D. Chaque évènement $\boldsymbol{E}$, possède une étiquette et une position dans SE. Selon la nature de D, les évènements peuvent être de type différent : entier, chaîne de caractère, binaire, etc.

L'objectif de la détection de patterns séquentiels est de trouver toutes les sous-séquences possibles qui apparaissent dans au moins ***minSupp*** séquences où *minSupp* est un seuil minimum définie par l'utilisateur.

3.5 Contribution

Nous représentons les patterns en format vertical, adopté dans [96] (les formats des patterns sont présentés plus en détails dans la section. 2.2.1). Le format vertical standard d'un pattern comporte (1) une **liste d'étiquettes** : la séquence des étiquettes de ses éléments, et (2) une **IDList** : la liste des positions (dans l'ensemble de données original) où le pattern apparaît.

Un *pattern* valide est soumis aux contraintes suivantes :

1. contrainte de fréquence : $support(pattern) \geq minSupp$, où *support* est une fonction qui mesure le nombre d'apparitions du pattern dans la base de données et *minSupp* est un

seuil de support minimum spécifié par l'utilisateur. Cela signifie que le pattern doit apparaître dans au moins *minSupp* séquences.

2. contrainte d'ordre des événements : $\forall\{EID_i, EID_j\} \in SID_l$, si $EID_i < EID_j$ alors $i < j$, où EID_i et EID_j sont respectivement la position du $i^{\text{ième}}$ et $j^{\text{ième}}$ évènement dans la $l^{\text{ième}}$ séquence. Cela préserve la propriété séquentielle des patterns.

Dans notre cas, le pattern contient plus d'information qu'un pattern standard. Il contient de plus que les étiquettes, des informations sur la durée. Chaque pattern, que nous cherchons à extraire, est composé d'une séquence d'étiquettes, qui représentent la séquence de produits consultés, couplée avec des durées moyennes passées sur chacun de ces produits. La variation de la durée est aussi importante et doit être prise en compte.

La variation de la durée. Nous ne pouvons pas nous appuyer uniquement sur la valeur de la durée pour mesurer l'intérêt du visiteur par le produit, comme cette valeur dépend de chaque visiteur. Pour cela, nous devons prendre en compte sa variation d'un produit à l'autre. De cette manière, les produits visualisés plus de temps que les autres sont susceptibles d'être intéressant pour le visiteur.

Nous avons donc besoin d'augmenter la représentation verticale standard pour prendre en compte la durée et ses variations. Pour atteindre cet objectif, nous introduisons **IDListExt** qui est une représentation verticale augmentée contenant **IDurations**, **IFlags**, et **INeighbors** en plus. La Table 3.2 contient un exemple des IDListExt.

Définition 6 ***(IDListExt).*** *Une extension de IDList, qui contient en plus (1) IDurations : liste des durées, (2) IFlags : liste des drapeaux (marqueurs), et (3) INeighbors : liste des éléments voisins.*

Définition 7 ***(IDurations).*** *Liste des durées du dernier élément du pattern dans chaque position où le pattern apparaît.*

Pour éviter la duplication des données, seules les durées du dernier élément du pattern sont conservées dans les *IDurations* de ce pattern. Pour obtenir les durées des autres éléments du

pattern, nous pouvons les calculer à partir du pattern parent du pattern en question. Plus de détails sont données en section 3.6.

Définition 8 ***(Duration).*** *La durée d_{it} d'un élément it dans un pattern p est définie comme suit :*

$$d_{it} = \frac{\sum_{i=1}^{n}(\textit{durée de } [it] \textit{ dans } [SE_i])}{p.|SID|} \tag{3.1}$$

où $it \in SE_i$ et $p \subseteq SE_i$.

Définition 9 ***(IFlags).*** *Liste de drapeaux pour chaque évènement indiquant si la durée d'un évènement est plus longue ou plus courte que celle de l'évènement suivant dans le même pattern.*

Pour faire la différence entre deux patterns qui ont la même séquence d'étiquettes, mais une différence dans la variation des durées, nous choisissons d'ajouter un drapeau à chaque évènement du pattern. Ce drapeau peut avoir une valeur de "none", "+" ou "-".

Étant donné un pattern $p = \{(E_i, D_i)(E_j, D_j)\}$, nous fixons $flag(E_i)$ à :

1. "+": si $(D_j - D_i) \geq minGap$
2. "-": si $(D_j - D_i) < minGap$
3. "none": si I_i est le dernier élément

où $minGap$ est un seuil minimum d'écart entre des éléments consécutifs, spécifié par l'utilisateur.

Par exemple, à partir de la Table 3.1, le pattern {A, B} peut être représenté avec des drapeaux et des durées par p_{AB}={(A+, 4.5) (B, 6.5)}. Nous pouvons déduire de cette représentation que les visiteurs passent en moyenne 6.5 minutes sur le produit B quand ils passent en moyenne 4.5 minutes sur le produit A auparavant.

Les patterns $p_1 = \{(A, 2)(B, 4)(C, 6)\}$ et $p_2 = \{(A, 9)(B, 4)(C, 6)\}$ peuvent être représentés avec des drapeaux par $p_1 = \{(A+, 2)(B+, 4)(C, 6)\}$ et $p_2 = \{(A-, 9)(B+, 4)(C, 6)\}$ respectivement avec $minGap = 1$.

Deux patterns ayant la même séquence d'évènement mais une différence dans la variation de la durée ne devraient pas être égaux. Cela garantit que tous les patterns trouvés présentent la même variation de durée. Par exemple, dans la Table 3.1, p_1={(B+, 6) (E, 9)} $\neq$ p_2={(B-, 4) (E, 3)}, où p_1 apparaît dans la première séquence (SID=1) et p_2 apparaît dans la troisième séquence (SID=3). Nous notons que $p_i == p_j$ si et seulement si, les séquences d'étiquettes et de drapeaux de p_i sont égales aux séquences d'étiquettes et de drapeaux de p_j.

Définition 10 ***(INeighbors).*** *Liste des étiquettes des évènements fréquents qui apparaissent après le dernier élément du pattern dans la base de données. Elle est utile pour la génération des candidats et permet un élagage plus efficace de l'espace des candidats.*

Par exemple, dans la Table 3.1, pour $minSupp = 2$, les voisins de A sont $\{< B, freq = 4 >, < C, freq = 2 >, < D, freq = 2 >, < E, freq = 2 >\}$, et ceux de B sont $\{< C, freq = 2 >, < D, freq = 2 >\}$.

Une représentation canonique d'un pattern peut être obtenue en fusionnant tous ses éléments fréquents (dans le même ordre d'apparition) tels que chaque élément a une étiquette, un drapeau et une durée moyenne. Deux patterns p_1 et p_2 sont égaux s'ils ont les mêmes séquences d'étiquettes et de drapeaux tels que : (1) $p_1.|\mathrm{E}| = p_2.|\mathrm{E}|$, et pour tout $1 \leq i \leq k$, nous avons (2) $p_1.E_i.label = p_2.E_i.label$ et (3) $p_1.E_i.flag = p_2.E_i.flag$, où E est la liste d'évènements.

Définition 11 ***(pattern).*** *Un* ***k-pattern*** *est une sous-séquence* ***fréquente****, où k est son nombre d'événements (*$k = pattern.|évènements|$*).*

3.6 Sequential Event Pattern Mining

Soit $D = \{SE_1, ..., SE_n\}$ une base de données de SE et $minSupp$ un seuil minimum spécifié par l'utilisateur. Le but est de trouver tous les patterns séquentiels apparaissant dans au moins $minSupp$ séquences. Dans cette section, nous décrivons les algorithmes que nous proposons pour (1) construire l'IDListExt de chaque événement fréquent et (2) extraire les patterns séquentiels.

Nous proposons ici, un nouvel algorithme **SEPM** (**S**equential **E**vent **P**attern **M**ining) pour extraire les patterns avec durée. Une version de cet algorithme est proposée aussi pour extraire les patterns sans durée. SEPM utilise la recherche en profondeur (Depth-First Search) pour extraire leurs patterns.

3.6.1 Construire la liste des IDListExt

L'algorithme 1 est proposé pour construire l'IDListExt de chaque événement fréquent dans *D*. Il commence par scanner *D* pour obtenir tous les événements uniques et fréquents appelés *freqEvents*. Ensuite, un deuxième scan est effectué pour construire l'IDListExt de chaque élément de *freqEvents*. La dernière opération supprime tous les voisins peu fréquents de tous les INeighbor.

Algorithm 1: Construire les IDListExt

Input: Event List: *db*

Output: Set of IDListExt: ν

```
1  freqEvents ← all labels of frequent items
2  forall el ∈ db do
3      antecedents ← ∅, EID ← 1
4      forall e ∈ el.events do
5          if freqEvents contains e.label then
6              IDListExt ← ν.findOrCreateIDListExt(e.label)
7              IDListExt.add(el.SID, EID, e.duration)
8              add e.label to all IDListExt.INeighbor in antecedents
9              antecedents ← antecedents ∪ IDListExt
10             EID ← EID + 1
11         end
12     end
13 end
14 remove infrequent neighbors from all INeighbors
```

3.6.2 SEPM sans durée

L'algorithme 2, appelé **SEPM-**, est proposé pour extraire les patterns sans durée. Les patterns résultants sont utilisés pour prouver l'utilité de la durée (voir la section 3.7 pour plus de détails). SEPM- commence par générer la liste des candidats (ligne 3). Ensuite, il calcule le support de chaque candidat, et si il est supérieure ou égale à la *minSupp* (ligne 4), le candidat devient un pattern (ligne 5). À partir de chaque pattern de taille K, SEPM- génère récursivement tous les patterns de taille K+1 (ligne 6). Ce processus récursif est répété jusqu'à ce que tous les patterns soient trouvés. Une intersection $\cap$ entre deux IDListExt est effectuée pour récupérer les positions du candidat généré. Au début de l'étape d'intersection, tous les SID communs

(entre les deux IDListExt) sont calculés. Pour chaque SID commun, nous cherchons si le voisin se trouve dans une position supérieure à la position maximale qui se trouve dans le SID du pattern. Si cette position existe, cela signifie que le candidat généré existe dans la séquence commune entre le pattern et le voisin.

Algorithm 2: SEPM-

Input: Pattern: *pattern*

Input: User-specified threshold: *minSupp*

Input: Set of IDListExt: ν

```
1 save(pattern)
2 forall neighbor ∈ pattern.INeighbors do
3     IDListExt ← pattern.IDListExt ∩ ν.find(neighbor).IDListExt
4     if IDListExt.|SID| ≥ minSupp then
5         newPattern ← createPattern(pattern, neighbor, IDListExt)
6         SEPM-(newPattern) // recursive call
7     end
8 end
```

3.6.3 SEPM avec durée

L'algorithme 3, appelé **SEPM+**, est proposé pour extraire les patterns avec durée. Il est composé des mêmes étapes que l'algorithme précédent 2, à part l'intersection $\cap$ entre les IDListExts qui est différente. Ici, l'intersection peut générer des patterns (de 0 à 2) en fonction de la variation de la durée dans chaque SID commun. Les drapeaux des patterns sont utilisés pour coder la variation des durées. Lorsqu'un pattern rejoint son voisin, la variation entre la durée de ce voisin et la durée du dernier élément du pattern dans chaque SID commun déterminera le drapeau du dernier élément du pattern (avant l'ajout du voisin au pattern). Par exemple, la jointure entre {A+B+C} et {D} peut produire : {$\emptyset$}, {A+B+C+D}, et/ou {A+B+C-D}.

Algorithm 3: SEPM+

Input: Pattern: *pattern*

Input: User-specified threshold: *minSupp*

Input: Set of IDListExt: ν

```
1  save(pattern)
2  forall neighbor ∈ pattern.INeighbors do
3      IDListExt ← pattern.IDListExt ∩ ν.find(neighbor).IDListExt
4      if IDListExt.|SID| ≥ minSupp then
5          newPatterns ← createPatternsWithDuration(pattern, neighbor, IDListExt)
6          forall newPattern ∈ newPatterns do
7              SEPM+(newPattern) // recursive call
8          end
9      end
10 end
```

Un exemple est présenté dans la Figure 3.1. Il montre comment les patterns et leurs listes IDListExt sont générés à partir d'un ensemble de données.

3.6.4 SEPM avec une extraction parallèle

L'algorithme 4, appelé **SEPM++**, est proposé pour extraire en parallèle les patterns avec durée afin d'accélérer le processus d'extraction. Le nombre de Thread utilisé peut être ajusté en fonction de processeurs, de cœurs logiques. Le résultat de SEPM++ est le même que celui de SEPM+. Au départ, un pool de Threads est créé (ligne 1), puis tous les patterns de taille 1 sont extraits (ligne 2). Ensuite, une tâche est créée pour chaque pattern et ajoutée à la liste des tâches du pool de Threads (ligne 3-6). Lorsqu'une tâche est ajoutée, elle sera directement exécutée par le premier Thread disponible. Un Thread génère tous les patterns possibles à partir du pattern d'une tâche donnée (comme dans SEPM+). Enfin, l'exécution du SEPM++ sera mise en pause tant qu'il y aura des Threads en cours d'exécution et des tâches non traitées

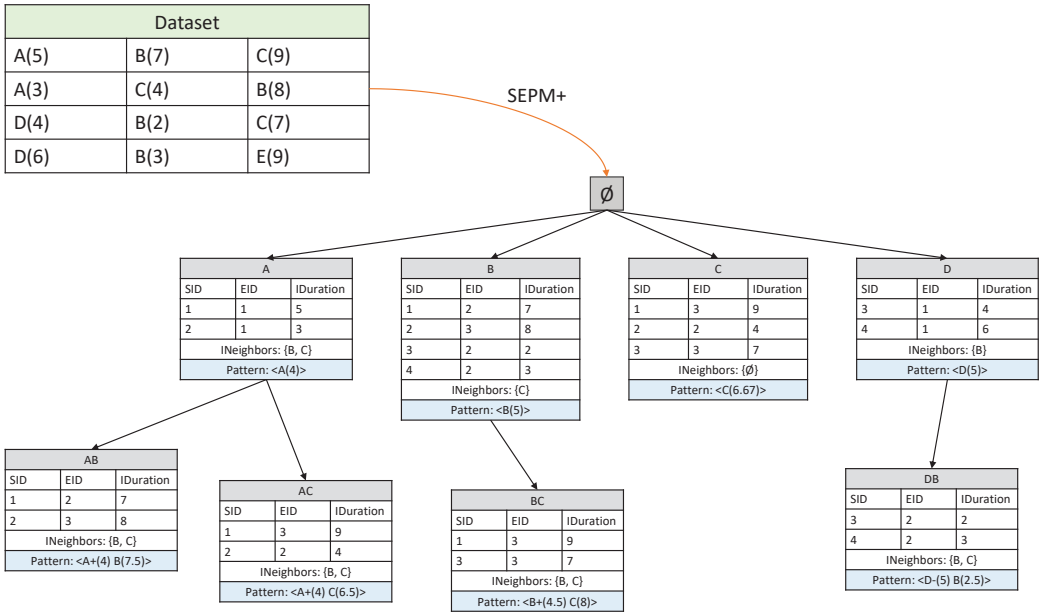


Figure 3.1: L'arbre des patterns extraits de l'ensemble de données.

(ligne 7).

Algorithm 4: SEPM++

Input: User-specified threshold: *minSupp*
Input: Number of threads: *nbThread*
Input: Set of IDListExt: *v* `// of all frequent events`

```
1 t_pool ← create a pool of nbThread threads
2 1_patterns ← build all 1-pattern from v
3 forall pattern ∈ 1_patterns do
4     task ← create a task from pattern
5     pool.add_task(task)
6 end
7 wait all thread in pool
8 shutdonw all threads in pool
```

3.7 Résultats expérimentaux

Les algorithmes que nous avons proposés ont été appliqués (1) à plusieurs ensembles de données **réelles** e-commerce pour démontrer la pertinence des patterns trouvés et (2) à plusieurs ensembles de données **synthétiques** pour prouver leur efficacité et leur extensibilité. Toutes les expériences ont été réalisées sur un ordinateur équipé d'un Core-i5 avec 5 cœurs physiques, fonctionnant sous Windows 10 et 16 Go de RAM.

3.7.1 Datasets

Dans cette section, nous décrivons les deux catégories d'ensembles de données utilisées dans nos expériences, et comment elles sont construites.

