Extraction de motifs séquentiels dans les flux de données

Alice Marascu

UNIVERSITÉ DE NICE - SOPHIA ANTIPOLIS

ÉCOLE DOCTORALE STIC
SCIENCES ET TECHNOLOGIES DE L'INFORMATION
ET DE LA COMMUNICATION

T H E S E

pour obtenir le titre de

Docteur en Sciences

de l'Université de Nice - Sophia Antipolis

Spécialité : INFORMATIQUE

Présentée et soutenue par

Alice-Maria MARASCU

Extraction de motifs séquentiels dans les flux de données

Thèse dirigée par Yves LECHEVALLIER

et préparée à l'INRIA Sophia Antipolis - Méditerranée,
équipe-projet AxIS

soutenue le 14 septembre 2009

Jury :

Marie-Odile CORDIER	-	Professeur, Université de Rennes 1/IRISA	-	Rapportrice
Bruno CRÉMILLEUX	-	Professeur, Université de Caen	-	Rapporteur
Stéphane LALLICH	-	Professeur, Université Lyon 2	-	Examinateur
Yves LECHEVALLIER	-	Directeur de recherche, INRIA	-	Directeur
Florent MASSEGLIA	-	Chargé de recherche, INRIA	-	Examinateur
Michel RUEHER	-	Professeur, Université de Nice-Sophia Antipolis	-	Président
Fabrice CLÉROT	-	Ingénieur, Orange R&D	-	Invité
Brigitte TROUSSE	-	Chargée de recherche, INRIA/AxIS	-	Invitée

"Si tu peux, en une heure de temps, me trier des cendres deux grands plats de lentilles, tu nous accompagneras"

Cendrillon, Frères Grimm

Je dédie cette thèse à mon parrain, *dr. Ion Camenita.*

Remerciements

Cette thèse s'est déroulée au sein de l'Institut National de Recherche en Informatique et Automatique - INRIA Sophia Antipolis - Méditerrannée dans l'équipe-projet AxIS. Je remercie la Région PACA pour avoir participé au financement de ma thèse.

Ce travail n'aurait jamais vu le jour sans l'aide et le soutien de certaines personnes que je tiens vivement à remercier.

Mes remerciements s'adressent à monsieur Yves Lechevallier pour avoir accepté d'être mon directeur de thèse. Il a toujours montré un intérêt inconditionnel pour mes travaux et répondu à mes sollicitations lorsque le besoin s'en faisait sentir. Malgré la distance, les nombreuses discussions que nous avons pu avoir se sont montrées plus qu'enrichissantes : l'orientation et la rigueur de ce travail en ont tiré un grand profit.

Je remercie sincèrement et du fond du cœur madame Brigitte Trousse pour l'accueil bienveillant qu'elle m'a réservé dans son équipe, et surtout pour l'intérêt et le soutien qu'elle m'a accordé tout au long de cette thèse. Mon séjour à Sophia Antipolis a été profondément marqué, autant sur le plan professionnel, par sa grande expérience, ses conseils pertinents et ses encouragements sincères et parfois « maternels », que sur le plan personnel par sa gentillesse et son dynamisme.

J'adresse des vifs remerciements à monsieur Florent Masseglia pour m'avoir acceptée en stage pendant mon Master, stage qui s'est poursuivi ensuite avec une thèse sous son encadrement. Je tiens à le remercier d'abord pour la gentillesse et la patience qui ont toujours accompagné nos discussions scientifiques. Je lui adresse un merci spécial pour sa grande disponibilité face à mes questions scientifiques. Je le remercie beaucoup d'avoir partagé avec moi sa grande expérience scientifique et pour ses encouragements qui m'ont toujours aidée tout au long de cette thèse.

Je remercie également monsieur Bernard Senach pour ses conseils ergonomiques et sa gentillesse.

Je suis très sensible à l'honneur que m'a fait le Professeur Michel Rueher en acceptant d'être le président de mon jury de thèse.

J'exprime toute ma gratitude aux professeurs madame Marie-Odile Cordier et monsieur Bruno Crémilleux pour avoir accepté de juger ce travail et pour leurs remarques constructives qui m'ont permis d'améliorer grandement mon document de thèse. Je les remercie chaleureusement.

Mes sincères remerciements vont également à monsieur Stéphane Lallich et monsieur Fabrice Clérot pour l'honneur qu'ils m'ont fait en acceptant de participer au jury de soutenance, malgré leur charge importante de travail.

Je remercie infiniment monsieur Viorel Negru et monsieur Constantin Lupsoiu pour leur confiance en moi.

Je tiens à remercier le personnel du service des Ressources Humaines de l'INRIA d'avoir traité et accordé beaucoup d'attention à mon dossier, en particulier : madame Cristine Calvet, madame Geneviève Lago et madame Vanessa Wallet. Je remercie les assistantes de l'équipe AxIS, Sophie Honnorat et Anaïs Cassino, pour leur gentillesse et disponibilité.

Je remercie également le personnel des services Gener et Semir de l'Inria Sophia Antipolis.

Un grand merci à tous mes anciens et actuels collègues d'équipe, à Ana, à Anca, à Ioana et à Roxana.

Enfin, je remercie mes parents, je vous ai sentis à coté de moi tout le temps, malgré la grande distance ; votre soutien m'a aidée à dépasser les moments difficiles de ma vie. Merci aussi à mon parrain et ma marraine.

Abstract

In recent years, many applications dealing with data generated continuously and at high speeds have emerged. These data are now qualified as data streams. Dealing with potentially infinite quantities of data imposes constraints that raise many processing problems. As an example of such constraints we have the inability to block the data stream as well as the need to produce results in real time. Nevertheless, many application areas (such as bank transactions, Web usage, network monitoring, etc.) have attracted a lot of interest in both industry and academia. These potentially infinite quantities of data prohibit any hope of complete storage ; we need, however, to be able to examine the history of the data streams. This led to the compromise of "summaries" of the data stream and "approximate" results.

Today, a huge number of different types of data stream summaries have been proposed. However, continuous developments in technology and in corresponding applications demand a similar progress of summary and analysis methods. Moreover, sequential pattern extraction is still little studied : when this thesis began, there were no methods for extracting sequential patterns from data streams.

Motivated by this context, we are interested in a method that summarizes the data stream in an efficient and reliable way and that has as main purpose the extraction of sequential patterns. In this thesis, we propose the CLUSO (**Clu**stering, **S**ummarizing and **O**utlier detection) approach. CLUSO allows us to obtain clusters from a stream of sequences of itemsets, to compute and maintain histories of these clusters and to detect outliers. The contributions detailed in this report concern :

- Clustering sequences of itemsets in data streams. To the best of our knowledge, it is the first work in this domain.
- Summarizing data streams by way of sequential pattern extraction. Summaries given by CLUSO consist of *aligned sequential patterns* representing clusters associated to their history in the stream. The set of such patterns is a reliable summary of the stream at time t. *Managing the history* of these patterns is a crucial point in stream analysis. With CLUSO we introduce a new way of managing time granularity in order to optimize this history.

– Outlier detection. This detection, when related to data streams, must be fast and reliable. More precisely, stream constraints forbid requiring parameters or adjustments from the end-user (ignored outliers or their late detection can be detrimental). Outlier detection in CLUSO is automated and self-adjusting.

We also present a case study on real data, written in collaboration with Orange Labs.

Keywords : data stream, sequential pattern, outlier detection, clustering.

Résumé

Ces dernières années ont vu apparaître de nombreuses applications traitant des données générées en continu et à de grandes vitesses. Ces données sont désormais connues sous le nom de flux de données. Leurs quantités de données potentiellement infinies ainsi que les contraintes qui en dérivent posent de nombreux problèmes de traitement. Parmi ces contraintes, citons par exemple l'impossibilité de bloquer un flux de données, ou encore le besoin de produire des résultats en temps réel. Néanmoins, les multiples domaines d'application de ces traitements (comme les transactions bancaires, l'usage du Web, la surveillance des réseaux, etc) ont suscité beaucoup d'intérêt tant dans les milieux industriels qu'académiques. Ces quantités potentiellement infinies de données interdisent tout espoir de stockage complet ; toutefois, on a besoin de pouvoir interroger l'historique des flux. Cela a conduit au compromis des « résumés » des flux de données et des résultats « approximatifs ».

Aujourd'hui, un grand nombre de méthodes propose différents types de résumés des flux de données. Mais le développement incessant de la technologie et des applications afférentes demande un développement au moins équivalent des méthodes d'analyse et de résumé. De plus, l'extraction de motifs séquentiels y est encore peu étudiée : au commencement de cette thèse, il n'existait aucune méthode d'extraction de motifs séquentiels dans les flux de données.

Motivés par ce contexte, nous nous sommes intéressés à une méthode qui résume les flux de données d'une manière efficace et fiable et qui permet principalement d'en extraire des motifs séquentiels. Dans cette thèse, nous proposons l'approche CLARA (**Cla**ssification, **R**ésumés et **A**nomalies). CLARA permet d'obtenir des *clusters* à partir d'un flux de séquences d'itemsets, de calculer et gérer des *résumés* de ces clusters et d'y détecter des *anomalies*. Les différentes contributions détaillées dans ce mémoire concernent :

- La classification non supervisée de séquences d'itemsets sous forme de flux. A notre connaissance, cette technique est la première à permettre une telle classification.
- Les résumés de flux de données à l'aide de l'extraction de motifs. Les résumés de CLARA sont composés de *motifs séquentiels alignés* représentant les clusters

associés à leur historique dans le flux. L'ensemble de ces motifs permet de résumer le flux de manière fiable à un instant t. La *gestion de l'historique* de ces motifs est un point essentiel dans l'analyse des flux. CLARA introduit une nouvelle gestion de la granularité temporelle afin d'optimiser cet historique.

– La détection d'anomalies. Cette détection, quand elle concerne les flux, doit être rapide et fiable. En particulier, les contraintes liées aux flux interdisent de consulter l'utilisateur final pour ajuster des paramètres (une anomalie détectée trop tard peut avoir de graves conséquences). Avec CLARA, cette détection est automatique et auto-adaptative.

Nous proposerons également un cas d'étude sur des données réelles, réalisé en collaboration avec Orange Labs.

Mots-clés : flux de données, motif séquentiel, détection d'anomalie, clustering.

Table des matières

Table des figures

printglossary

CHAPITRE 1

INTRODUCTION

LE RYTHME de plus en plus accéléré du développement du World Wide Web n'arrête pas de nous surprendre... En 2008 on comptait pas moins de 186 millions de sites Web et 1,5 milliards d'utilisateurs Internet dans le monde entier (selon royal.pingdom.com (2008)) ! Les 210 milliards d'emails envoyés par jour en 2008 suffisent pour s'imaginer le va-et-vient continu de l'information sur Internet. Les serveurs Web essayent de tenir le pas, les augmentations des capacités des sites variant en 2008 de 22.2% pour Google GFE à 336.8% pour Nginx. Mais tenir ce rythme précipité n'est pas une tâche facile. Cette croissance a engendré le développement de beaucoup d'applications qui utilisent l'Internet. Regardons seulement le grand afflux de compagnies de téléphonie mobile et des sites de vente en ligne de ces dernières années. Pensons aussi à la difficulté d'organiser, d'interroger les données, de faire des opérations de mises à jour et de retrouver l'information importante dans cet amalgame démesuré de données et... aux immenses quantités de traces d'usage produites. Mais les conséquences du développement alerte de la technologie et de l'Internet ne s'arrêtent ici ; de multiples applications doivent maintenant traiter un type de données avec des caractéristiques spéciales : des quantités de données potentiellement infinies et générées de manière continue. Il s'agit des flux de données. Il n'est pas envisageable

de stocker sur les disques durs une quantité potentiellement infinie de données et malgré l'important développement de la technologie et des vitesses de calculs, traiter une si grande quantité de données en temps réel reste un défi. Cependant, les données étant passagères et les capacités de stockage limitées, faire des résumés des données passées représente un bon compromis. A cela s'ajoute aussi la variation continue des comportements des données du flux et la nécessité de traiter les données au moins en temps réel. L'information est actuellement générée à des vitesses qui dépassent la capacité de traitement des systèmes actuels. L'approximation, l'adaptation à la vitesse et aux capacités de mémoire disponibles constituent les éléments caractéristiques définissant les algorithmes qui traient les flux de données.