1. Les données E-commerce

La première catégorie de données comprend trois ensembles extraits de vrais sites de e-commerce [21, 12, 4]. Chaque ensemble de données contient un ensemble d'actions des utilisateurs sur des produits appelés "clickstreams". Les actions peuvent être (selon l'ensemble de données) : cliquer, visualiser, interroger ou acheter. Nous avons choisi de ne conserver que les actions de clic et d'achat, car elles sont communes aux trois ensembles de données. Il est à noter que les actions sont effectuées uniquement sur les pages des produits, les actions sur les autres pages du site (FAQ, contactez-nous, etc.) ne sont pas fournies. Les actions sont regroupées par session où chaque session appartient à un seul utilisateur et possède un identifiant unique et un timestamp. Les actions d'une session sont triées par leur heure de début.

Afin d'appliquer nos algorithmes à ces ensembles de données et avoir une meilleure qualité de résultat, nous devons les nettoyer des bruits au préalable. Ces bruits peuvent être dus à des erreurs provenant de l'outil de traçage ou à des erreurs lors du chargement des pages. Ils peuvent causer plusieurs problèmes tels que des sessions non fermées (long délai entre deux actions consécutives), beaucoup de clics consécutifs sur le même produit, etc. Dans nos

Dataset	**|SE|**	**|E|**	**|Distinct E|**	$\bar{\mathbf{E}}$	**Session avec achat**
CIKM	238k	2,121k	75k	3.11	12k
YooChoose	670k	3,87k	45k	3.77	410k
AliBaba[1]	1,119k	13,966k	4,085k	6.46	630k

Table 3.3: Résumé des propriétés des ensembles de données du e-commerce. E : Événements. SE : Séquence d'évènements. $\bar{\text{E}}$: le nombre de E en moyen dans SE.

expérimentations, nous avons choisi de fixer une durée maximale sur une page à cinq minutes, pour éliminer des erreurs probables dans les données. Toute valeur dépassant ce seuil sera réduite à la valeur du seuil. Les durées seront donc toujours supérieures à zéro et inférieures à cinq minutes. Pour des raisons expérimentales, cette valeur est fixée à cinq minutes, ce qui correspond au troisième quartile (75e percentile) des valeurs des durées dans les ensembles de données. Dans un cas idéal, des algorithmes de détection des outliers devraient être utilisés pour éliminer les valeurs aberrantes sans avoir à fixer une valeur de durée maximale. Nous n'avons pas abordé ce point car il n'était pas notre priorité dans cette thèse.

Après le nettoyage, les clickstreams devraient être convertis en données d'événements séquentiels. Chaque session est convertie en une liste d'événements et chaque clic dans une session est converti en un événement attaché à sa liste d'événements correspondante. La durée d'un événement E_i est définie comme le temps entre deux événements consécutifs (E_i et E_{i+1}) dans la même session. Toutes les durées des clics dans une session peuvent être obtenues, à l'exception de la durée du dernier. Cette durée est remplacée par la durée moyenne de tous les clics dans la session en question.

La Table 3.3 contient quelques statistiques sur ces trois ensembles de données, et la Table 3.4 contient le nombre et la longueur des patterns trouvés par SEPM+.

[1] un échantillon de 1,119k lignes

Dataset	minSupp	Nb Patterns	Longueur Max
CIKM	0.01%	23,221	4
YooChoose	0.01%	26,807	8
AliBaba	0.01%	8,461	6

Table 3.4: Statistiques des patterns trouvés en utilisant SEPM+. Le nombre de patterns trouvés peut être différent de ceux trouvés par le SEPM car la représentation des patterns est différente. Les ensembles de données contiennent des sessions avec et sans achat.

2. Les données Synthétiques

L'ensemble de données synthétiques a été généré en utilisant *IBM data quest generator*. Ils sont utilisés pour étudier l'effet du changement des propriétés de la base de données d'entrée sur le temps d'exécution. La génération des ensembles de données peut être contrôlée par plusieurs paramètres :

- **ncrust** (nombre de sessions dans la base de données) pour faire varier le nombre de listes d'événements (noté par **D**)
- **slen** (moyenne des actions par client) pour faire varier le nombre d'événements par liste (noté par **L**)
- **nitems** (le nombre des différents produits disponibles) pour faire varier le nombre des étiquettes uniques des événements (noté par **T**)

Trois ensembles de données synthétiques sont générés en faisant varier chaque paramètre. La Table 3.5 présente quelques statistiques sur les trois ensembles de données synthétiques générés

- Pour *D*, nous fixons *minSupp* à 10% et nous faisons varier *D* entre 250k et 400k (jeu de données **SYN1-4**)
- Pour *L*, nous fixons *minSupp* à 10% et nous faisons varier *L* entre 8 et 15 (jeu de données **SYN5-8**).

Dataset	|SE|	|E|	|Distinct E|	$\bar{E}$
SYN1	250k	7,630k	3,240	30
SYN2	300k	9,160k	3,240	30
SYN3	350k	10,680k	3,240	30
SYN4	400k	12,215k	3,240	30
SYN5	300k	7,080k	3,240	25
SYN6	300k	9,160k	3,240	30
SYN7	300k	12,200k	3,240	40
SYN8	300k	14,220k	3,240	50
SYN9	300k	9,160k	2,250	30
SYN10	300k	9,160k	4,350	30
SYN11	300k	9,160k	5,490	30
SYN12	300k	9,160k	6,140	30
SYN13	300k	9,160k	3,240	30

Table 3.5: Un résumé des ensembles de données synthétiques. E : Événements. SE : Séquence d'évènements. $\bar{E}$: le nombre de E en moyen dans SE.

- Pour *T*, *minSupp* est fixé à 5% et *T* est variée entre 3 et 10 (ensemble de données **SYN9-12**).
- Enfin, l'ensemble de données **SYN13** a été généré pour évaluer la performance des algorithmes par rapport au minSupp

3.7.2 Catégorisation

Certains des patterns de longueurs 4 trouvés sont présentés dans la Table 3.7. Nous pouvons observer que ces patterns sont plus instructifs que ceux qui ne contiennent que des étiquettes puisqu'ils contiennent également la durée. Pour tirer profit de ces patterns et les analyser, nous proposons de les classer par catégories. Nous constatons que les durées ne sont pas toujours équivalentes, elles sont différentes d'un élément à l'autre dans le même pattern (voir la Table. 3.7). Nous constatons également une différence dans la variation de la durée dans un même pattern. Sur la base de ces deux observations, nous décidons de classer les patterns en

fonction de deux propriétés : l'écart type des durées appelé **sd**, et le nombre de variations des durées appelé **variation**, tel que :

$$sd_p = \sqrt{\frac{\sum_{i=1}^{l_p}(d_i - v)^2}{l_p}} \tag{3.2}$$

où l_p est la longueur du pattern **p**, d_i est la durée de son élément i^{th}, et v est la moyenne de toutes les durées dans p,

$$variation_p = \sum_{i=1}^{l_p-1}(\textbf{diff}(flag_i, flag_{i+1})) \tag{3.3}$$

où $flag_i$ est le drapeau de l'élément i^{th} dans p et $diff(x, y)$ est égal à 0 si x est égal à y et à 1 sinon.

Pour détecter les comportements anormaux, nous utilisons deux seuils spécifiés par l'utilisateur : *maxVariation* et *maxSD* pour les filtrer. Dans nos tests, nous fixons *maxVariation* à 10, *maxSD* à 2, et nous ne gardons que les patterns comportant au moins 4 éléments. L'algorithme 5 décrit les principales étapes de la catégorisation et la Table 3.6 présente quelques statistiques sur les différentes catégories détectées.

Dataset	var=0	var=1	var=2	var=3	...	Abnormal
CIKM	12,539	6,170	4,200	115	...	23
YooChoose	16,734	8,127	551	448	...	119
AliBaba	5,091	1,304	660	525	...	145

Table 3.6: Statistiques sur les catégories résultantes. var=i signifie que la variation est égale à i.

Algorithm 5: Categorization

Input: Patterns: *patterns*

Input: User-specified threshold: *maxSD*

Input: User-specified threshold: *maxVariation*

```
1  categories ← ∅
2  forall pattern ∈ patterns do
3      pVariation ← pattern.variation()
4      pSD ← pattern.sd()
5      index ← −1
6      if pVariation < maxVariation and pSD < maxSD then
7          index ← find_category(npVariation)
8      else
9          index ← 0 // First category is reserved to the abnormal
               patterns
10     end
11     add pattern to categories[index]
12 end
```

3.7.3 Patterns discriminants

Dans cette section, nous évaluons l'efficacité des patterns avec durées exploités avec SEPM+. L'objectif de cette expérience est de prouver qu'en utilisant la durée, nous pouvons rendre

Étiquette du produit (durée en minutes)							
9344+(3.17)	→	9344-(4.5)	→	3599+(1.99)	→	14458(5)	
3785+(0.69)	→	32320+(2.8)	→	8848+(4)	→	35358(5)	
4352300-(2.04)	→	2790543+(0.17)	→	5051027+(0.47)	→	2859111(1.58)	
4960783+(0.17)	→	2081505+(0.15)	→	2020265+(0.21)	→	4614885(0.22)	

Table 3.7: Exemple de patterns trouvés de longueur 4.

les patterns plus discriminants que ceux sans la durée. Dans cette expérience, nous utilisons SEPM+ et SEPM-[2] pour détecter respectivement les patterns avec durée et les patterns sans durée.

Préparations des patterns

Nous commençons par diviser le clickstream en deux groupes : le premier contient les sessions des utilisateurs avec au moins un achat et le deuxième contient le reste (ceux qui ne contiennent aucun achat). Ensuite, en utilisant SEPM+ et SEPM-, nous explorons les patterns dans chaque groupe. Avec chaque algorithme, nous obtenons des patterns qui conduisent à un achat appelé **P+** et des patterns qui ne conduisent pas à un achat appelé **P-**. Pour une utilisation correcte de ces patterns, les patterns non discriminants appelés P_{non_disc} doivent être supprimés. P_{non_disc} sont les patterns qui existent à la fois dans P+ et P-. L'objectif est de prouver que SEPM+ est capable de rendre discriminants les patterns non discriminants trouvés par SEPM-.

Par exemple, étant donné un pattern $P_{non\_disc\_1} = \{A, B\}$ qui existe à la fois dans P+ et P-, il est difficile de savoir si $P_{non\_disc\_1}$ représente des comportements d'achat ou non, puisque les patterns sont représentés uniquement par des étiquettes (résultats de la SEPM-). Au contraire, avec l'algorithme SEPM+ que nous proposons, la durée ajoutée aux patterns sera utilisée pour séparer les patterns qui ont la même séquence d'étiquettes, mais pas la même séquence de

[2]tout autre algorithme classique d'extraction de patterns séquentiels (comme SPADE, SPAM, etc.) peut être utilisé puisqu'ils ont tous la même sortie/la même représentations des patterns.

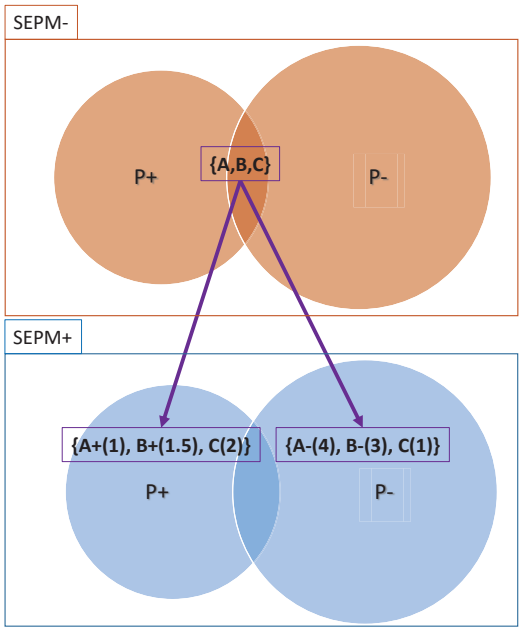


Figure 3.2: $\{A, B\}$ est un pattern non discriminant dans SEPM-, mais ce n'est pas le cas dans SEPM+ car les patterns sont représentés différemment.

drapeaux (voir la Figure. 3.2).

Efficacité

Pour mesurer le pouvoir de discrimination de la durée ajoutée, nous commençons par (1) supprimer tous les patterns de longueurs 1 car ils ne sont pas très pertinents dans l'analyse. Ensuite (2) nous détectons les patterns P_{non_disc} qui existent à la fois dans P+ et P- avec SEPM- (voir la figure 3.2). Enfin (3) nous vérifierons si P_{non_disc} (trouvé précédemment) existe dans P+ ou P- avec SEPM+. De cette manière, nous vérifions si les patterns non discriminants rejetés par SEPM- deviennent discriminants avec SEPM+.

Algorithme utilisé	Paramètres & Résultats	CIKM	YooChoose	AliBaba
SEPM+/SEPM-	MinSupp	4 SE	4 SE	4 SE
SEPM+/SEPM-	MinGap	1.5 min	1.5 min	1.5 min
SEPM+	NbPatterns (avec achat)	5k	93k	14k
SEPM+	NbPatterns (sans achat)	72k	209k	13k
SEPM-	NbPatterns (avec achat)	5k	175k	22k
SEPM-	NbPatterns (sans achat)	87k	282k	18k
SEPM+	P_{non_disc} (rejetés)	8%	2%	1%
SEPM-	P_{non_disc}	88%	31%	9%
	Effectiveness	**98%**	**97%**	**99%**

Table 3.8: Efficacité de la durée ajoutée, où *effectiveness* est le pourcentage de pattern non discriminant rejeté par SEPM- et conservés par SEPM+.

Résultats

La Table 3.8 montre les résultats de cette expérience. Comme on peut le voir, tous les patterns non discriminants rejetés par un algorithme sans durée (pendant la phase d'analyse) deviennent discriminants avec SEPM+. Cela prouve que nous parvenons à transformer des patterns non discriminants en patterns discriminants en ajoutant la durée dans la représentation du pattern.

3.7.4 Performances

Dans cette section, nous appliquons sept algorithmes (SPEM [-,+,++], PrefixSpan, SPAM et SPADE) à des ensembles de données synthétiques. Les codes sources des algorithmes (PrefixSpan, SPAM, et SPADE) sont fournis par [26]. Les Figures [3.3, 3.4, et 3.5] montrent le temps de réponse de ces algorithmes en fonction de certaines dimensions de l'ensemble de

données. Dans tous les tests sauf le dernier, 5 Threads sont utilisés pour exécuter SEPM++. La Figure 3.6 montre le temps de réponse de ces algorithmes lorsque *minSupp* varie et la Figure 3.7 montre le temps de réponse de SEPM++ en fonction du nombre de Threads utilisés. Comme on peut le voir, dans tous les cas, les algorithmes que nous proposons se comportent de la même manière que les autres algorithmes lorsque les propriétés de la base de données changent. Les résultats montrent que nos algorithmes se comportent correctement en fonction des différents types de bases de données.

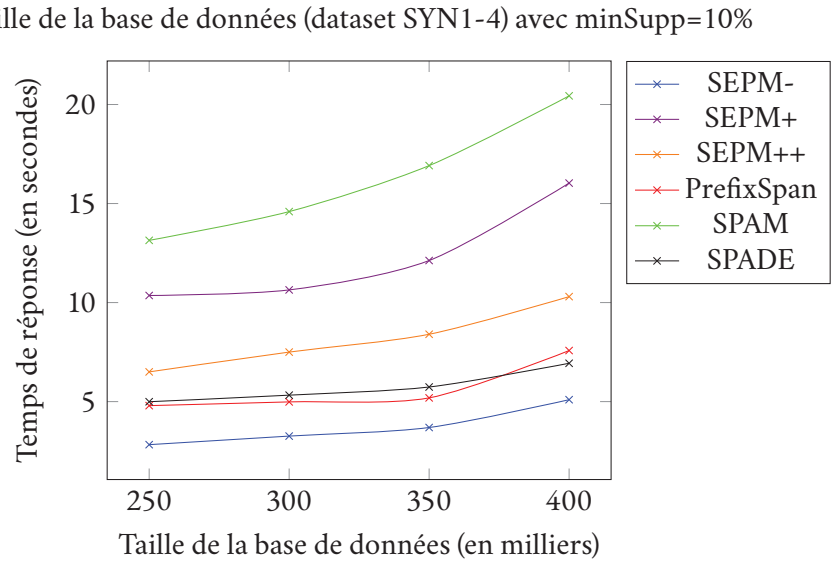


Figure 3.3: L'effet de varier les dimensions de la base de données.