Etant données les multiples applications possibles, beaucoup se sont penchés sur la fouille dans les flux de données (Data Stream Mining). Le but est d'extraire de l'information utile tout en respectant les contraintes délicates des flux de données. L'objectif de la fouille de données statiques - l'extraction " de structures compréhensibles, potentiellement utiles, valides et nouvelles (inconnues) "Fayyad *et al.* (1996a) - représente dans la même mesure un objectif pour les flux de données. Que veut-on savoir sur un flux de données ? "Les motifs fréquents" se compte parmi les réponses à cette question. Les racines des motifs séquentiels sont les règles d'association et les itemsets fréquents. Les motifs séquentiels apportent un côté temporel aux motifs et cette caractéristique leur offre beaucoup d'applications (un site Web de vente sur Internet serait intéressé de savoir que les internautes qui achètent le livre A - contenant un régime d'amaigrissement - vont revenir acheter dans les deux jours qui suivent le livre B - contenant les recettes des plats du régime spécifié dans le livre A). Quand j'ai commencé la thèse de doctorat il n'y avait aucune méthode d'extraction de motifs séquentiels dans les flux de données malgré un besoin important. Motivés par ce domaine, on s'est intéressés à la mise en place d'une méthode d'extraction de motifs séquentiels approximatifs dans les flux de données.

En 2008, 53.8 trillions de spams ont été envoyés, ce qui représente... 70% des emails envoyés ! Quant aux virus, on en compte pas moins d'1 million en avril 2008. En 2008 les codes malicieux ont augmenté face à l'année précédente de... 468% ! Malheureusement, ces codes malicieux font aussi parti des données des flux et cela ajoute encore un défi

dans le traitement des flux : la détection d'anomalies. A notre connaissance, cette détection repose toujours sur (au minimum) un paramètre. Toutefois, la rapidité avec laquelle les données d'un flux défilent ne laisse pas le temps d'ajuster ce (ces) paramètre(s). Une méthode de détection d'anomalies répondant à ces problèmes est proposée dans cette thèse.

Adapter des algorithmes conçus pour l'environnement statique à l'environnement dynamique n'est pas toujours possible principalement en raison de leurs multiples passages sur les données, du besoin de connaître tout l'ensemble de données (opération de jointure par exemple) ou encore de leur complexité exponentielle qui risque de bloquer le flux de données. Afin d'offrir des conditions qui simulent le cas statique, beaucoup de méthodes utilisent le concept de fenêtres Chang & Lee (2003a,b); Chi *et al.* (2006); Giannella *et al.* (2003), ainsi les opérateurs qui ont besoin de connaître tout l'ensemble de données en avance peuvent être utilisés. Pour faire face au grand afflux de données, Tatbul *et al.* (2003) propose de délester les données : soit aléatoirement soit en fonction de leur importance ; mais la majorité des méthodes proposées font des résumés des données du flux. Faire des résumés suppose de faire des choix parmi les données, mais comment choisir les bonnes données à garder quand on ne connaît pas les données qui arriveront ? Un élément peut ne pas être fréquent au début du flux, alors que dans le temps il peut devenir fréquent. Comme la boule magique des flux de données n'est pas inventée, on se basera sur la probabilité des fréquences. En conséquence, on va privilégier les données fréquentes (qui ont une probabilité plus grande de réapparition) au détriment des autres (qui ont une probabilité plus petite de réapparition). Néanmoins, ce choix impose un prix à payer : des résultats approximatifs.

Quatre méthodes sont proches de la nôtre. En voici un survol.

Giannella *et al.* (2003) propose une méthode d'extraction d'itemsets à partir des flux de données. Cette méthode ne traite pas les motifs séquentiels. La méthode est basée sur un traitement par batches de transactions où, pour un batch, on traite et résume seulement les transactions de ce batch. Ils gardent une structure appelée FP-Stream qui contient un résumé des données vues jusqu'à présent. Le degré d'ancienneté des données décide le degré de compression des données selon une échelle logarithmique.

Kum (2004) s'intéresse à l'extraction de motifs séquentiels, mais dans un cadre statique. Leur méthode, appelée ApproxMAP, peut se résumer comme un algorithme d'alignement multiple de motifs séquentiels appliqué sur les résultats d'un clustering. Le clustering permet de former des groupes de séquences similaires et l'alignement qui est ensuite appliqué permet de résumer les séquences avec un très bon degré d'approximation.
Au commencement de ma thèse il n'existait pas de méthode d'extraction de motifs séquentiels dans les flux de données. Mais parallèlement, les auteurs de Raïssi *et al.* (2005) se sont intéressés à ce sujet et ils ont proposé, entre temps, l'approche SPEED qui extrait des motifs séquentiels dans les flux de données. Toutefois, ils se sont intéressés à des séquences d'items, alors que traitons des séquences d'itemsets. La section 8.2.2 donne plus de détails sur ces travaux.
Chen *et al.* (2005) prend en compte plusieurs flux de données à la fois et ils extraient les motifs séquentiels qu'ils considèrent être des motifs séquentiels multidimensionnels.

Contributions

Le sujet de cette thèse se situe au croisement des motifs séquentiels et des flux de données, en héritant de leurs contraintes et difficultés de traitement. Notre but a été de contourner tous ces contraintes et difficultés et d'arriver à un algorithme capable d'extraire les motifs séquentiels dans les flux de données. Et... nous considérons que cet objectif est atteint.

Nous proposons la méthodologie **CLARA** (**Cla**ssification, **R**ésumés et **A**nomalies). CLARA permet d'obtenir des *clusters* à partir d'un flux de séquences d'itemsets, de calculer et gérer des *résumés* de ces clusters et d'y détecter des *anomalies*.
Clusters : avec CLARA, nous proposons une classification non supervisée de séquences d'itemsets sous forme de flux. A notre connaissance, cette technique est la première à permettre une telle classification.
Résumés : les résumés de CLARA sont composés de motifs séquentiels approximatifs représentant les clusters associés à leur historique. Ces motifs ne sont pas fréquents au sens du support mais l'ensemble de ces motifs permet de résumer le flux de manière fiable. La gestion de l'historique des motifs est un point essentiel dans l'analyse des flux. CLARA introduit une nouvelle gestion de la granularité temporelle afin d'optimiser cet

historique.
Anomalies : la détection d'anomalies dans les flux doit être rapide et fiable. En particulier, les contraintes liées aux flux interdisent de consulter l'utilisateur final pour ajuster des paramètres (une anomalie détectée trop tard peut avoir de graves conséquences). Avec CLARA, cette détection est automatique et auto-adaptative.

Si on désassemble notre méthodologie CLARA, ces pièces de puzzle représentent elles-mêmes des contributions qui peuvent être considérées séparément et qui sont parfaitement utilisables dans d'autres contextes. Ces contributions sont :

- Une nouvelle stratégie de gestion de l'historique des motifs séquentiels approximatifs fréquents, REGLO (Résumés à Erreur Globale Optimisée), qui permet une représentation à granularités temporelles adaptatives. L'historique d'un élément fréquent porte le nom de "courbe". Cette courbe est formée en fait par plusieurs segments qui forment une partition de la courbe. Afin de s'adapter à un espace de mémoire limité, il faut compresser ces segments d'une manière intelligente. Le premier problème traité porte sur la recherche d'une distribution optimale des segments disponibles afin de minimiser l'erreur globale dans un traitement en ligne. Un système d'optimisation erreur globale-locale permet à tout moment de compresser les endroits au moindre impact sur le cadre général. Ainsi, on garde une image de l'historique très peu altérée. De plus, cette stratégie a le grand avantage de se limiter à une certaine quantité de mémoire fixe que l'on peut spécifier au début du processus. A notre connaissance, ceci représente la première stratégie optimisant l'erreur globale et locale à la fois et qui a la capacité de s'auto-adapter à un espace de mémoire fixe ;

- Une technique d'approximation rapide et fiable de deux segments, AMi (Approximation par les Milieux). Cette technique permet de résumer deux segments et peut être utilisée dans tout contexte, quelle que soit la signification des segments. Une version de la régression avec des formules simplifiées est également proposée ;

- Un composant innovant de stockage des motifs séquentiels approximatifs fréquents et d'administration de leurs historiques. Ceci est composé par un arbre préfixé

et par des ATGTs (Adaptative Time Granularity Table). Afin de s'ajuster à la quantité de mémoire disponible, l'ATGT utilise une méthode rapide de détection des endroits les plus compressibles (dictée par REGLO) et une méthode rapide de compression (basée sur AMi). Ce composant permet de répondre à l'interrogation sur une valeur à un certain moment de temps avec une erreur bornée ;

- Trois algorithmes de clustering qui permettent un regroupement des séquences similaires dans le but d'une compression ultérieure ;

- Une structure rapide capable de détecter la détérioration de la qualité du centroïde d'un cluster. Cette structure permet d'exploiter une technique d'alignement de séquences de manière incrémentale ;

- Une méthode automatique, sans paramètre et auto-adaptative de détection d'anomalies. Dans un environnement dynamique il faut réagir sur mesure : c'est-à-dire aussi rapidement que possible. En raison de cela, il faut implémenter des méthodes rapides et qui, implicitement, n'aient pas besoin d' intervention humaine (ce qui prendrait du temps) dans leur déroulement. Nous proposons une méthode capable de séparer des données ordonnées en ordre croissant d'après une caractéristique (dans notre cas la taille des clusters, mais ceci peut être remplacée par toute autre caractéristique) en distinguant les anomalies des données normales. Notre méthode n'a besoin d'aucune intervention humaine, elle est auto-adaptative et sans paramètres, ce qui la rend parfaitement adaptable dans le contexte des flux.

Plan de la thèse

Ce document est composé de trois parties principales.
Tout d'abord le lecteur est invité à faire un *"Tour d'Horizon"* où on présente un survol des domaines que notre sujet de thèse touche. Cette section permet à un lecteur non- ou peu-initié de prendre un premier contact avec les notions utilisées dans cette thèse et va le préparer pour une bonne compréhension de la suite du document.
Ensuite, dans la deuxième partie intitullée *"Fil d'Ariane"*, le lecteur entre dans le labyrinthe dont la sortie dépend de réponses à certaines questions. Cette partie contient la

présentation des questions auxquelles cette thèse s'est intéressée et y répond. Le lecteur commence à suivre le fil d'Ariane et il découvre, étape par étape, plusieurs éléments. Au début le lecteur trouve une carte générale du labyrinthe qui va lui faciliter le parcours du dur chemin qu'il devra suivre. La première destination sur la carte est une présentation des travaux existants sur les motifs séquentiels et les flux de données. Ensuite il constate pourquoi il est nécessaire d'appliquer des méthodes d'approximation dans le contexte des flux de données et une technique d'approximation lui est également présentée. Son chemin continue avec la découverte d'une nouvelle stratégie de gestion de l'historique des séries et d'une nouvelle méthode d'unification de segments. Trois méthodes de clustering représentent son nouveau stop et l'utilité de grouper les éléments similaires est exposée. Son voyage se poursuit avec la découverte d'une méthode de détection d'anomalies sans paramètre et auto-adaptative et la nécessité d'une telle méthode est présentée.
Toutes les pièces de puzzle étant découvertes, il se dirige vers la *"Sortie du labyrinthe"* où il peut lire les conclusions de ce voyage. L'expérience du labyrinthe a trouvé son exploitation dans le monde réel et un exemple est présenté en détail avec d'autres exemples d'applications possibles et des perspectives.