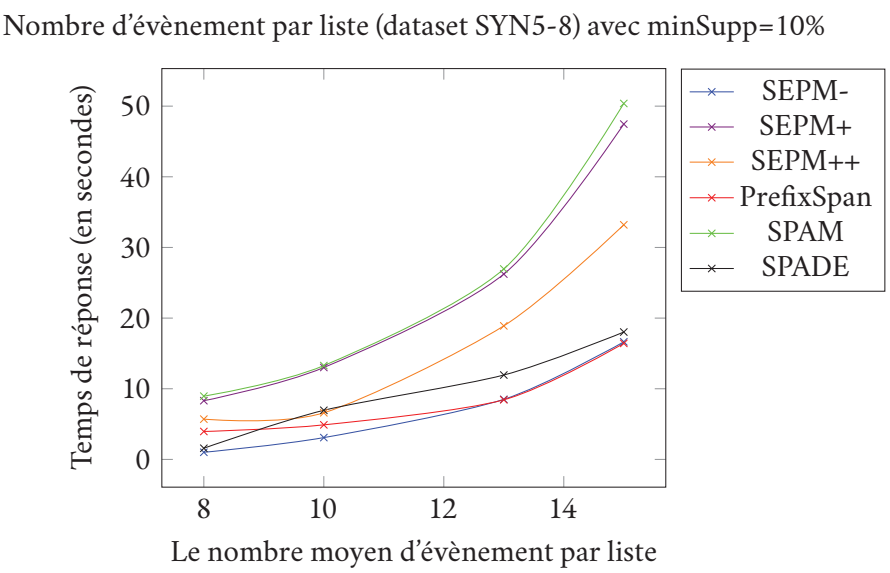


Figure 3.4: L'effet de varier les dimensions de la base de données.

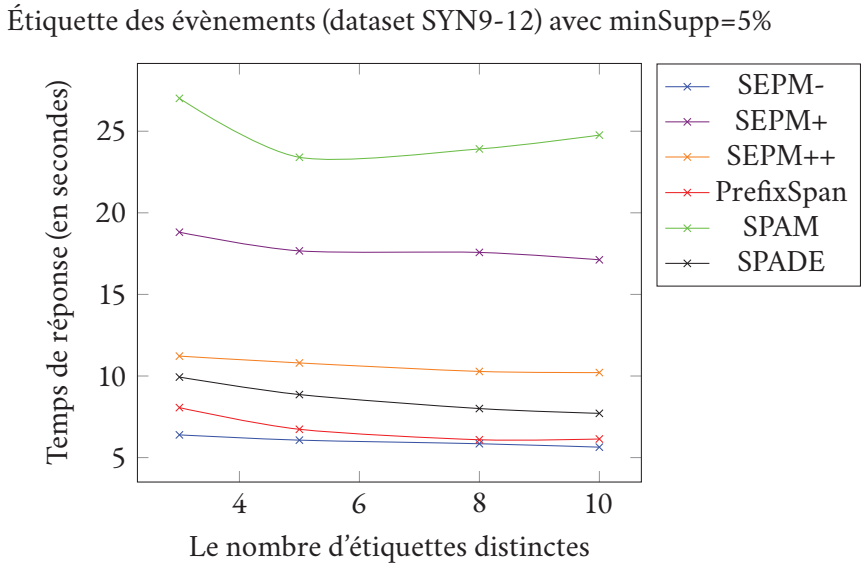


Figure 3.5: L'effet de varier les dimensions de la base de données.

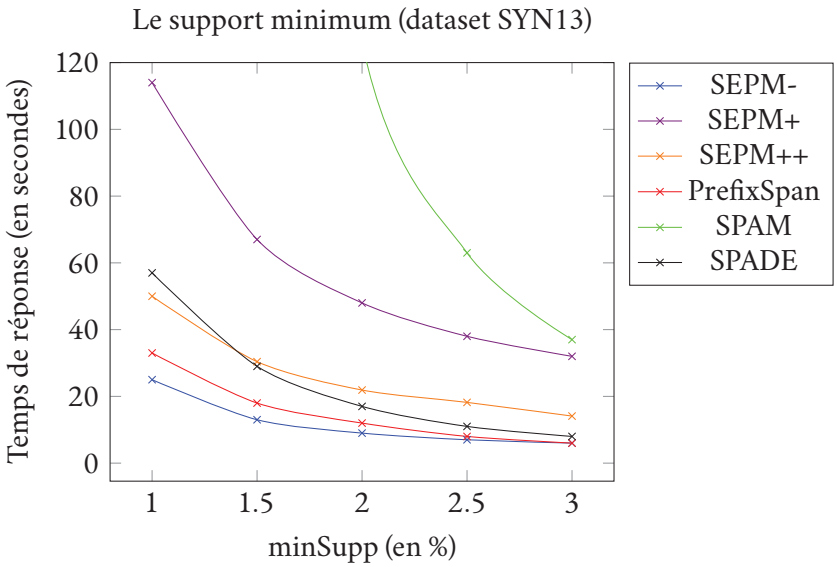


Figure 3.6: L'effet de varier le *minSupp*

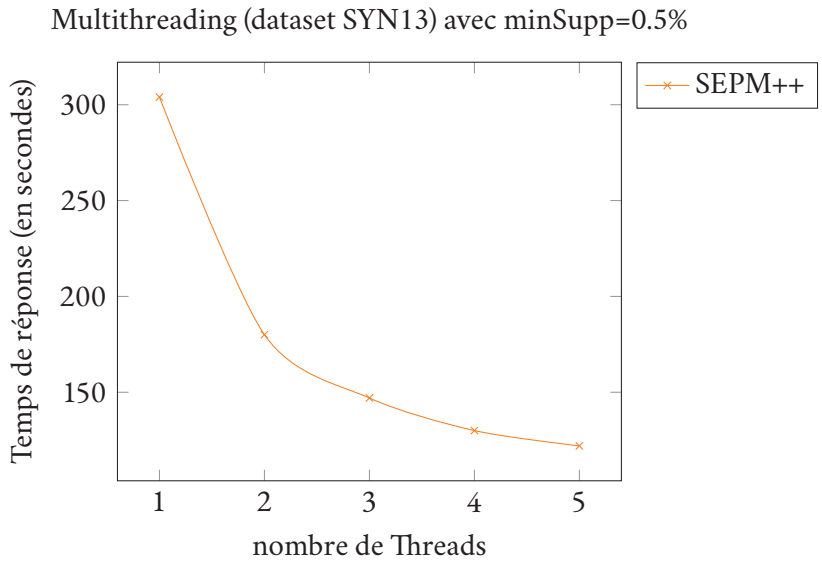


Figure 3.7: L'effet de varier le nombre de Thread

3.8 Conclusion

Les vendeurs en ligne tentent de comprendre les comportements de leurs clients et de découvrir comment ils naviguent sur leur site de commerce électronique, pourquoi ils sélectionnent un produit particulier, voire pourquoi ils abandonnent leur processus d'achat. Nous pensons que la durée passée par un client sur une page d'un produit sur un site d'e-commerce est crucial pour comprendre ses préférences et cette durée peut être utilisée comme un indicateur de l'intérêt du client pour un produit. Motivés par cette idée, nous avons proposé un nouveau modéle de comportement appelé "comportement fréquent". Nous avons ajouté la durée aux patterns et nous avons proposé deux algorithmes (SEPM+ et SEPM++) pour exploiter les patterns séquentiels, y compris la durée moyenne de leurs items.

Nous montrons que les patterns qui en résultent sont utiles pour découvrir des relations cachées entre les produits. Nous avons également catégorisé les patterns afin de simplifier leur interprétation et de détecter les comportements discriminants. Les résultats expérimentaux sur des ensembles de données réels et synthétiques montrent l'efficacité des résultats et l'extensibilité des algorithmes que nous proposons.

4

Modéliser un comportement intéressant

Dans ce chapitre, nous élargissons notre travail pour étudier le comportement intéressant. Ici, le comportement n'est plus modélisé par une session, mais plutôt par une séquence de sessions.L'objectif est donc d'étudier les séquences de sessions des utilisateurs dès leurs arrivés sur le site d'e-commerce et jusqu'à l'achat ou non. Nous modélisons ce type de comportement par des séries temporelles multi-variés et nous proposons un framework de clustering de ses séries. Le framework prend en entrée un ensemble de séries temporelles et retourne en sortie l'ensemble de clusters de ces séries.

Plan du chapitre

4.1 Introduction

Dans ce chapitre, nous nous intéressons à la technique du clustering, qui consiste à diviser un ensemble d'objets en plusieurs sous-groupes homogènes appelés clusters. Les objets d'un même cluster doivent être aussi semblables que possible les uns aux autres (similitudes internes) et aussi différents que possible des autres dans un autre cluster (similitudes externes). Les objets d'un même cluster partagent des critères similaires, et ceci peut être mesuré par une fonction de similarité qui doit être adaptée en fonction de chaque type de données et de l'objectif à atteindre. Par exemple, il peut s'agir de la distance euclidienne si les objets sont des vecteurs dans espace cartésien, ou de la distance de Levenshtein si les objets sont des chaînes de caractères. Le choix de la fonction de similarité n'est pas une tâche triviale. Il devient plus compliqué lorsque les données sont complexes, comme les données de séries temporelles qui peuvent avoir des longueurs ou des formes différentes. L'utilisation des mesures de similarité classiques peut conduire à de mauvais résultats, ce qui impacte directement la qualité des clusters.

Dans notre proposition, nous tirons parti de plusieurs familles d'approches (celles basées sur la forme, et celles basées sur un modèle), pour améliorer la qualité (précision) du clustering, tout en utilisant le Bagging. Le framework que nous proposons est génératif et les méthodes utilisées peuvent être remplacées par d'autres selon les besoins.

4.2 Motivations

Pour étudier plus en détail le comportement des utilisateurs, il faut prendre en compte l'ensemble des actions des utilisateurs dans leurs globalités. Ce qui revient à analyser leurs séquences de sessions. Ici, la session d'un utilisateur contiendra, en plus de la séquence des étiquettes, des informations supplémentaires et utiles, comme la durée de la session, le nombre de produits visualisés, le nombre de catégories, etc. Chaque session est représentée par un vecteur

de données et l'ensemble des sessions d'un utilisateur est représenté par une série temporelle composée de ses vecteurs. Pour cela, nous décidons d'utiliser la technique de clustering des séries temporelles pour regrouper les parcours des clients qui se ressemblent. En plus du clustering, nous cherchons à trouver le modèle descriptif de chaque cluster qui nous donnera des informations sur les propriétés du cluster.

4.3 Contribution

Comme mentionné précédemment, notre objectif est de construire un modèle descriptif des données des séries temporelles. Pour atteindre cet objectif, nous proposons le framework suivant appelé **G**enerative time series **C**lustering with **Bag**ging (GCBag). Il s'agit d'un framework génératif, les méthodes utilisées peuvent être remplacées par d'autres méthodes qui sont mieux adaptées aux données étudiées.

Le schéma du GCBag est décrit dans la Figure 4.1. Il prend comme entrée un ensemble de séries temporelles T et renvoie un ensemble de clusters K accompagné de la matrice d'appartenance correspondante.

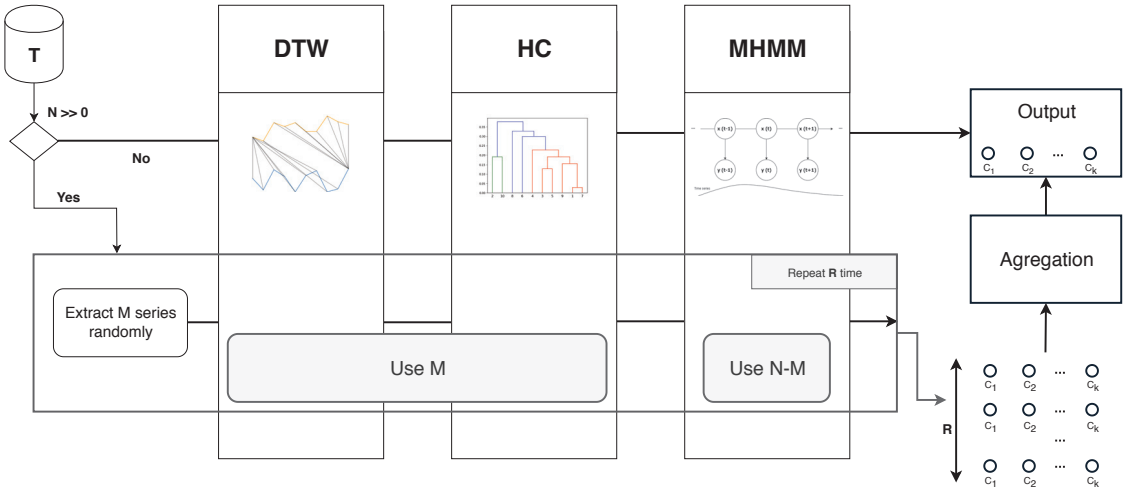


Figure 4.1: Le schéma du framework GCBag proposé.

GCBag commence d'abord par calculer le nombre de séries temporelles T, désigné par $|T|$. S'il s'agit d'une petite valeur, GCBag utilise une stratégie de clustering simple, sinon, il utilise une stratégie plus avancée.

La stratégie simple. Étant donnée un petit ensemble de séries T, GCBag commence par construire la matrice de dissimilarité des séries temporelles, en utilisant DTW, et qui va contenir toutes les mesures de similarité entre les séries. Ensuite, cette matrice est utilisée comme entrée pour le HC (Hierarchical Clustering) afin d'obtenir un clustering initial. Ce clustering initial est la première version des clusters résultats. Cette version peut contenir des erreurs que nous devons corriger. Toute autre méthode de clustering qui prend une matrice de dissimilarité en entrée peut être utilisée (comme le spectral clustering, le partition around medoids, ...). Disposant de ces clusters initiaux, le GCBag affecte un HMM à chaque cluster et utilise l'algorithme EM (Expectation-Maximization) pour estimer leurs paramètres. Une fois les paramètres des HMMs sont estimés, les probabilités que chaque série temporelle appartienne à chaque HMM sont calculées, et chaque série temporelle va rejoindre le groupe avec la plus grande probabilité. Nous répétons ce processus (ré-estimation et ré-allocation) jusqu'à la convergence, lorsque la probabilité globale (le likelihood) cesse d'augmenter. Reportez-vous à la section 4.3.2 pour plus de détails sur le clustering utilisant un mélange de HMM.

La stratégie avancée. Comme l'estimation des paramètres d'un HMM est très sensible à son point de départ, il est fortement recommandé d'utiliser une méthode d'initialisation intelligente avant d'ajuster les HMMs aux données. Dans la stratégie simple, nous appliquons DTW et HC sur l'ensemble des données pour fournir des clusters de départ. Mais dans le cas où les données sont à grande échelle, d'autres défis sont présents et à relever. Par exemple, la DTW souffre de la lourdeur des calculs nécessaires pour trouver son chemin d'alignement optimal, ce qui peut augmenter considérablement le temps d'exécution du clustering. De plus, l'utilisation de la DTW sur des données contenant des bruits peut conduire à un résultat de mauvaise qualité. Pour surmonter ces inconvénients et améliorer la qualité du clustering, nous utilisons une méthode de Bagging. Cette méthode extrait un échantillon de T, de manière

uniforme et avec remplacement, et l'utilise comme entrée pour DTW+HC afin d'obtenir les clusters de départ. Ensuite, un HMM est attribué à chaque cluster et est formé sur les données restantes pour estimer ses paramètres. Enfin, le processus de ré-estimation et de ré-allocation est répété jusqu'à la convergence. Cette méthode de Bagging est répétée R fois pour utiliser plusieurs échantillons différents et produire plusieurs résultats différents.

Après les R itérations du Bagging, nous obtenons $R \times K$ clusters, où K est le nombre de clusters extrait par itération. Comme nous avons besoin d'un seul groupe de clusters, nous devrions unifier ces $R \times K$ clusters à fin d'obtenir que K clusters. Pour cela, nous avons proposé une méthode appelée "Group-and-Merge" (GaM). Cette méthode prend comme entrée un ensemble de clusters (ensemble de HMMs) et retourne le consensus de ces clusters. La méthode GaM est détaillée dans la section 4.3.3. Nous n'avons pas utilisé un algorithme de clustering consensuel existant, car il ne préserve pas l'aspect descriptif des clusters et ne fonctionne que sur la matrice d'appartenance. Si le modèle descriptif n'est pas important pour l'analyse, le GaM peut être remplacé par n'importe quelle méthode de regroupement par consensus, qui peut unifier la liste de la matrice d'appartenance. Après la fusion, lorsque les clusters finaux sont extraits, la matrice d'appartenance (séries temporelles vs clusters) est construite en utilisant le likelihood.

La Table 4.1 contient les notations utilisées dans le reste du chapitre.