Première partie

TOUR D'HORIZON

CHAPITRE 2

A LA RENCONTRE DE LA FOUILLE DE DONNEES

2.1 Introduction

L'IMAGE SUIVANTE est une représentation satirique notoire de l'évolution[1] :

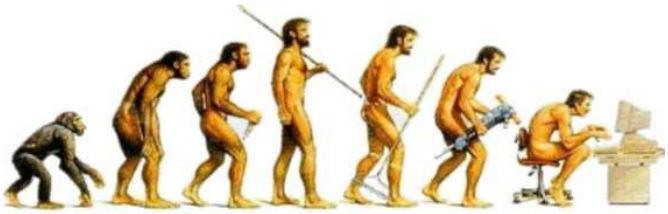


FIG. 2.1: Evolution de l'homme

La dernière étape présentée dans la figure 2.1 a débuté dans les années 1940-1945 quand les premiers ordinateurs sont apparus. C'était le début d'une nouvelle ère qui a vu se développer tout un système complexe : code barre, scanner, télescopes, satellites, cameras de surveillances, e-mails et le petit voisinage est devenu un réseau mondial (Internet) qui est même en cours d'extension (missions d'exploration de l'espace). Tout

[1] *http : //hydrodictyon.eeb.uconn.edu/courses/EEB*210/*evolution.jpg*

cela s'accompagne d'une immense quantité de données et de la multiplication des préfixes binaires (tera, peta, exa, zetta, yotta...). Le monde digital essaie de tenir le rythme. Presque tout ce qui est lié aux dispositifs électroniques digitaux respecte la loi de Moore (vitesse des processeurs, capacité de mémoire, nombre de pixels des cameras digitales etc). Malgré cela, la quantité de données à traiter est souvent trop importante pour être traitée en temps réel. Nous sommes dans l'ère de l'information et grâce aux technologies sophistiquées actuelles, on est capable accumuler des quantités de données dispersées démesurées, le meilleur exemple étant les satellites. Malheureusement, les progrès du monde digital ne sont pas suffisants pour nous aider à nous repérer dans ces amalgames de données. Ce problème a été initialement résolu par la création de bases de données structurées et de systèmes de gestion de bases de données (SGBD), mais la prolifération des SGBD a conduit ensuite à une accumulation encore plus importante d'informations. De plus, dernièrement les prix de stockage ont baissé fabuleusement, on dispose de calculateurs de plus en plus puissants à domicile, les ordinateurs deviennent de moins en moins chers et l'accès à l'Internet est entré dans le quotidien. On a, aujourd'hui, plus d'information que l'on peut traiter.

Ce chapitre présente un rapide survol du champ de la fouille de données en abordant successivement les points suivants :

- origines de l'extraction de connaissances ;
- définition de la fouille de données et de l'extraction de connaissances ;
- étapes d'un processus d'extraction de connaissances ;
- types de données concernées ;
- principales fonctionnalités.

2.2 Pourquoi et d'où vient l'extraction de connaissances ?

On avait besoin d'une discipline pour résumer automatiquement les données, pour extraire l'"essence" de l'information stockée et pour découvrir des motifs (modèles) dans les amalgames de données (ou ensemble de données brutes). En 1989, Gregory Piatetsky-Shapiro a donné un nom à cette discipline : Extraction de Connaissances à partir de Données (ECD), ou en Anglais, Knowledge Discovery in Databases (KDD). Il ne s'agissait pas d'une discipline toute nouvelle, car elle avait des racines bien ancrées, la plus longue des racines étant les statistiques. Sans les statistiques on n'aurait pas eu

aujourd'hui l'ECD car les statistiques sont le fondement de la majorité des technologies utilisées dans la fouille de données. La deuxième racine est l'intelligence artificielle ; contrairement aux statistiques, cette discipline est basée sur des heuristiques et essaie d'apporter aux méthodes statistiques des calculs inspirés de la réflexion humaine. L'apprentissage automatique constitue la troisième racine et peut être représenté comme une sorte d'intersection entre les statistiques et l'intelligence artificielle. L'objectif de l'apprentissage automatique est de faire en sorte que les programmes apprennent des expériences passées, en utilisant des statistiques, pour les concepts fondamentaux, ainsi que des algorithmes et des heuristiques d'intelligence artificielle avancées. D'autres racines sont la reconnaissance de formes, les bases de données, l'acquisition des connaissances pour les systèmes experts, la visualisation de données et les calculs de haute performance. L'élément unificateur de tous ces domaines est l'extraction d'information de haut-niveau à partir des données de bas-niveau, le tout dans un contexte de larges quantités de données. Aujourd'hui, cette discipline se trouve parmi les dix technologies émergeantes du XXI-e siècle[1] aux cotés de disciplines telles que les nano-technologies ou la bio-informatique.

2.3 Que sont la fouille de données et l'extraction de connaissances ?

La première définition de l'ECD a été donnée par Fayyad, Piatetsky-Shaphiro et Smith en 1996 : "un processus non-trivial d'identification de structures compréhensibles, potentiellement utiles, valides et nouvelles (inconnues) dans les bases de données" Fayyad *et al.* (1996a). Ensuite, plusieurs définitions sont apparues :

- La fouille de données est un processus d'extraction d'informations recevables, compréhensibles, préalablement inconnues et valides et leurs utilisations pour prendre des décisions cruciales, Zekulin - Friedman (1997) ;
- La fouille de données est un ensemble de méthodes utilisées dans le processus d'extraction de connaissances afin de distinguer des relations et des motifs préalablement inconnus dans les données Cios *et al.* (1998) ;

[1] MIT Technology Review, janvier-février 2001

- La fouille de données est l'utilisation des méthodes statistiques sur ordinateur afin de découvrir des motifs utiles dans les bases de données, Leichter & Whiteside (1989) ;
- La fouille de données est un processus de support de décision à la recherche de motifs imprévus et inconnus dans de grandes bases de données, Parsaye - Friedman (1997) ;
- La fouille de données est un processus de découverte de motifs avantageux dans les données John (1997).

Historiquement, la notion de recherche des motifs utiles dans les données a reçu plusieurs noms en anglais : data mining, knowledge extraction, discovery of regularities, patterns discovery, data archeology, data dredging, business intelligence, information harvesting, data pattern processing. Le terme de data mining a été utilisé plutôt par les statisticiens, les analystes de données et des communautés de systèmes d'administration de l'information (MIS). Il y a beaucoup de confusion entre la fouille de données et l'extraction de connaissances et leurs définitions dépendent beaucoup du domaine d'activité et du point de vue de celui qui formule la définition. En théorie, "la fouille de données est une étape dans le processus d'extraction de connaissances, qui consiste dans l'application d'algorithmes de découverte et d'analyse de données qui, avec des limites computationnelles acceptables, produit une certaine énumération de motifs (ou modèles) à partir des données'" Fayyad *et al.* (1996a). Mais en pratique, le nom de fouille de données s'est imposé beaucoup plus que les autres, à tel point que, très souvent, on l'utilise pour nommer tout le processus d'extraction de connaissances.

2.4 Etapes du processus de l'ECD

De la même façon que l'on a plusieurs définitions pour la fouille de données/extraction de connaissances selon les domaines d'applications et la vision de chacun des domaines, les étapes du processus de l'ECD (figure 2.2) varient d'un domaine à l'autre, mais tous ces domaines ont en commun ces étapes :

- la sélection des données ;
- le prétraitement des données ;
- la transformation des données ;
- la fouille de données ;

– l'interprétation et l'évaluation des modèles.

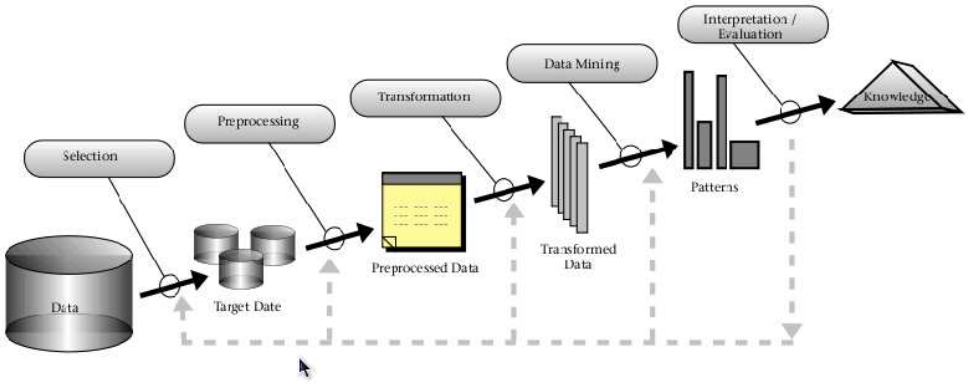


FIG. 2.2: Etapes de l'ECD Fayyad *et al.* (1996b)

2.5 A quel type de données s'applique la fouille de données ?

En principe, la fouille de données peut s'appliquer à tous les types de données. Toutefois, selon chaque type de données, les algorithmes de la fouille de données diffèrent. Quelques exemples de types de données (Zaïane (1999)) auxquels peut s'appliquer la fouille de données sont :

– **"flat file"** : Ce sont des fichiers en format texte ou binaire, contenant un enregistrement par ligne, avec des champs séparés par des délimiteurs, tels que les virgules ou les tabulations. Dans ce type de fichiers, les données peuvent être des transactions, des séries temporelles, des mesures scientifiques etc ;

– **base de données relationnelle** : Une base de données relationnelle est une base de données consistant dans des tableaux séparés, avec des liaisons explicitement définies et dont les éléments peuvent être combinés sélectivement comme des résultats à des interrogations. Chaque tableau contient des colonnes (correspondantes à des tuples) et des lignes (correspondantes à des attributs) ;

– **entrepôt de données** : Ce terme désigne une base de données utilisée pour collecter et stocker des informations provenant de multiples bases de données

(souvent hétérogènes) et qui les traite comme un tout-unitaire (comme une seule base de données) ;

- **base de données transactionnelle** : Une base de données transactionnelle est un ensemble d'enregistrements représentant des transactions, chaque enregistrement ayant une marque temporelle, un identificateur et un ensemble d'articles ;
- **base de données multimédia** : Ce type de base de données inclut des vidéos, des images, des audio et des textes média ;
- **base de données temporelles** : Elles contiennent des données organisées dans le temps, comme par exemple des activités log ;
- **bases de données orientées objet et relationnelle objet** : Il s'agit d'un type spécial de base de données (ou base de données relationnelle) où les données sont des objets ;
- **bases de données spatiales** : Une telle base de données est optimisée afin de garder des données spatiales et de pouvoir en être interroger ;
- **world wide web** : Le WWW est le dépôt le plus hétérogène et dynamique possible.