4.3.1 Bagging

Le bagging permet d'accroître la robustesse du modèle en offrant plusieurs points de départ au framework et l'aide à éviter autant que possible les maximums locaux et le sur-apprentissage (overfittings).

À chaque itération, M séries temporelles sont extraites, uniformément et avec remplacement, de T. Chaque série temporelle a la même probabilité $(1/|T|)$ d'être extraite, et peut apparaître dans différents échantillons, mais pas plus d'une fois dans le même échantillon. Les échantillons sont utilisés comme entrée dans DTW+HC pour obtenir un clustering ini-

Notation	Description
Inputs	
T	Ensemble de données contenant toutes les séries temporelles T_i est la série temporelle à la position i dans T avec $0 \leq i \leq \|T\|$ $\|T_i\|$ est le nombre d'observations dans T_i
λ	Modèle HMM ayant $\lambda = (\pi, A, B)$
π	Probabilités de départ
A	Matrice de transition
B	Probabilités d'émission
C	Nombre d'état du HMM
Clustering Parameters	
K	Nombre de clusters
M	Nombre de séries temporelles dans l'échantillon
R	Nombre d'itérations dans la méthode de Bagging

Table 4.1: Notations

tial. Après l'initialisation, les séries temporelles restantes sont utilisées pour former les HMMs et estimer leurs paramètres. Chaque itération du bagging dans la "stratégie avancée" peut être considérée comme une "stratégie simple" où les données d'entrée sont divisées en deux sous-groupes, l'un utilisé pour l'initialisation et l'autre pour la formation. Pour un clustering efficace, il est nécessaire que la condition suivante soit valide : $K \leq M \leq 0,6 \times |T|$.

4.3.2 Le clustering en tant que mélange de HMMs

Nous supposons que chaque série temporelle est générée à partir d'un HMM ayant un mélange de composantes *C*, et que chaque composante est un modèle gaussien. Comme le GCBag est un framework génératif, il permet de remplacer le modèle gaussien par toute autre distribution de probabilité (comme la distribution de Poisson).

Pour estimer le mélange, une approche EM pour le soft-clustering proposé dans [7] est utilisée. Dans le cas du soft-clustering, chaque série temporelle peut appartenir à plus d'un clus-

ter avec différentes probabilités[1]. L'utilisation du soft-clustering nous permet de détecter les chevauchements entre les clusters et d'éviter les erreurs d'affectation qui peuvent être rencontrées avec le hard-clustering (où chaque série appartient à un et un seul cluster).

Ici, nous voulons estimer $\theta = \{\lambda^{(k)}, p^{(k)} | 1 \leq k \leq K\}$, où $p^{(k)}$ est la probabilité a priori que les séries temporelles se trouvent dans le $k^{\text{ième}}$ cluster. Si nous cherchons K clusters, nous devrions estimer les paramètres de l'ensemble $\{\theta_1, \theta_2, ..., \theta_K\}$ qui maximisent la fonction de likelihood $\mathcal{L}$ où :

$$\mathcal{L} = \prod_{t=1}^{T} \sum_{k=1}^{K} p^{(k)} P(O^{(t)} | \lambda^{(k)}) \tag{4.1}$$

L'entraînement est effectué séparément pour chacun des K clusters, et la probabilité postérieure w_{tk}, qu'un $O^{(t)}$ observé appartienne à un cluster k, est calculée pour chaque observation et chaque cluster. Chaque itération de la méthode EM comprend les deux étapes suivantes :

- L'étape Espérance [2]

$$w_{tk} = \frac{p^{(k)} P(O^{(t)} | \lambda^{(k)})}{\sum_{k=1}^{K} p^{(k)} P(O^{(t)} | \lambda^{(k)})} \tag{4.2}$$

- L'étape Maximisation

$$\bar{p}^{(k)} = \frac{\sum_{t=1}^{|T|} w_{tk}}{L} \tag{4.3}$$

Pour mettre à jour les estimations $\bar{\lambda}^k$:

$$\bar{\pi}_i^{(k)} = \frac{\sum_{t=1}^{|T|} w_{tk} \gamma_1^{(tk)}(i)}{\sum_{t=1}^{|T|} w_{tk}} \tag{4.4}$$

[1] la somme de ces probabilités doit être égale à 1

[2] "Expectation" en anglas

$$\bar{a}_{ij}^{(k)} = \frac{\sum_{t=1}^{|T|} w_{tk} \sum_{l=1}^{|T_t|-1} \xi_l^{(tk)}(i,j)}{\sum_{t=1}^{|T|} w_{tk} \sum_{l=1}^{|T_t|-1} \gamma_l^{(tk)}(i)} \tag{4.5}$$

$$\bar{\mu}_i = \frac{\sum_{t=1}^{|T|} w_{tk} \sum_{l=1}^{|T_t|-1} \gamma_l^{(tk)}(i).O_l}{\sum_{t=1}^{|T|} w_{tk} \sum_{l=1}^{|T_t|-1} \gamma_l^{(tk)}(i)} \tag{4.6}$$

$$\bar{\sum}_i = \frac{\sum_{t=1}^{|T|} w_{tk} \sum_{l=1}^{|T_t|-1} \gamma_l^{(tk)}(i).(O_l - \mu_i)(O_l - \mu_i)'}{\sum_{t=1}^{|T|} w_{tk} \sum_{l=1}^{|T_t|-1} \gamma_l^{(tk)}(i)} \tag{4.7}$$

où $\gamma_l(i)$ est la probabilité d'être dans l'état i au temps l, et $\xi_l(ij)$ est la probabilité d'être dans l'état i au temps l et de transiter vers l'état j au temps $l+1$. Ces étapes sont répétées jusqu'à la convergence, lorsque la probabilité cesse d'augmenter pour un nombre fixe d'itérations.

4.3.3 L'agrégation des clusters

Comme nous utilisons une méthode de bagging, après chaque itération $r < R$ une partition $\mathcal{G}^r = \{\theta_1^r, \theta_2^r, ..., \theta_K^r\}$ est retournée. Le but est maintenant de trouver le consensus des clusters à partir de l'ensemble des $\mathcal{G}$. Nous proposons une méthode appelée Group-and-Merge (GaM) pour fusionner ces partitions. Nous commençons d'abord par regrouper les clusters similaires, puis nous les fusionnons pour créer la version de consensus.

Groupe. Nous laissons chaque $\theta^{(i)}$ générer un échantillon T^i. Ensuite, nous construisons une matrice de dissimilarité D pour exprimer la paire de similarité entre les HMMs. Pour mesurer la similarité entre deux HMMs, nous avons utilisé la métrique proposée dans [32], qui est basée sur la distance de Kullback-Leibler (KL). Par conséquent, la distance $D_{KL}(\lambda_i; \lambda_j)$ entre deux modèles HMMs λ_i et λ_j est exprimée comme la distance KL entre leurs distributions

discrètes correspondantes $\tilde{P}_{\lambda_i}$ et $\tilde{P}_{\lambda_j}$, où :

$$D_{KL}(\lambda_i;\lambda_j) = D_{KL}(\tilde{P}_{\lambda_i};\tilde{P}_{\lambda_j}) = \sum_{i=1}^{N} \tilde{P}(T_i|\lambda_i).log\frac{\tilde{P}(T_i|\lambda_i)}{\tilde{P}(T_i|\lambda_j)} \tag{4.8}$$

et

$$D(\lambda_i;\lambda_j) = \frac{1}{2}(D_{KL}(\lambda_i;\lambda_j) + D_{KL}(\lambda_j;\lambda_i)) \tag{4.9}$$

Grâce à la matrice de dissimilarité, une méthode de regroupement hiérarchique (ou d'autres algorithmes de regroupement basés sur la distance) est utilisée pour regrouper les HMMs similaires.

Merge. À ce niveau, les HMMs similaires sont regroupés et l'objectif est maintenant de les fusionner. Nous choisissons d'utiliser le centre de chaque groupe de HMM pour les représenter en tant que groupe de consensus. Pour cela, nous pouvons utiliser les valeurs moyennes (ou médianes) de tous les paramètres (π, A, B) des HMMs du même groupe.

4.4 Résultats expérimentaux

Dans cette section, nous présentons nos expériences pour évaluer le framework GCBag proposé, sur des ensembles de données réelles et synthétiques, en les comparant à trois autres algorithmes de clustering de séries temporelles :

1. DTW+HC [62] (nous remplaçons k-means par HC) : utilise la DTW pour calculer la distance entre les séries temporelles et HC pour les clusteriser

2. DTW+HC+MHMMs [93] : utilise DTW+HC suivi d'un mélange de HMMs

3. HMMs+HC [32] : calcule les distances entre les HMMs (sur la base de la distance KL) et utilise ensuite HC pour les clusteriser

4.4.1 Résultats sur les données synthétiques

Pour toutes les expérimentations sur les données synthétiques, la *v-measure* [72] est utilisé pour évaluer la qualité du clustering, avec :

$$v\text{-}measure = (1+\beta) + \frac{h.c}{\beta.h + c} \tag{4.10}$$

$$h = 1 - \frac{H(Y_{true}|Y_{pred})}{H(Y_{true})} \tag{4.11}$$

$$c = 1 - \frac{H(Y_{pred}|Y_{true})}{H(Y_{pred})} \tag{4.12}$$

où h est l'homogénéité, c est la complétude, H est l'entropie conditionnelle, et β est le poids de la moyenne harmonique de l'homogénéité et de la complétude.

Toutes les données ici sont générées par des HMMs, nous savons donc à l'avance qu'elles contiennent des clusters, ce qui justifie notre choix d'utiliser une métrique d'évaluation supervisée. En ce qui concerne les paramètres du GCBag, nous avons fixé M à 60% de $|T|$ et R à 12.

Test 1. Test simple. Des séries unidimensionnelles ont été générées à partir de deux HMMs λ_1 et λ_2 (comme dans [80, 7]). Chaque HMM a deux composantes, avec :

$$A_1 = \begin{pmatrix} 0.6 & 0.4 \\ 0.4 & 0.6 \end{pmatrix}; A_2 = \begin{pmatrix} 0.4 & 0.6 \\ 0.6 & 0.4 \end{pmatrix}$$

Chaque composante obéit à une densité gaussienne où la moyenne et la variance sont respectivement :

$$\mu_1 = \mu_2 = \begin{pmatrix} 0 & 3 \end{pmatrix}; V_1 = V_2 = \begin{pmatrix} 1 & 1 \end{pmatrix}$$

Nous générons 40 séries temporelles (20 de chaque HMM) de longueurs 200. Pour générer

une série temporelle de longueur L à partir d'un HMM donné, nous effectuons une boucle de L itérations sur les actions suivantes : (1) pour tout $t < L$, l'état caché sélectionné génère un point selon sa distribution gaussienne, puis (2) nous sélectionnons, pour l'itération $t+1$, l'état caché suivant (qui peut être le même que l'itération courante) selon la matrice de transition. Pour sélectionner l'état caché à $t = 0$, nous utilisons les probabilités de départ π. Le regroupement de ces séries temporelles n'est pas trivial, car elles varient entre les mêmes limites [0, 3] et avec des matrices de transition presque égales. Ci-dessous, les résultats renvoyés par le GCBag.

$$\hat{A}_1 = \begin{pmatrix} 0.62 & 0.38 \\ 0.39 & 0.61 \end{pmatrix}; \hat{A}_2 = \begin{pmatrix} 0.41 & 0.59 \\ 0.58 & 0.42 \end{pmatrix}$$

$$\hat{\mu}_1 = \begin{pmatrix} 0 & 2.93 \end{pmatrix}; \hat{\mu}_2 = \begin{pmatrix} 0 & 3 \end{pmatrix}$$

$$\hat{V}_1 = \begin{pmatrix} 1 & 0.97 \end{pmatrix}; \hat{V}_2 = \begin{pmatrix} 1 & 0.99 \end{pmatrix}$$

Comme on peut le voir, pour des séries temporelles presque égales et générées à partir de deux HMMs différents, le GCBag peut détecter la différence entre elles et estimer correctement les paramètres.

Nous avons également effectué des tests pour évaluer le GCBag par rapport à d'autres algorithmes, lorsque les dimensions des ensembles de données changent. La Table 4.2 montre les résultats de ces tests. Les données sont générées avec les mêmes HMMs présentés ci-dessus, mais avec des dimensions différentes de T. Nous pouvons voir que le GCBag conserve un score *v-mesure* stable et meilleur que les autres algorithmes dans la plupart des cas.

Test 2. Sensibilité au bruit. Nous avons effectué une série de tests pour évaluer les performances du GCBag lorsqu'il traite des données bruyantes. Tout d'abord, 200 séries temporelles de longueurs 80 sont générées à partir des mêmes HMMs du test 1, puis des bruits sont ajoutés. Pour créer les bruits, nous générons 200 séquences avec une moyenne fixée à 1 et un écart-type *stdNoise* avec des valeurs différentes.

Les résultats de ces tests sont présentés dans la Table 4.3, et montrent que le GCBag, dans la

Paramètres		Résultats			
#Time Series	**Longueur**	**DTW+HC**	**DTW+HC+MHMMs**	**HMM+HC**	**GCBag**
80	80	0.02	0.58	0.32	**0.76**
100	80	0.18	**0.69**	0.24	0.6
200	80	0.19	0.69	0.11	**0.74**
400	80	0.1	0.56	0.07	**0.58**
800	80	0.1	0.58	0.07	**0.62**
1000	80	0.17	0.64	0.09	**0.71**
80	200	0.48	**1**	**1**	**1**
100	200	0.53	0.92	0.8	**0.95**
200	200	0.54	**0.92**	0.8	**0.92**
400	200	0.43	0.88	0.88	**0.91**
800	200	0.54	**0.92**	0.8	**0.92**
1000	200	0.58	0.92	0.86	**0.94**

Table 4.2: Comparaison de *v-mesure* du clustering des séries temporelles synthétiques. *plus le score est élevé, meilleure est la performance.*

Paramètres	Résultats			
stdNoise	**DTW+HC**	**DTW+HC+MHMMS**	**HMM+HC**	**GCBag**
0	0.19	0.69	0.11	**0.74**
1	0.02	**0.36**	0.17	0.35
1.15	0.04	0.33	0.18	**0.31**
1.2	0.01	0.24	0.18	**0.3**
1.25	0.013	0.22	0.09	**0.26**

Table 4.3: Comparaison de *v-mesure* du clustering des séries temporelles synthétiques contenant des bruits. *plus le score est élevé, meilleures sont les performances.*

plupart des cas, détecte mieux les clusters que les autres algorithmes, même si l'entrée contient des bruits.

Test 3. L'utilité du bagging. Pour illustrer l'utilité de la méthode de Bagging, nous avons suivi le score *v-mesure* à chaque itération du Bagging pendant le clustering et pour différentes valeurs de M. Le clustering dans une itération de Bagging est indépendant de celui de l'itération précédente, pour cela les courbes peuvent ne pas être linéaire. Nous avons généré 200 séries temporelles de longueurs 80 à partir de mêmes HMMs du test 1, et nous avons fixé M à {80, 120, 160} ce qui correspond à {40%, 60%, 80%} de $|T|$.

Le résultat affiché dans la Figure 4.2, montre que le GCBag trouve un résultat de bonne qualité, car il utilise la méthode de Bagging. Le GCBag parvient à trouver ces résultats même s'il détecte des clusters de faible qualité pendant son exécution. Pour être sûr d'obtenir le bon résultat, nous pouvons augmenter le nombre d'itérations du Bagging pour permettre à GCBag de visiter plus de solutions.

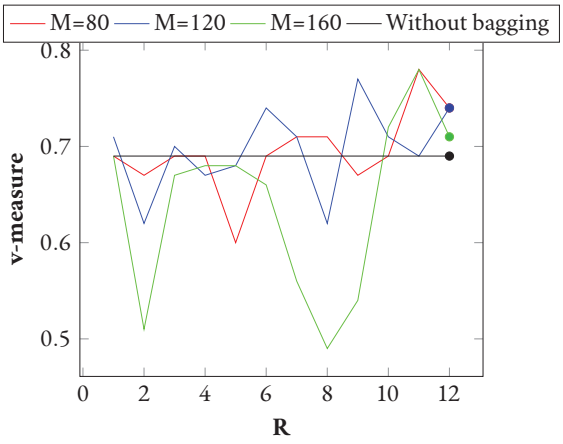


Figure 4.2: Les différentes valeurs de *v-measure* recueillies à chaque itération du Bagging. Le dernier *v-mesure* (à R=12) est la valeur du résultat renvoyée par le GCBag.