2.6 Quelles sont les fonctionnalités de la fouille de données ?

Globalement, les tâches de la fouille de données peuvent être divisées en : tâches descriptives (description des propriétés des données) et tâches prédictives (déductions sur les données pour faire des prédictions). Comme l'a synthétisé Zaïane (1999), les fonctionnalités de la fouille de données sont :

- **la caractérisation** : Par la caractérisation des données, on fait un résumé de toutes les caractéristiques générales des classes à étudier et on produit des *règles de caractérisation* ;
- **la discrimination** : En comparant les caractéristiques générales de deux classes (classe cible et classe de contraste), la discrimination obtient des *règles discriminatoires*. Les techniques sont similaires avec celles de la caractérisation, sauf que la discrimination utilise en plus des mesures comparatives ;
- **l'analyse d'association** : Le but de l'analyse d'association est la découverte des *règles d'association* ; basée sur la fréquence des items (le support) ou la probabilité

conditionnée qu'un item apparaît quand un autre item apparaît (la confiance), elle cherche à identifier les ensembles d'items fréquents. Notons également la recherche d'enchaînements fréquents, qui utilise des principes similaires et qui est la base de l'extraction de motifs séquentiels ;

- **la classification** : Dans une première étape, la classification (ou la classification supervisée) utilise un ensemble de données d'entrainement pour se construire un modèle (des étiquettes) et, ensuite, dans une deuxième étape, classe les objets en se comparant à ce modèle ;
- **la prédiction** : Très attractive pour les affaires (ou pour le domaine commercial), elle suppose de prédire des valeurs inconnues ou manquantes ;
- **le clustering** : Similaire à la classification, le but du clustering est d'organiser les données dans des classes ; par contre, on n'a aucune information préalable et le clustering tout seul doit découvrir les classes. Le clustering est également nommé *classification non-supervisée*, parce que la classification n'est pas dictée par un modèle (des étiquettes). Le principe général de répartition des éléments à des classes repose sur une minimisation des distances intra-classe et sur une maximisation des distances inter-classes ;
- **l'analyse des anomalies** : Appelées aussi *exceptions* ou *surprises*, les anomalies représentent des éléments qui ont un comportement différent du comportement général, ou autrement dit, des déviations de la tendance générale. Leurs significations sont multiples, variant du non-important (bruit) au très important (détection des fraudes) ;
- **l'analyse de l'évolution et de la déviation** : Ceci s'applique aux données temporelles. L'analyse de l'évolution cherche à repérer des tendances et consiste à caractériser, comparer, classifier les données ou à grouper les données dans des clusters. L'analyse de la déviation mesure les différences entre les valeurs attendues et celles obtenues et s'interroge sur les causes des déviations.

Après cette présentation générale de la fouille de données, le chapitre suivant entre dans les détails en se focalisant sur un axe de travail particulier, celui de l'analyse des motifs fréquents.

CHAPITRE 3

A LA RENCONTRE DES MOTIFS FREQUENTS

LES SUPERMARCHES représentent l'exemple le plus suggestif d'applications des motifs fréquents. Dans un supermarché on trouve une multitude de produits avec des demandes/consommations différentes. On a besoin de connaître les produits les plus demandés pour des décisions de prix, de promotions ou de rangement. Dans un supermarché les produits les plus recherchés peuvent être le pain, le lait et l'eau (*itemset fréquents*). Ensuite on a besoin de connaître des associations de produits qui se vendent ensemble, comme par exemple le soda et les chips (*règles d'associations*). En ajoutant une caractéristique temporelle à un ensemble d'articles, on obtient par exemple que les clients qui achètent un ordinateur portable reviennent plus tard pour acheter une souris (*motifs séquentiels*).

On parle d'extraction de motifs fréquents depuis relativement peu de temps (1993) Agrawal *et al.* (1993), malgré cela, dès son apparition ce domaine a reçu beaucoup d'attention et continue d'en recevoir. L'extraction de motifs fréquents suppose la recherche d'éléments fréquents dans un ensemble de transactions. Plusieurs tâches de la fouille de données, telles que les règles d'associations, la classification ou le clustering utilisent les motifs fréquents. La recherche des motifs fréquents a de multiples applications comme le commerce électronique, la bioinformatique, la détection d'intrusions etc.

Ce chapitre aborde les principales notions de l'analyse des motifs fréquents en les illustrant à chaque fois de quelques exemples d'applications réelles possibles. Il traite ensuite en section 3.2 successivement des points suivants :

- notion d'itemset ;
- règles d'associations ;
- motif séquentiel.

La dernière section discute de l'extraction de motifs séquentiels par rapport à l'extraction d'itemsets et de règles d'association.

3.1 Définitions

Dans la suite on gardera les concepts de client et d'achat de Agrawal & Srikant (1995) et nous adapterons ces concepts à d'autres types de données.

Definition 3.1 *Soit* $\mathcal{A} = \{a_1,\ a_2,\ ...,\ a_n\}$ *un ensemble de* n *attributs. Chaque attribut* a_i *de* $\mathcal{A}$ *est également appelé* ***item****. Un* ***itemset*** I *est un ensemble d'items non vide noté* $I = (i_1,\ i_2,\ ...,\ i_k)$ *ou* $i_j \in \mathcal{A}$ *et* $\forall i, j \in I,\ i \neq j$*. Si l'itemset* I *contient* k *éléments, il est appelé* $k - itemset$*.*

Definition 3.2 *Soit* $\mathcal{C}$ *un ensemble de clients et* $\mathcal{D}$ *un ensemble de dates. Soit* $C \in \mathcal{C}$ *un client et* $D \in \mathcal{D}$ *une date. Une* ***transaction*** *constitue, pour* C*, l'ensemble des items* I *associés à (achetés par)* C *à la date* D *et s'écrit sous la forme d'un triplet :* $< C, D, I >$*.*