4.4.2 Résultats sur les données réelles

Nous avons également mené des expériences sur des ensembles de données réelles provenant de [22]. Ici, la mesure de la silhouette [73] est utilisée pour évaluer la qualité des clusters. Nous avons fixé M à $60\% \times |T|$ et R à 12. La Table 4.4 montre les résultats de cette expérimentation. Nous remarquons que sur un grand ensemble de données réelles, GCBag se comporte très bien par rapport aux autres algorithmes et découvre des clusters de meilleure qualité, ce qui répond à notre objectif initial.

Data Set	**Data description**			**Résultats**			
	#Time Series	**Longueur**	**#Classes**	**DTW+HC**	**DTW+HC+MHMMS**	**HMM+HC**	**GCBag**
ArrowHead	36	251	3	**0.29**	0.21	0.11	0.26
Wine	57	234	2	**0.4**	**0.4**	**0.4**	**0.4**
SonyAIBORobotSurface	27	65	2	**0.15**	0.13	0.13	0.1
ECG200	100	96	2	0.15	0.05	0.16	**0.19**
ProximalPhalanxOutlineCorrect	600	80	2	0.37	0.32	0.06	**0.4**
BME	30	128	3	0.36	0.36	0.22	**0.51**

Table 4.4: Comparaison de l'indice de silhouette du clustering sur des séries temporelles réelles. *plus le score est élevé, meilleures sont les performances.*

4.5 Conclusion

Dans ce chapitre, nous avons proposé un nouveau modéle de comportement appelé "comportement intéressant". Ce comportement est modélisé par des séries temporelles multi-variées et est extrait grâce à un nouveau framework appelé GCBag (Generative time series Clustering with Bagging), conçu pour construire un modèle descriptif pour les données des séries temporelles. Il est basé sur plusieurs techniques de séries temporelles et clustering (comme DTW, HC, Mixture of HMMs) encapsulées dans une méthode de Bagging. Les algorithmes classiques de clustering basés sur les HMMs appliquent DTW+HC sur l'ensemble des données, afin de créer un point de départ pour la formation des HMMs. Dans ces algorithmes, l'estimation des paramètres HMMs est très dépendante des résultats de DTW+HC, qui peuvent être de mauvaise qualité. Notre principal objectif est d'améliorer la qualité du clustering en fournissant

aux HMMs plusieurs points de départ pour l'estimation de leurs paramètres, en utilisant une méthode de Bagging.

Dans chaque itération de Bagging, GCBag commence par construire un clustering initial en extrayant un échantillon de l'ensemble des données d'entrée et en utilisant ensuite DTW+HC pour les clusteriser. Ensuite, les clusters résultants sont utilisés pour initialiser les HMMs avant leur ajustement aux données restantes. Chaque itération du Bagging génère une version indépendante des clusters. La version finale peut alors être construite en utilisant le représentant de chaque cluster de HMMs similaire. Pour cela, nous avons proposé une méthode appelée "Group-and-Merge" pour regrouper des clusters (HMMs) similaires et les fusionner afin d'obtenir un cluster représentant. Lorsqu'une version finale du clustering est construite, la matrice d'appartenance est construite en calculant pour chaque série temporelle et chaque HMM leur likelihoods.

Plusieurs expériences sont menées pour évaluer les résultats du GCBag. Elles mettent en évidence l'avantage d'utiliser la méthode de Bagging dans le clustering.

5

Application des algorithmes aux données e-commerce

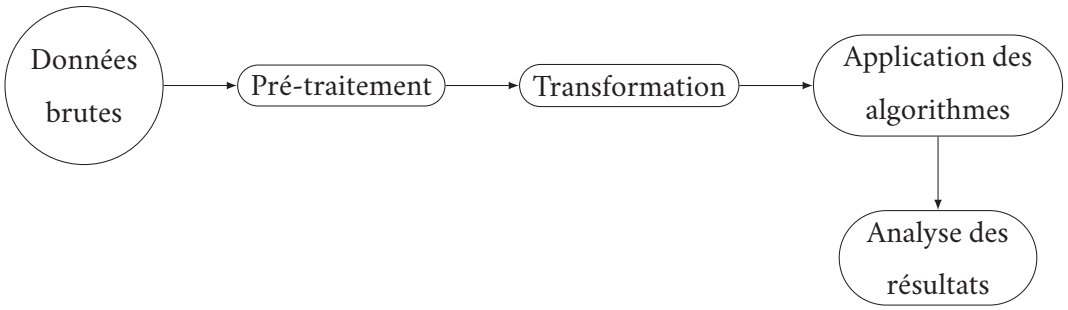
Dans ce chapitre, notre objectif est d'extraire les nouveaux modèles de comportements (fréquent et intéressant) proposés dans cette thèse à partir d'un vrai ensemble de données. Nous présentons les différents méthodes de préparation des données (nettoyages et transformations), l'application des algorithmes et l'analyse des résultats. Nous appliquons nos algorithmes sur un vrai ensemble de données e-commerce issues du site Google Merchandise et nous nous intéressons à l'analyse des résultats retournés. Ce chapitre représente une preuve de faisabilité de nos contributions. Les résultats montrent que les comportements détectés par nos algorithmes sont de grandes utilités et peuvent contribuer à une améliorations du site d'e-commerce.

Plan du chapitre

5.1 Introduction

Nous avons présenté, dans les chapitres précédents, nos deux nouveaux modèles de comportements. Ces deux modèles sont basés sur deux contributions. Dans la première contribution (chapitre. 3), nous étudions le comportement fréquent. Nous modélisons ce comportement par une nouvelle représentation des motifs et nous l'extrayons grâce à un nouvel algorithme de détection de motifs appelé "SEPM". Dans la deuxième contribution (chapitre. 4), nous étudions le comportement intéressant. Nous modélisons ce comportement par des clusters de séries temporelles multi-variées et nous l'extrayons grâce à un nouveau framework de clustering des séries temporelles appelé "GCBag".

Dans ce chapitre, nous menons une étude complète sur l'application des ces algorithmes aux vrais données e-commerce issue du site d'e-commerce Google Merchandise. Nous nous intéressons ici à l'analyse des résultats retournés par nos algorithmes. Dans la section 5.2, nous présentons en chiffre les données utilisées. Nous appliquons SEPM à ces données dans la section 5.3 et ensuite nous appliquons GCBag aux même données dans la section 5.4. Nous concluons dans la section 5.5.



5.2 Présentation des données

5.2.1 Source

Dans le cadre d'une compétition Kaggle [1], Google a mis en ligne des données contenant les traces d'utilisation de son site d'e-commerce du 01/08/2016 au 01/08/2017. Google propose sur son site d'e-commerce "Google Merchandise Store" [2] la vente des produits de marque Google comme des vêtements, des stylos, des cahiers, des mugs, etc. Les Figures A.1 et A.2 (dans l'annexe) montrent respectivement la page d'accueil du site Google Merchandise et la page du produit "Cloud Bundle".

Les visiteurs du site Google Merchandise peuvent :

1. Consulter les produits ou les catégories
2. Ajouter un produit au panier
3. Acheter le produit
4. Créer un compte

Chaque interaction de l'utilisateur avec le site est enregistrée en tant que trace dans "Google Analytics" [3]. Les traces contiennent des informations relatives aux sessions de chaque utilisateur ou visiteur. Nous connessons pour chaque sessions : son utilisateur (un identifiant unique s'il se connecte et temporaire s'il ne l'est pas), les pages consultées, le timestamp de chaque clic, les produits achetés, etc.

Les données sont disponibles sur bigquery cloud [4]. Elles sont stockées dans plusieurs tables et chaque table contient les traces d'une journée d'interaction avec le site. Un script python est développé pour télécharger ces données et les sauvegarder dans des fichiers csv (∼20G).

[1] https://www.kaggle.com/bigquery/google-analytics-sample
[2] https://www.googlemerchandisestore.com/
[3] https://analytics.google.com/
[4] https://bigquery.cloud.google.com/table/bigquery-public-data:google_analytics_sample.ga_sessions_20170801

5.2.2 Les données

Les données disponibles contiennent 903,653 sessions et 237 colonnes (features). Chaque session correspond à un seul parcours sur le site par un seul utilisateur. Elle est identifiée par un Id unique et par l'Id de son utilisateur. L'Id de l'utilisateur est temporaire pour la session si l'utilisateur n'est pas connecté, et il est fixe si l'utilisateur s'est connecté.

Nous avons séparées les données en 2 fichiers :

1. Le premier contient les profiles des sessions. Il contient des informations générales sur les sessions : l'id de l'utilisateur, l'id de la session, la durée de la session, le nombre de clics, avec ou sans achat, etc.

2. Le deuxième contient les clics durant les sessions. Nous trouvons dans ce fichier des informations comme : l'id de l'utilisateur, l'id de la session, les pages consultées, le numéro du clic, le timestamps du clic, etc.

La Table 5.1 présentent les valeurs globales des données.

Nombre d'utilisateurs	714,167
Nombre d'utilisateurs qui ont plus de 1 session	93,492
Nombre d'utilisateurs qui ont plus de 2 sessions	34,781
Nombre d'utilisateurs qui ont plus de 3 sessions	17,622
Nombre de sessions	903,653
Nombre de session avec achat	11,552
Nombre d'actions total	4,153,675

Table 5.1: La taille des données contenant les traces d'utilisations de Google Merchandise.

5.2.3 Pré-traitement

Cette étape consiste à identifier les caractéristiques les plus pertinentes pour notre étude et corriger (si besoin) les données inexactes. Cette étape est importante pour traiter seulement les données qui nous intéressent.

Le nettoyage des données consiste à :

1. Identifier les données essentielles et supprimer le reste
2. Éliminer les doublons
3. Éliminer les sessions contenant des valeurs très grandes et anormales

Nous trouvons plusieurs features qui sont à NULL. Cela est dû à plusieurs raisons : l'utilisateur n'accepte pas de partager ses données, Google ne veut pas diffuser ces informations, etc. Pour cela, les colonnes (caractéristiques/features) qui ne contiennent que des valeurs à NULL sont supprimées. Nous gardons que 77 colonnes (11 features quantitatives et 66 features qualitatives [binaires et catégorielles]).

Nous ne gardons que les traces d'interactions avec les produits. Nous avons supprimé par exemple tous les clics sur la page de connexion ou sur la page de contact.

Nous avons supprimé les sessions qui contiennent des valeurs anormales. Nous nous sommes intéressés aux valeurs des durées de la session et au nombre de clics dans la session. Leurs valeurs anormales peuvent être causé par des erreurs de calcul (session non fermé par l'utilisateur) ou des web crawlers.

5.2.4 Transformation

Les deux algorithmes (SEPM, GCBag) proposés utilisent des formats de données en entrée différents. Nous devons donc pour chaque algorithme adapter les données pour pouvoir les utiliser.

SEPM

L'algorithme SEPM prend en entrée un fichier contenant dans chaque ligne une séquence de produits consultés où chaque produit est couplé avec la durée (qui correspond à la durée passée sur la page du produit). Chaque ligne correspond à une seule session.

Pour construire ce fichier, nous avons extrait les produits de chaque session à partir des données. La durée d'une action A est égale à la différence entre le timestamp de l'action qui suit l'action A et le timestamp de A. Exemple : si un utilisateur a consulté le produit P1 à 19:22:00 et ensuite il a consulté le produit P2 à 19:23:00, la durée de consultation du produit P1 est égale à 19:23:00 - 19:22:00 = 1 minute.

La Figure 5.1 montre un exemple de la transformation des données au format d'entrée de SEPM.

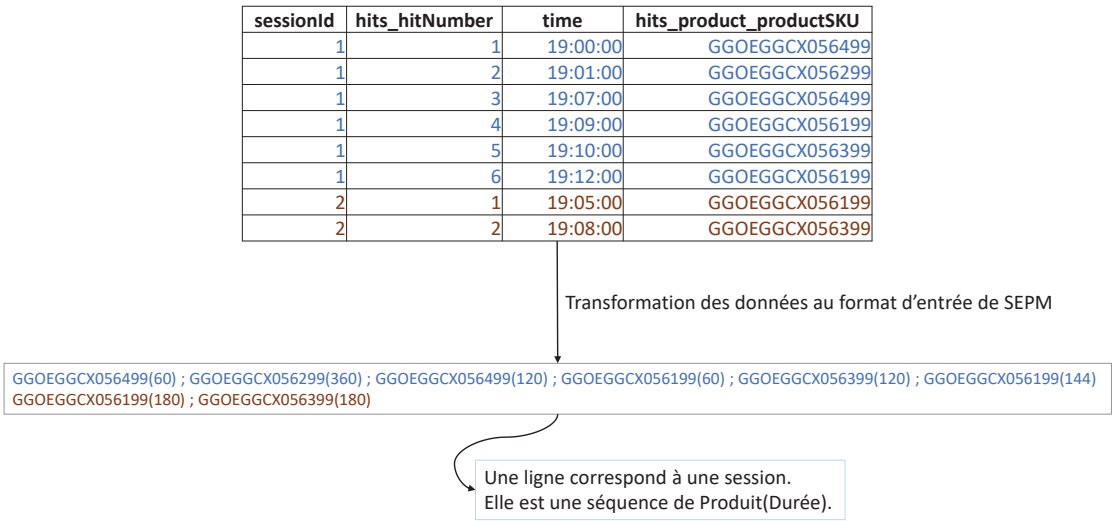


Figure 5.1: Un exemple de la transformation des données pour appliquer SEPM.

Nous rencontrons deux cas où la durée passée sur un produit ne peut pas être calculée. Le premier lorsque l'utilisateur consulte un seul produit dans la session. Dans ce cas, nous disposons du timestamp du produit mais pas le suivant. Pour ce cas, nous mettons la durée à 0. Le deuxième cas est rencontré avec la dernière durée passé sur le dernier produit consulté. Nous décidons de fixé cette durée à la moyenne des durées trouvées dans la session.

GCBag

Le framework GCBag prend en entrée un fichier contenant des séries temporelles où chaque série temporelle correspond à la séquence des sessions d'un utilisateur. Parmi les features, nous ne gardons que les données quantitatives qui sont :

1. **totals_hits :** nombre total d'appels durant la session
2. **totals_timeOnSite :** durée totale de la session (en secondes)

Nous avons aussi augmenter les informations des sessions en ajoutant les features suivant :

1. **nb_product :** nombre de produit visualisé pendant la session
2. **nb_category :** nombre de catégorie de produit visualisé pendant la session
3. **mean_time_per_page :** le temps moyen passé par page
4. **time_before_first_click :** le temps passé avant le premier clic
5. **time_on_purchase_page :** le temps passé sur la page de paiement

Une série temporelle est créée pour chaque utilisateur. La série regroupe par ordre croissant temporel la séquence des sessions des utilisateurs. Après la transformation, nous obtenons une matrice contenant en colonne les features et en ligne les sessions regrouper par utilisateur.

La Figure 5.2 montre un exemple de la matrice obtenue et à utiliser en entrée pour GCBag.

5.3 Application de SEPM

Une fois que les données sont nettoyées et transformées, nous pouvons appliquer l'algorithme SEPM. Dans cette section, nous menons deux types d'expériences : la première utile pour étudier la variation du nombre de patterns par catégorie quand les paramètres de SEPM changent et la deuxième se concentre sur les patterns retournés et leurs relations avec le comportement.

	visitId	visitStartTime	date	totals_visits	fullVisitorId	totals_hits	totals_pageviews	totals_timeOnSite	totals_bounces	totals_transactions	totals_newVisits	totals_totalTransactionRevenue	nb_product	nb_category	mean_time_per_page	time_before_first_click	time_on_purchase_page
1	1489888808	1489888808	20170318	1	usr1	8	4	77.0	0	0.0	0	0	4.0	1.0	48.91	5.871	
2	1489882668	1489882668	20170318	2	usr1	15	14	648.0	0	0.0	1.0	0	33.0	2.0	193.46	19.297	
3	1490148213	1490148213	20170321	3	usr1	28	22	853.0	0	1.0	0	20990000.0	38.0	4.0	317.98	4.358	396.23
4	1499709643	1499709643	20170710	1	usr2	17	17	626.0	0	1.0	1.0	43980000.0	49.0	5.0	364.53	26.382	175.54
5	1499964450	1499964450	20170713	2	usr2	2	2	26.0	0	0.0	0	0	12.0	1.0	13.032	26.064	

Figure 5.2: Un exemple de la matrice obtenue après la transformation des données pour appliquer GCBag.

Première expérience.

Dans un premier temps, nous étudions l'impact de chaque paramètre sur la quantité de patterns par catégorie retournées par l'algorithme. Cette expérience nous aide à savoir déterminer les paramètres de SEPM selon les besoins de l'analyse.

Nous trouvons dans la Table 5.2 les paramètres de SEPM pour chaque test mené. Dans les tests [1-3], nous varions le support minimal entre 30 et 100, ensuite dans les tests [4-6] nous varions le le gap minimale (entre deux actions consécutives) entre 2 et 7. Nous varions aussi la valeur de maxVar entre 4 et 8 dans les tests [7-9], et finalement, nous varions le maxSD entre 10 et 30 dans les tests [10-12]. Dans tous les tests, nous ne traitons que les séquences qui ont plus de 6 actions dans la base de données d'entrée et nous ne sauvegardons que les patterns qui ont plus de 3 éléments. Les résultats de ces tests sont affichés dans les Figures 5.3, 5.4, 5.5, et 5.6.

	SEPM		**Catégorisation**	
Test	**minSupp**	**minGap**	**maxVar**	**maxSD**
1	100	1	10	40
2	50	1	10	40
3	30	1	10	40
4	50	2	10	40
5	50	5	10	40
6	50	10	10	40
7	50	1	4	40
8	50	1	6	40
9	50	1	8	40
10	50	1	10	10
11	50	1	10	20
12	50	1	10	30

Table 5.2: Les paramètres de SEPM.

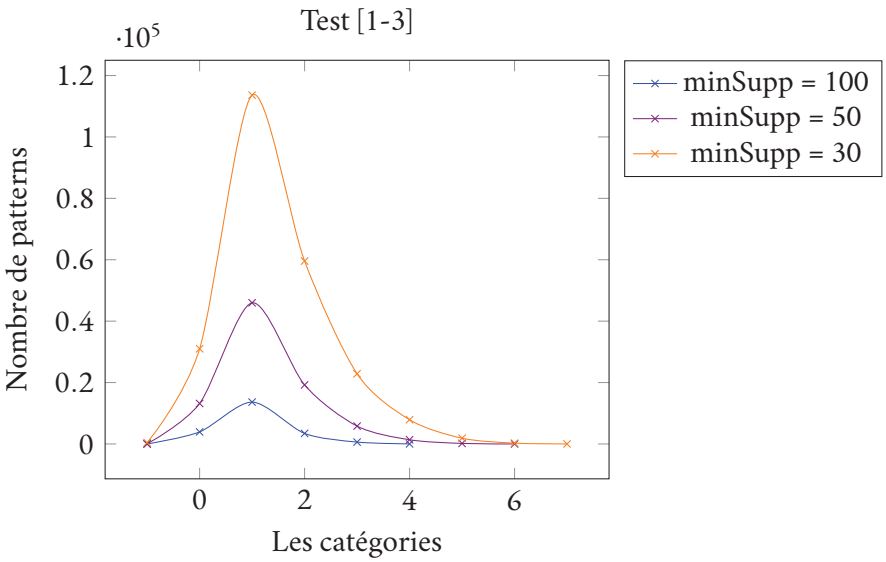


Figure 5.3: L'effet de varier le support minimal. *La catégorie -1 contient les patterns anormaux.*

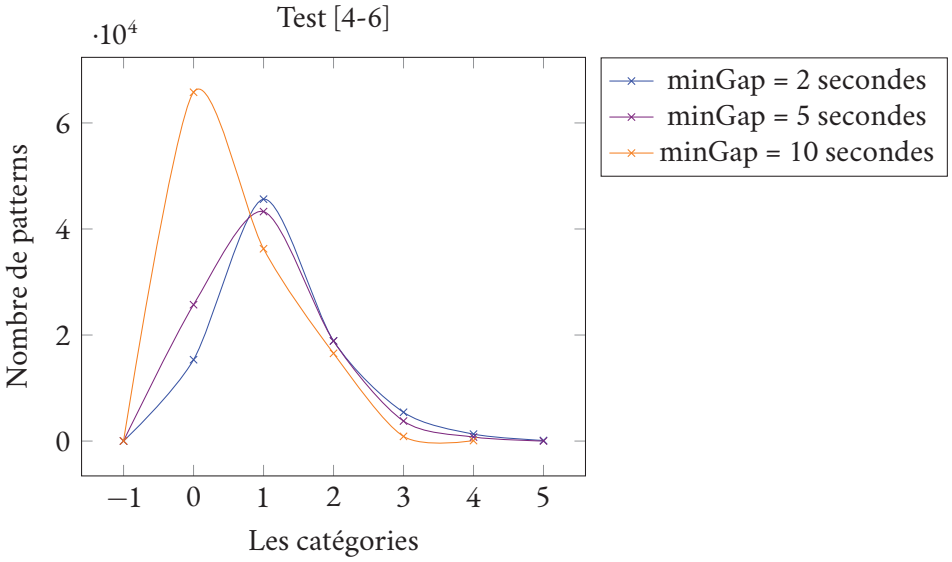


Figure 5.4: L'effet de varier le gap minimal. *La catégorie -1 contient les patterns anormaux.*

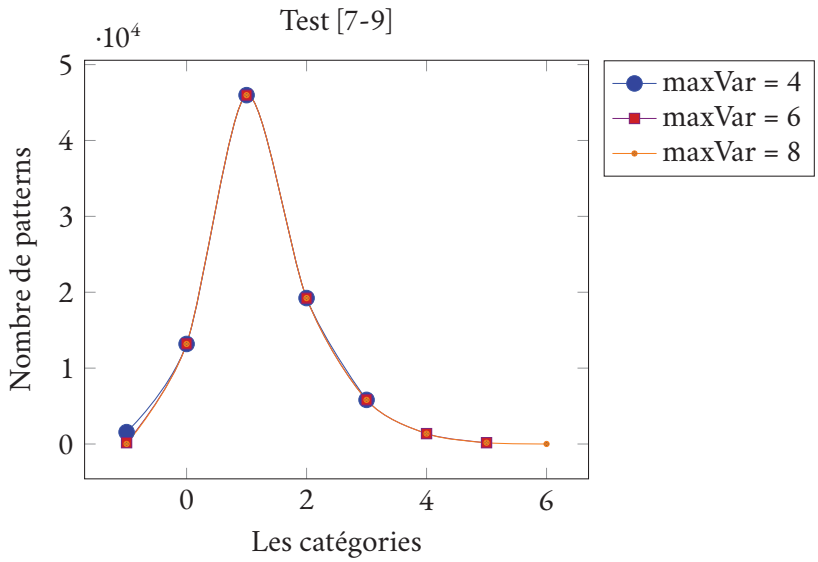


Figure 5.5: L'effet de varier le max variation. *La catégorie -1 contient les patterns anormaux.*

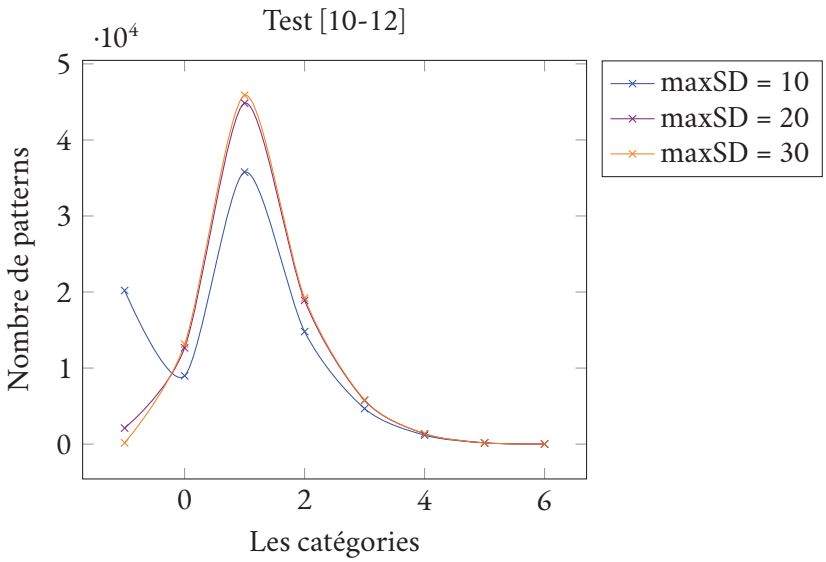


Figure 5.6: L'effet de varier le max sd. *La catégorie -1 contient les patterns anormaux.*

D'après la Figure 5.3, nous remarquons que lorsque le support minimal diminue, le nombre

de patterns augmente plus dans la catégorie 1 (où le nombre de variation est égal à 1) que dans les autres. En effet, il est tout à fait normal de trouver plus de patterns quand le support minimal diminue, mais grâce à notre algorithme de catégorisation, nous avons pu détecter la nature de ces nouveaux patterns. Nous trouvons que la majorité des patterns avec un support minimal faible, contient une seule variation de durée.

D'après la Figure 5.4, nous notons que lorsque le gap minimal augmente, nous retrouvons plus de patterns dans la catégorie 0 et moins dans les autres. Le nombre de patterns devient important dans la catégorie 0 lorsqu'on atteint un min gap de 10 secondes. Nous constatons que la majorité des patterns ont des durées qui ne dépassent pas les 10 secondes.

D'après la Figure 5.5, nous remarquons que la valeur de maxVar n'a pas un grand impact sur le nombre de patterns par catégorie. Quand le maxVar augmente, le nombre de patterns dans la catégorie -1 (patterns anormaux) diminue.

D'après la Figure 5.6, nous remarquons que lorsque le maxSD augmente les patterns augmentent plus dans la catégorie 1. Donc la catégorie 1 contient des patterns avec un écart type plus grand que ceux dans les autres catégories.

Selon l'objectif de l'analyse, nous devons trouver l'équilibre entre ces paramètres. Dans notre cas, le support minimal nous aide à découvrir plus de patterns mais majoritairement concentrés dans la catégorie 1 qui représente une navigation avec une seule variation. Le gap minimal nous aide à équilibrer la quantité de patterns dans les catégories selon les durées. Le maxVar et maxSD jouent plus le rôle des filtres. Ils nous aident à définir les valeurs de la variation et de l'écart-type maximaux avant de considérer qu'un pattern est anormal.

Deuxième expérience.

Dans cette deuxième expérience, nous nous concentrons sur la relation entre les catégories trouvées et le comportement fréquent. Pour cela nous prenons les résultats du test 2 de l'expérience précédente. Nous analysons les trois catégories 0, 1 et 6 (que nous supposons plus importantes).

La catégories 0. Cette catégorie de patterns regroupe les patterns qui ne contiennent pas de changement dans le sens de variation de leurs durées. La Figure 5.7 contient un exemple de 4 patterns trouvés dans cette catégorie. Il s'agit des patterns présent dans des sessions où les utilisateurs passent de plus en plus ou de moins en moins de temps sur les produits consultés.

Figure 5.7: Exemple de patterns dans la catégorie 0. *La durée est en seconde.*

La catégories 1. Nous retrouvons dans cette catégorie les patterns où les utilisateurs commencent par passer de moins en moins de temps sur les produits et ensuite commencent à passer de plus en plus de temps. Ou l'inverse, les utilisateurs passent de plus en plus de temps sur les produits et ensuite passent de moins en moins de temps sur les produits. La Figure 5.8 contient un exemple de patterns trouvés dans cette catégorie. Ces patterns correspond à un changement du comportement durant la session. Quand le sens de la variation passe de "-" à "+", nous pouvons dire que l'utilisateur devient intéressés par les produits qu'il consulte. Et l'inverse, quand le sens de la variation passe de "+" à "-", nous pouvons dire qu'il est de moins en moins intéressé.

Figure 5.8: Exemple de patterns dans la catégorie 1. *La durée est en seconde.*

La catégories 6. Cette catégorie contient des patterns dont le sens de la variation change plusieurs fois au cours de la session. Parfois les utilisateurs passent moins de temps sur les produits et parfois plus de temps. La Figure 5.9 contient un exemple de patterns trouvés dans cette catégorie. Ces patterns représentent plusieurs types de comportement :

1. un comportement comparateur : l'utilisateur consulte différents produits pour les comparer. Ce comportement contient dans la même séquence le même produit plusieurs fois ou des durées de consultation importante. Souvent ces produits consultés appartiennent au même catalogue de produit.

2. un comportement aléatoire : l'utilisateur consulte aléatoirement des produits de différent catalogues et avec des petites durées de consultation.

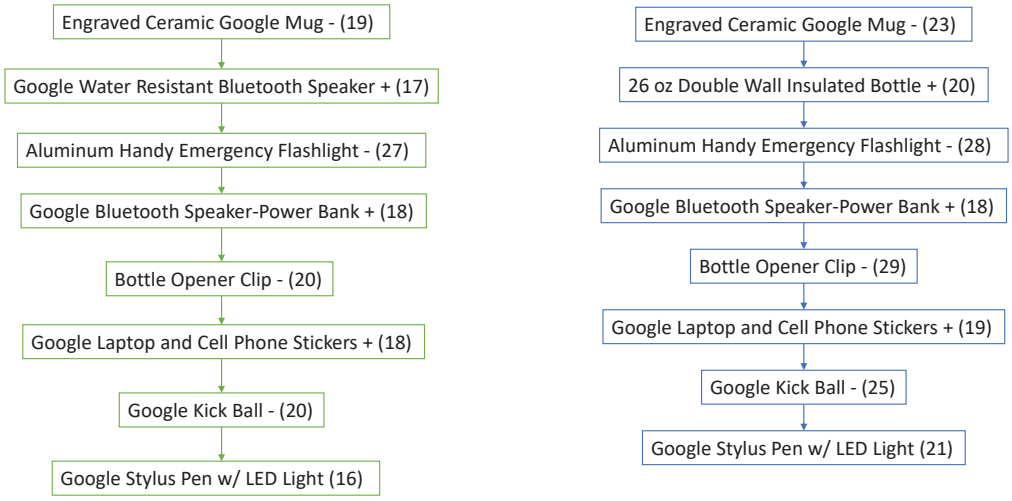


Figure 5.9: Exemple de patterns dans la catégorie 6. *La durée est en seconde.*

5.4 Application de GCBag

Un des comportements intéressants à étudier est le comportement des acheteurs. Il est utile de savoir quels sont les comportements des acheteurs pour pouvoir les fidéliser. Pour cela, nous filtrons les séries temporelles pour n'en garder que celles qui contiennent au moins un achat.

Une fois le dataset des séries temporelles est construit, nous appliquons GCBag à ce dataset avec les paramètres suivant :

1. **Nombre de cluster** = 5
2. **Nombre de composant dans chaque HMM** = 3
3. **La taille de l'échantillon** = 60% du dataset = 0.6*10023 = 6014
4. **Nombre d'itération du bagging** = 20

Pour définir le nombre de cluster, nous avons exécuté GCBag avec plusieurs valeurs du nombre de cluster et nous avons calculé pour chaque exécution la silhouette du clustering résultant. Nous avons trouvés qu'avec 5 clusters, la silhouette est la meilleure et pour cela nous l'utilisons dans la suite.

La Table 5.3, nous montre quelques statistiques sur le nombre de sessions dans les séries temporelles par cluster.

Cluster	**Nombre de séries**	**Nombre minimal de sessions**	**Nombre maximal de sessions**	**La médian (la valeur à la position 50%)**
0	813	3	41	8
1	4284	1	11	2
2	151	6	278	19
3	3194	1	3	1
4	1581	3	19	5

Table 5.3: Quelques statistiques sur le nombre de sessions dans les séries temporelles par cluster.

Les Figures 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, et 5.16, nous montrent l'évolution des features par clusters. Chaque courbe représente l'évolution de la moyenne de feature dans le cluster. Nous gardons que les 30 dernières sessions dans les séries temporelles qui en contiennent plus.

Cluster 0.

Le cluster 0 contient les utilisateurs qui ont un nombre de sessions moyen avant l'achat. Nous remarquons que les utilisateurs dans ce cluster, passent plus de temps avant le premier clique et aussi plus de temps sur le site quand ils dépassent les 15 sessions. Les utilisateurs qui ont moins de 15 sessions gardent un comportement presque similaires (pas de changement important des caractéristiques) avant l'achat.

Cluster 1.

Ce cluster contient des séries temporelles courtes. Nous remarquons que les utilisateurs dans ce cluster qui n'achètent pas après 4 sessions passent plus de temps sur le site et consultent plus de produit de différent catégories avant d'acheter.

Cluster 2.

Le cluster 2 contient les séries temporelles les plus longues, c'est à dire les utilisateurs qui visitent plusieurs fois le site avant d'acheter. Une baisse lègre est remarquée dans les features de leurs sessions. Mais en générale, il s'agit des utilisateurs qui ne sont pas sûr d'acheter ou qui aiment comparer les prix avec d'autres sites.

Cluster 3.

Le cluster 3 regroupent les utilisateurs qui sont les plus rapides dans l'achat. Nous remarquons qu'il y a une augmentation dans le temps avant le premier achat. Donc, ces utilisateurs consultent directement les produits qui les intéressent, et achètent directement quand ils le trouvent.

Cluster 4.

Le cluster 4 regroupe les utilisateurs qui ont un nombre de sessions moyen avant l'achat. Lorsque ces utilisateurs n'achètent pas au bout de 12 sessions, ils ont tendance à passer plus de temps sur le site pour parcourir d'autres produits avant l'achat final. Nous remarquons aussi que plus les utilisateurs de ce cluster ont de sessions, moins ils passent de temps sur la page d'achat. Ces utilisateurs passent à l'achat quand ils sont sûr d'acheter.

5.5 Conclusion

Nous avons pu dans ce chapitre mener une étude détaillé sur l'application de nos algorithmes sur des données réelles. Au début, nous avons expliqué comment les données brutes sont préparées et transformées avant d'être utilisées. Ensuite, nous avons appliqué SEPM pour extraire les différents comportements fréquents modélisés par des patterns fréquents, séquentiels et temporels. Aussi, nous avons montré comment les paramètres de SEPM peuvent influencer sur la quantité des patterns par catégorie. Une étape utile pour fixer les valeurs des paramètres. Aussi, nous avons interprété quelques catégories de patterns. Finalement, nous avons appliquer GCBag sur les séries temporelles qui contiennent au moins un achat pour extraire des comportements intéressants des acheteurs. Nous avons pu identifier plusieurs comportement d'achats qui sont utiles pour pouvoir fidéliser les acheteurs.

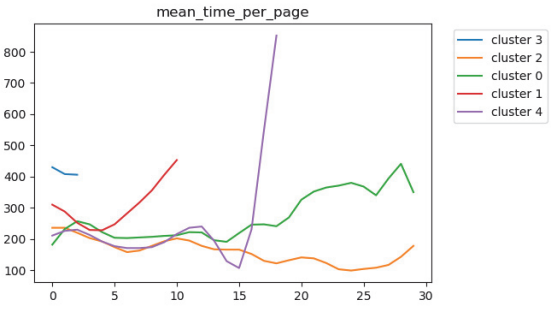


Figure 5.10: Évolution de la moyenne de temps par page.

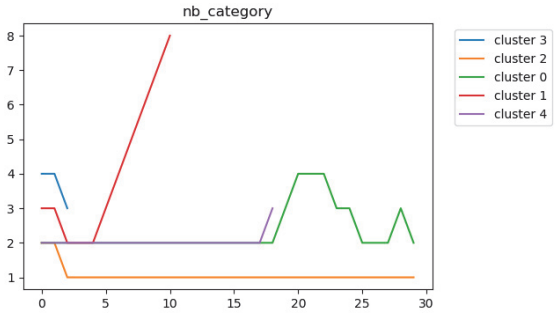


Figure 5.11: Évolution du nombre de catégorie.

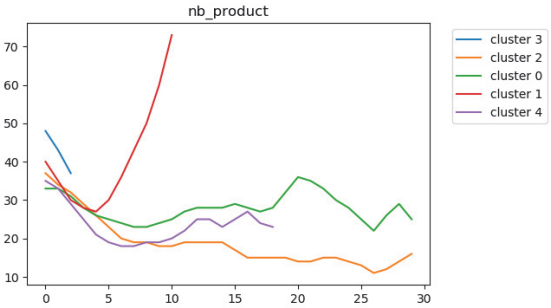


Figure 5.12: Évolution du nombre des produits visualisés.

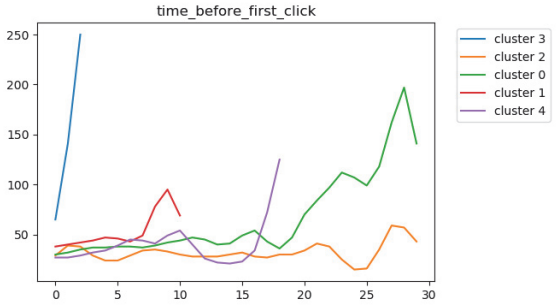


Figure 5.13: Évolution du temps avant le premier clique.

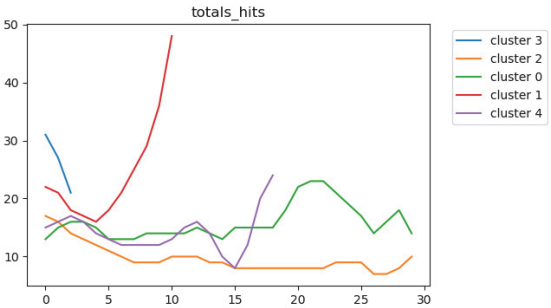


Figure 5.14: Évolution du nombre de cliques.

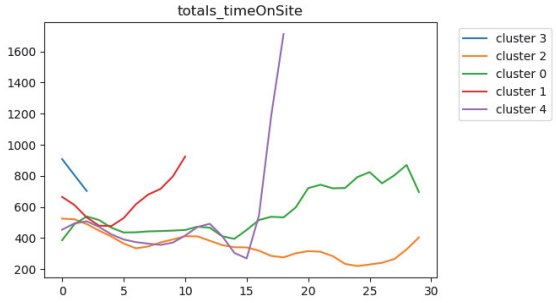


Figure 5.15: Évolution du temps sur le site.

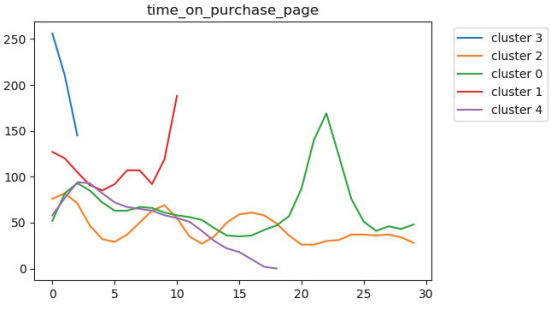


Figure 5.16: Évolution du temps sur la page d'achat.

6

Conclusion et perspectives

Depuis quelques années, le commerce électronique attire de plus en plus de client. Aujourd'hui, nous pouvons acheter n'importe quel produit, depuis n'importe quel pays à n'importe quelle heure. Il est donc nécessaire d'étudier comment les utilisateurs naviguent sur le site et achètent leurs produits pour que le commerçant puissent adapter leurs sites web par rapport aux besoins de leurs clients. Pour pouvoir analyser les comportements des clients, nous avons besoins de leurs traces laissées sur le site web pendant leurs navigations et qui sont collectées à travers des outils de traçage.

Les actions des utilisateurs sont très importantes, mais la véritable valeur ajoutée est créée par les algorithmes qui analysent correctement et parviennent à certaines conclusions con-

tribuant à améliorer le produit ou le service.

Pour répondre à ce besoin, nous avons étudié, dans cette thèse, le comportement des utilisateurs sur les sites de marchands en ligne. Nous nous sommes intéressés à deux types de comportements : le fréquent et l'intéressant.

Dans un premier temps, nous avons étudié le comportement fréquent en le modélisant par des patterns fréquents, séquentiels et temporels. D'abord, nous avons augmenter la représentation des patterns en ajoutant les durées et leurs sens de variations. En effet, la durée passée par chaque utilisateur sur un produit peut être utilisée comme un indice d'intérêt pour le produit. En outre, sa variation dans la séquence est importante, et peut nous aider à savoir si l'utilisateur est de plus en plus intéressé ou l'inverse. Ensuite, nous avons proposer un nouvel algorithme SEPM (*Sequential Event Pattern Mining*), qui utilise la représentation augmenté des séquences et retourne les patterns temporels qui sont des séquences de couple étiquette-durée. Nous avons mené plusieurs expériences sur des données synthétiques et réelles pour prouver les performances de l'algorithme et la qualité des patterns temporels.

Dans un deuxième temps, nous avons étudié les comportements intéressants des acheteurs. Nous modélisons ce comportement par des clusters de séries temporelles multi-variées. Une série temporelle représente la séquence de sessions d'un utilisateur et chaque session contient plusieurs informations sur la navigation de l'utilisateur pendant son parcours sur le site web. Pour extraire les groupes d'utilisateurs qui se comportement de la même manière, nous avons proposé le framework GCBag (*Generative time series Clustering with Bagging*) qui prend en entrée les séries temporelles et renvoie en sortie leurs répartitions dans les clusters. En tant que framework génératif, GCBag combine la puissance de plusieurs techniques conçues pour les séries temporelles comme DTW (*la Dynamic Time Warping*) pour mesurer les distances entre les séries temporelles et le HMM (*Hidden Markov Model*) pour modéliser les clusters. L'originalité du GCBag réside dans l'utilisation du *Bagging* lors du clustering dans le but d'améliorer la qualité des clusters retournés. Plusieurs expériences sont menées pour démontrer l'efficacité du GCBag par rapport aux algorithmes de clustering des séries temporelles

existantes, tant sur des données synthétiques que sur des séries temporelles réelles.

Finalement, nous avons appliqué nos deux algorithmes sur des données réelles issues du site d'e-commerce "Google Merchandise Store". Nous avons expliqué comment à partir des données brutes, nous préparons et transformons les données pour pouvoir appliquer nos algorithmes et comment nous pouvons analyser les résultats pour trouver les différents types de comportements.

Plusieurs pistes d'amélioration sont identifié pour les futurs travaux. Nous citons ci-dessous les plus intéressants.

1. **SEPM**.

   (a) En plus du sens de la variation de la durée, nous allons étudier la valeur de la durée et voir comment elle varie d'un utilisateur à l'autre. Dans notre proposition actuelle, nous prenons compte que de la variation de la durée mais nous ignorons la valeur de la durée. Il sera intéressant d'inclure cette valeur dans le processus de minage.

   (b) Adapter l'algorithme à l'architecture MapReduce pour pouvoir traiter des données à grande échelle. L'architecture MapReduce permet de manipuler de grandes quantités de données en les distribuant dans un cluster de machines pour être traitées comme dans [30, 18, 100].

2. **GCBag**.

   (a) Intégrer plus de métriques de calculs de similarité (en plus de la DTW) et plus de modèle probabiliste (en plus du HMM) dans le Bagging à fin d'améliorer les qualités des résultats et de le rendre plus génératives.

   (b) Proposer une version d'auto-apprentissage qui n'a pas besoin de paramètres en entrée. L'algorithme sera capable en toute autonomie de trouver les différents paramètres comme le nombre de cluster, le nombre d'itération du Bagging, la taille de l'échantillon, etc.

3. Fusionner les résultats des deux approches SEPM et GCBag pour extraire plus de comportements.

4. Modéliser et étudier le comportement de l'hésitant. Ce comportement est très utile aussi pour les commerçants. Il permet de savoir quand l'utilisateur a-t-il été proche de l'achat ou quand l'a-t-il abandonné. Ainsi, ils savent quel est le moment opportun pour contacter le client et quels sont les arguments à utiliser pour le convaincre comme le prix, le temps de livraison, le mode d'utilisation du produit, etc.

A

Annexe

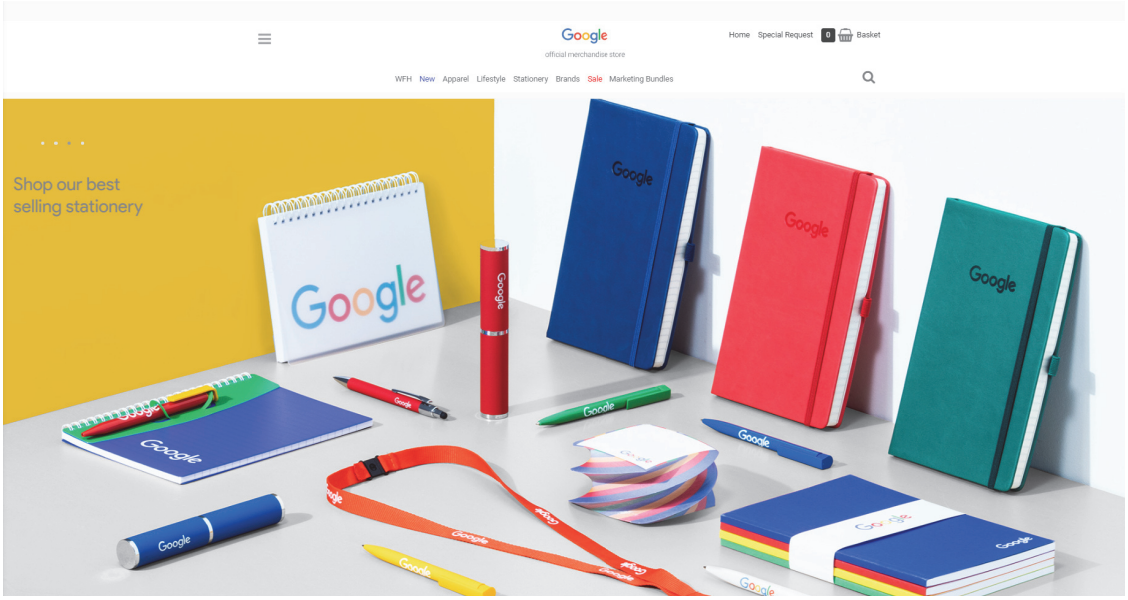


Figure A.1: La page d'accueil du site d'e-commerce Google Merchandise.

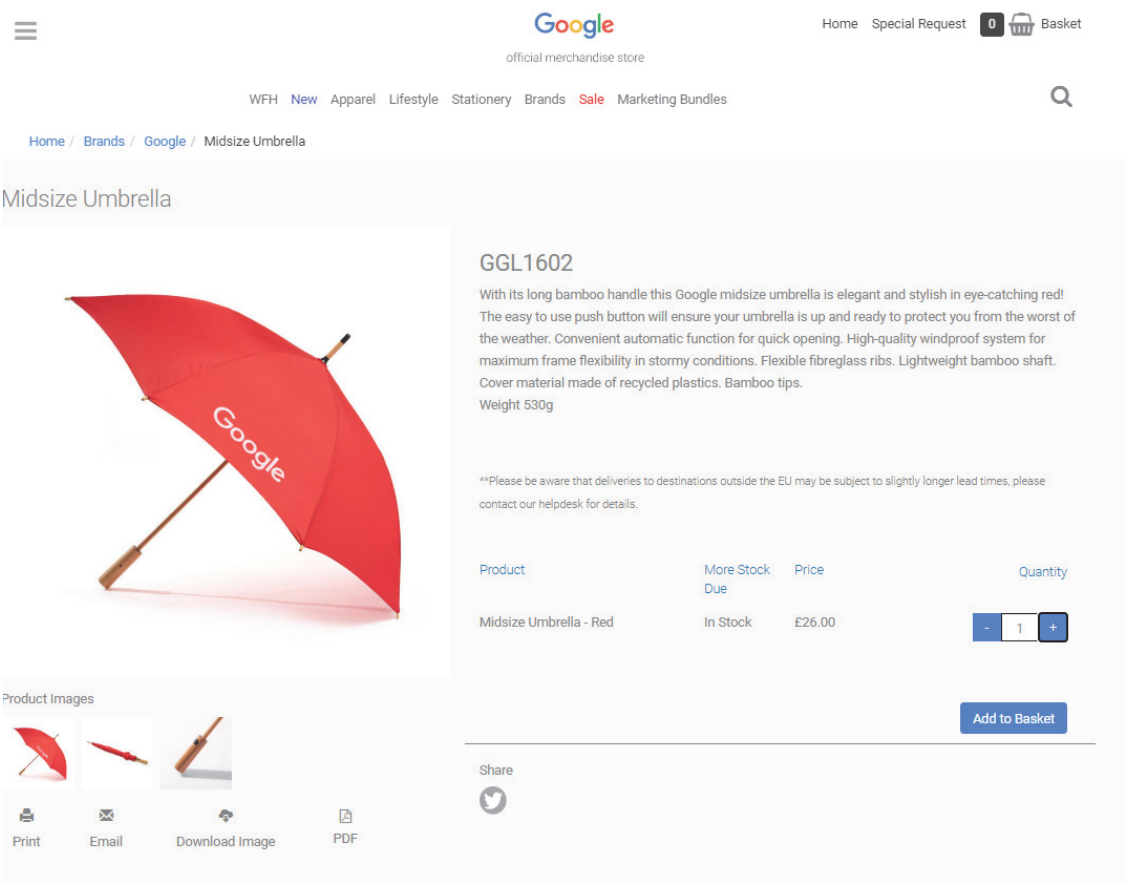


Figure A.2: La page du produit ”Cloud Bundle” disponible sur le site Google Merchandise.

Références

[1] Aghabozorgi, S. and Teh, Y. W. (2014). Stock market co-movement assessment using a three-phase clustering method. *Expert Systems with Applications*, 41(4):1301–1314.

[2] Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the eleventh international conference on data engineering*, pages 3–14. IEEE.

[3] Alborzi, M. and Khanbabaei, M. (2016). Using data mining and neural networks techniques to propose a new hybrid customer behaviour analysis and credit scoring model in banking services based on a developed rfm analysis method. *International Journal of Business Information Systems*, 23(1):1–22.

[4] Alibaba (2018). (dataset) user behavior data from taobao for recommendation. In *(URL) https://tianchi.aliyun.com/dataset/dataDetail?dataId=649*.

[5] Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

[6] Almuhisen, F., Durand, N., and Quafafou, M. (2018). Sequential formal concepts over time for trajectory analysis. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 598–603. IEEE.

[7] Alon, J., Sclaroff, S., Kollios, G., and Pavlovic, V. (2003). Discovering clusters in motion time-series data. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages I–I. IEEE.

[8] Alzahrani, M. Y. and Mazarbhuiya, F. A. (2016). Discovering sequential patterns from medical datasets. In *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 70–74. IEEE.

[9] Ansari, A. and Riasi, A. (2016). Customer clustering using a combination of fuzzy c-means and genetic algorithms. *International Journal of Business and Management*, 11(7):59.

[10] Asadi, N., Mirzaei, A., and Haghshenas, E. (2016). Creating discriminative models for time series classification and clustering by hmm ensembles. *IEEE Transactions on Cybernetics*, 46(12):2899–2910.

[11] Ayres, J., Flannick, J., Gehrke, J., and Yiu, T. (2002). Sequential pattern mining using a bitmap representation. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 429–435.

[12] Ben-Shimon, D., Tsikinovsky, A., Friedmann, M., Shapira, B., Rokach, L., and Hoerle, J. (2015). Recsys challenge 2015 and the yoochoose dataset. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 357–358.

[13] Benkabou, S.-E. (2018). *Outlier detection for time series data : application to tyre data*. Theses, Université de Lyon.

[14] Benkabou, S.-E., Benabdeslem, K., and Canitia, B. (2017). ℓ 2-type regularization-based unsupervised anomaly detection from temporal data. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2354–2361. IEEE.

[15] Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

[16] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

[17] Cai, L., Thornhill, N. F., Kuenzel, S., and Pal, B. C. (2018). Wide-area monitoring of power systems using principal component analysis and k-nearest neighbor analysis. *IEEE Transactions on Power Systems*, 33(5):4913–4923.

[18] Chen, C.-C., Shuai, H.-H., and Chen, M.-S. (2017). Distributed and scalable sequential pattern mining through stream processing. *Knowledge and Information Systems*, 53(2):365–390.

[19] Chen, L. and Ng, R. (2004). On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 792–803.

[20] Chen, L., Özsu, M. T., and Oria, V. (2005). Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 491–502.

[21] CIKM (2016). (dataset) cikm cup 2016 track 2: Personalized e-commerce search challenge. In *(URL) https://competitions.codalab.org/competitions/11161*.

[22] Dau, H. A., Bagnall, A., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., and Keogh, E. (2019). The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305.

[23] Dursun, A. and Caber, M. (2016). Using data mining techniques for profiling profitable hotel customers: An application of rfm analysis. *Tourism management perspectives*, 18:153–160.

[24] Esling, P. and Agon, C. (2012). Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):1–34.

[25] Fournier-Viger, P., Gomariz, A., Campos, M., and Thomas, R. (2014a). Fast vertical mining of sequential patterns using co-occurrence information. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 40–52. Springer.

[26] Fournier-Viger, P., Lin, J. C.-W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., and Lam, H. T. (2016). The spmf open-source data mining library version 2. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 36–40. Springer.

[27] Fournier-Viger, P., Lin, J. C.-W., Kiran, R. U., Koh, Y. S., and Thomas, R. (2017). A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1(1):54–77.

[28] Fournier-Viger, P., Wu, C.-W., Gomariz, A., and Tseng, V. S. (2014b). Vmsp: Efficient vertical mining of maximal sequential patterns. In *Canadian conference on artificial intelligence*, pages 83–94. Springer.

[29] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

[30] Gan, W., Lin, J. C.-W., Fournier-Viger, P., Chao, H.-C., and Yu, P. S. (2019). A survey of parallel sequential pattern mining. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(3):1–34.

[31] García-Hernández, R. A., Martínez-Trinidad, J. F., and Carrasco-Ochoa, J. A. (2006). A new algorithm for fast discovery of maximal sequential patterns in a document collection. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 514–523. Springer.

[32] Ghassempour, S., Girosi, F., and Maeder, A. (2014). Clustering multivariate time series using hidden markov models. *International journal of environmental research and public health*, 11(3):2741–2763.

[33] Gomariz, A., Campos, M., Marin, R., and Goethals, B. (2013). Clasp: An efficient algorithm for mining frequent closed sequences. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 50–61. Springer.

[34] Górecki, T. and Łuczak, M. (2013). Using derivatives in time series classification. *Data Mining and Knowledge Discovery*, 26(2):310–331.

[35] Gupta, L., Molfese, D. L., Tammana, R., and Simos, P. G. (1996). Nonlinear alignment and averaging for estimating the evoked potential. *IEEE transactions on biomedical engineering*, 43(4):348–356.

[36] Hu, J., Ray, B., and Han, L. (2006). An interweaved hmm/dtw approach to robust time series clustering. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 145–148. IEEE.

[37] Huaulmé, A., Voros, S., Riffaud, L., Forestier, G., Moreau-Gaudry, A., and Jannin, P. (2017). Distinguishing surgical behavior by sequential pattern discovery. *Journal of biomedical informatics*, 67:34–41.

[38] Jagan, S. and Rajagopalan, S. (2015). A survey on web personalization of web usage mining. *International Research Journal of Engineering and Technology*, 2(1):6–12.

[39] Jeong, Y.-S., Jeong, M. K., and Omitaomu, O. A. (2011). Weighted dynamic time warping for time series classification. *Pattern recognition*, 44(9):2231–2240.

[40] Jia, R., Li, R., Yu, M., and Wang, S. (2017). E-commerce purchase prediction approach by user behavior data. In *2017 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pages 1–5. IEEE.

[41] Kim, J. K., Song, H. S., Kim, T. S., and Kim, H. K. (2005). Detecting the change of customer behavior based on decision tree analysis. *Expert Systems*, 22(4):193–205.

[42] Kontostathis, A., Galitsky, L. M., Pottenger, W. M., Roy, S., and Phelps, D. J. (2004). A survey of emerging trend detection in textual data mining. In *Survey of text mining*, pages 185–224. Springer.

[43] Lee, J.-G., Han, J., and Whang, K.-Y. (2007). Trajectory clustering: a partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 593–604.

[44] Li, H. (2014). Asynchronism-based principal component analysis for time series data mining. *Expert systems with applications*, 41(6):2842–2850.

[45] Li, H. (2016). Accurate and efficient classification based on common principal components analysis for multivariate time series. *Neurocomputing*, 171:744–753.

[46] Li, H. (2019). Multivariate time series clustering based on common principal component analysis. *Neurocomputing*, 349:239–247.

[47] Li, M., Chen, X., Li, X., Ma, B., and Vitányi, P. M. (2004). The similarity metric. *IEEE transactions on Information Theory*, 50(12):3250–3264.

[48] Li, Z.-X., Guo, J.-S., Hui, X.-B., and Song, F.-F. (2013). Dimension reduction method for multivariate time series based on common principal component. *Control and Decision*, 28(4):531–536.

[49] Liao, T. W. (2005). Clustering of time series data—a survey. *Pattern recognition*, 38(11):1857–1874.

[50] Liao, V. C.-C. and Chen, M.-S. (2014). Dfsp: a depth-first spelling algorithm for sequential pattern mining of biological sequences. *Knowledge and information systems*, 38(3):623–639.

[51] Lin, N. P., Hao, W.-H., Chen, H.-J., Chueh, H.-E., Chang, C.-I., et al. (2008). Fast mining of closed sequential patterns. *WSEAS Transactions on Computers*, 7(3).

[52] Ma, Q., Zheng, J., Li, S., and Cottrell, G. W. (2019). Learning representations for time series clustering. In *Advances in Neural Information Processing Systems*, pages 3776–3786.

[53] Mabroukeh, N. R. and Ezeife, C. I. (2010). A taxonomy of sequential pattern mining algorithms. *ACM Computing Surveys (CSUR)*, 43(1):1–41.

[54] Madiraju, N. S., Sadat, S. M., Fisher, D., and Karimabadi, H. (2018). Deep temporal clustering: Fully unsupervised learning of time-domain features. *arXiv preprint arXiv:1802.01059*.

[55] Marteau, P.-F. (2008). Time warp edit distance with stiffness adjustment for time series matching. *IEEE transactions on pattern analysis and machine intelligence*, 31(2):306–318.

[56] Martínez, A., Schmuck, C., Pereverzyev Jr, S., Pirker, C., and Haltmeier, M. (2020). A machine learning framework for customer purchase prediction in the non-contractual setting. *European Journal of Operational Research*, 281(3):588–596.

[57] Mobasher, B., Dai, H., Luo, T., and Nakagawa, M. (2002). Using sequential and non-sequential patterns in predictive web usage mining tasks. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 669–672. IEEE.

[58] Möller-Levet, C. S., Klawonn, F., Cho, K.-H., and Wolkenhauer, O. (2003). Fuzzy clustering of short time-series and unevenly distributed sampling points. In *International symposium on intelligent data analysis*, pages 330–340. Springer.

[59] Montero, P., Vilar, J. A., et al. (2014). Tsclust: An r package for time series clustering. *Journal of Statistical Software*, 62(1):1–43.

[60] Najafabadi, M. K., Mahrin, M. N., Chuprat, S., and Sarkan, H. M. (2017). Improving the accuracy of collaborative filtering recommendations using clustering and association rules mining on implicit data. *Computers in Human Behavior*, 67:113–128.

[61] Neysiani, B. S., Soltani, N., Mofidi, R., and Nadimi-Shahraki, M. H. (2019). Improve performance of association rule-based collaborative filtering recommendation systems using genetic algorithm. *Int. J. Inf Technol. Comput. Sci.*, 2:48–55.

[62] Niennattrakul, V. and Ratanamahatana, C. A. (2007). On clustering multimedia time series data using k-means and dynamic time warping. In *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, pages 733–738. IEEE.

[63] Niennattrakul, V. and Ratanamahatana, C. A. (2009). Shape averaging under time warping. In *2009 6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, volume 2, pages 626–629. IEEE.

[64] Paparrizos, J. and Gravano, L. (2017). Fast and accurate time-series clustering. *ACM Transactions on Database Systems (TODS)*, 42(2):1–49.

[65] Patel, D., Hsu, W., and Lee, M. L. (2008). Mining relationships among interval-based events for classification. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 393–404.

[66] Patil, S. and Khandagale, H. (2016). Survey paper on enhancing web navigation usability using web usage mining techniques. *International Journal of Modern Trends in Engineering and Research (IJMTER)*, 3(02).

[67] Pavlou, P. A. and Fygenson, M. (2006). Understanding and predicting electronic commerce adoption: An extension of the theory of planned behavior. *MIS quarterly*, pages 115–143.

[68] Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., and Hsu, M.-C. (2004). Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Transactions on knowledge and data engineering*, 16(11):1424–1440.

[69] Petitjean, F., Ketterlin, A., and Gançarski, P. (2011). A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693.

[70] Räsänen, T. and Kolehmainen, M. (2009). Feature-based clustering for electricity use time series data. In *International conference on adaptive and natural computing algorithms*, pages 401–412. Springer.

[71] Rätsch, G., Onoda, T., and Müller, K.-R. (2001). Soft margins for adaboost. *Machine learning*, 42(3):287–320.

[72] Rosenberg, A. and Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 410–420.

[73] Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

[74] Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49.

[75] Samadi, S. Y., Billard, L., Meshkani, M., and Khodadadi, A. (2017). Canonical correlation for principal components of time series. *Computational Statistics*, 32(3):1191–1212.

[76] Schafer, J. B., Konstan, J. A., and Riedl, J. (2001). E-commerce recommendation applications. *Data mining and knowledge discovery*, 5(1-2):115–153.

[77] Schapire, R. E. (1990). The strength of weak learnability. *Machine learning*, 5(2):197–227.

[78] Shang, H. L. (2014). A survey of functional principal component analysis. *AStA Advances in Statistical Analysis*, 98(2):121–142.

[79] Singhal, A. and Seborg, D. E. (2005). Clustering multivariate time-series data. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 19(8):427–438.

[80] Smyth, P. (1997). Clustering sequences with hidden markov models. In *Advances in neural information processing systems*, pages 648–654.

[81] Srikant, R. and Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In *International Conference on Extending Database Technology*, pages 1–17. Springer.

[82] Srivastava, J., Cooley, R., Deshpande, M., and Tan, P.-N. (2000). Web usage mining: Discovery and applications of usage patterns from web data. *Acm Sigkdd Explorations Newsletter*, 1(2):12–23.

[83] Stefan, A., Athitsos, V., and Das, G. (2012). The move-split-merge metric for time series. *IEEE transactions on Knowledge and Data Engineering*, 25(6):1425–1438.

[84] Tóth, K., Kósa, I., and Vathy-Fogarassy, Á. (2017). Frequent treatment sequence mining from medical databases. In *eHealth*, pages 211–218.

[85] Vintsyuk, T. K. (1968). Speech discrimination by dynamic programming. *Cybernetics*, 4(1):52–57.

[86] Vlachos, M., Kollios, G., and Gunopulos, D. (2002). Discovering similar multidimensional trajectories. In *Proceedings 18th international conference on data engineering*, pages 673–684. IEEE.

[87] Vlachos, M., Lin, J., Keogh, E., and Gunopulos, D. (2003). A wavelet-based anytime algorithm for k-means clustering of time series. In *In proc. workshop on clustering high dimensionality data and its applications*. Citeseer.

[88] Wang, K., Xu, Y., and Yu, J. X. (2004). Scalable sequential pattern mining for biological sequences. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 178–187.

[89] Wang, X., Mueen, A., Ding, H., Trajcevski, G., Scheuermann, P., and Keogh, E. (2013). Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309.

[90] Wessberg, J., Stambaugh, C. R., Kralik, J. D., Beck, P. D., Laubach, M., Chapin, J. K., Kim, J., Biggs, S. J., Srinivasan, M. A., and Nicolelis, M. A. (2000). Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. *Nature*, 408(6810):361–365.

[91] Wu, Y. and Ester, M. (2015). Flame: A probabilistic model combining aspect based opinion mining and collaborative filtering. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 199–208.

[92] Yang, J. and Leskovec, J. (2011). Patterns of temporal variation in online media. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 177–186.

[93] Yao, Y., Zhao, X., Wu, Y., Zhang, Y., and Rong, J. (2019). Clustering driver behavior using dynamic time warping and hidden markov model. *Journal of Intelligent Transportation Systems*, pages 1–14.

[94] Yates, A., Kolcz, A., Goharian, N., and Frieder, O. (2016). Effects of sampling on twitter trend detection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2998–3005.

[95] Zakaria, J., Mueen, A., and Keogh, E. (2012). Clustering time series using unsupervised-shapelets. In *2012 IEEE 12th International Conference on Data Mining*, pages 785–794. IEEE.

[96] Zaki, M. J. (2001). Spade: An efficient algorithm for mining frequent sequences. *Machine learning*, 42(1-2):31–60.

[97] Zaman, T. S., Islam, N., Ahmed, C. F., and Jeong, B.-S. (2012). iwap: A single pass approach for web access sequential pattern mining. *GSTF Journal on Computing(JoC)*, 2(1).

[98] Zeng, M., Cao, H., Chen, M., and Li, Y. (2019). User behaviour modeling, recommendations, and purchase prediction during shopping festivals. *Electronic Markets*, 29(2):263–274.

[99] Zignani, M., Quadri, C., Del Vicario, M., Gaito, S., and Rossi, G. P. (2017). Temporal communication motifs in mobile cohesive groups. In *International Conference on Complex Networks and their Applications*, pages 490–501. Springer.

[100] Zihayat, M., Hut, Z. Z., An, A., and Hut, Y. (2016). Distributed and parallel high utility sequential pattern mining. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 853–862. IEEE.