Definition 3.3 *Un* ***motif séquentiel*** *(également appelé* ***séquence****) est une liste ordonnée non vide d'itemsets notée* $< s_1\ s_2\ ...s_k >$*, où* s_j *est un itemset. Une* ***séquence de données*** DS *représente les achats d'un client : soit* T_1*,* T_2*, ...,* T_n *les transactions d'un client, ordonnées par* $id - date$ *croissants et soit* $itemset(T_i)$ *l'ensemble des items correspondant à* T_i*, alors la séquence de données de ce client est* $D =< itemset(T_1)\ itemset(T_2)\ ...\ itemset(T_n) >$*.*

Exemple 1 *Soit* C *un client et* $S =< (3)(4\ 5)(8) >$*, la séquence de données représentant les achats de ce client.* S *peut être interprétée par « C a acheté l'item 3, puis en même temps les items 4 et 5 et enfin l'item 8 ».*

Definition 3.4 *Une* ***base de données*** *de clients* BD *est l'ensemble des séquences de données des clients. Soit* C *l'ensemble des clients,* BD *est donnée par :*

$BD = \{DS_c | c \in C)\}$,

Definition 3.5 *Une* ***base de données*** *de clients BD est l'ensemble des séquences de données des clients. Soit C l'ensemble des clients, BD est donnée par :*
$BD = \{DS_c | c \in C)\}$,

Definition 3.6 *Le* ***support*** *d'un itemset I dans BD, noté $supp(I, BD)$ ou $supp(I)$ quand le contexte est clair, est le pourcentage de tous les clients dans BD dont l'union des transactions contient I :*
$supp(I, BD) = |\{DS \in BD \mid I \subseteq \bigcup_{\forall d \in \mathcal{D}} itemset(DS_d)\}|/|\{DS \in BD\}|$.

La définition 3.6 donne la définition du support d'un itemset la plus répandue dans la communauté ECD. Cependant, cette notion de support a connu plusieurs variantes, comme l'indiquent les remarques 3.1 et 3.2.

Remarque 3.1 *Dans la littérature, on peut trouver différentes notions de support, selon le contexte applicatif :*

- *Dans Agrawal & Srikant (1995) le support d'un itemset est donné par le pourcentage de transactions qui contiennent l'itemset ;*
- *Lu et al. (1998) proposent d'extraire des règles d'associations inter-transactions. Dans ce cas, on compte les apparitions répétées pour la même transaction ;*
- *Mannila et al. (1997) utilise le concept de "fenêtre glissante" (sliding window). Dans ce contexte, à un instant t, on regarde seulement les éléments qui appartiennent à une fenêtre qui glisse et le support est le pourcentage de fenêtres qui contiennent l'itemset.*

Remarque 3.2 *Certaines méthodes (en particulier, les méthodes basées sur l'échantillonnage) calculent les items ou itemsets fréquents avec une erreur de support ε. Cette erreur étant donnée au commencement par l'utilisateur comme paramètre. Soit σ le support minimal et $\rho = \varepsilon/\sigma$, où ε est l'erreur du support maximale. On dit qu'un item ou un itemset est :*

- *Fréquent si son support est plus grand que σ ;*
- *Sous-fréquent si son support est plus petit que σ, mais plus grand que ε ;*
- *Non-fréquent s'il n'est ni fréquent ni sous-fréquent.*

Definition 3.7 *Soit* $ms_1 =< a_l\ a_2\ ...a_n >$ *et* $ms_2 =< b_1\ b_2\ ...b_m >$ *deux motifs séquentiels.* ms_1 *est* ***inclus*** *dans* ms_2 *(*$ms_1 \prec ms_2$*) si et seulement si il existe* $i_1 < i_2 < ... < i_n$ *des entiers, tels que* $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, ..., a_n \subseteq b_{i_n}$.

Exemple 2 *Le motif séquentiel* $ms_1 =< (3)(4\ 5)(8) >$ *est inclus dans le motif séquentiel* $ms_2 =< (7)(3\ 8)(9)(4\ 5\ 6)(8) >$ *(car (3)* $\subseteq (3\ 8), (4\ 5) \subseteq (4\ 5\ 6) et (8) \subseteq (8)$*). En revanche,* $< (3)(5) > \not< (3\ 5) >$ *(et vice-versa).*

Definition 3.8 *Soit* $ms =< a_l\ a_2\ ...a_n >$ *un motif séquentiel et* $sd =< b_1\ b_2\ ...b_m >$ *une séquence de données.* ms *est* ***inclus*** *dans* sd *(*$ms \prec sd$*) si et seulement si il existe* $i_1 < i_2 < ... < i_n$ *des entiers, tels que* $a_1 \subseteq itemset(b_{i_1}), a_2 \subseteq itemset(b_{i_2}), ..., a_n \subseteq itemset(b_{i_n})$.

La notion de support d'un motif séquentiel est donnée par la définition 3.9. Un client supporte un motif séquentiel ms (fait partie du support de ms) si s est inclus dans la séquence de données de ce client. Le support d'un motif séquentiel ms est calculé comme étant le pourcentage des clients qui supporte ms.

Definition 3.9 *Le* ***support*** *d'un motif séquentiel* ms *dans* BD*, noté* $supp(ms, BD)$ *ou* $supp(ms)$ *quand le contexte est clair, est le pourcentage de toutes les séquences de données* DS *dans* BD *telles que* $ms \prec DS$ *:*
$supp(s, BD) = |\{DS \in D \mid s \prec DS\}|/|\{DS \in D\}|$.

Soit $minSupp$ le support minimum, fixé par l'utilisateur. Comme l'indique la définition 3.10, un motif séquentiel qui vérifie le support minimum est un motif séquentiel fréquent.

Definition 3.10 *Soit* ms*, un motif séquentiel. Si* $supp(ms, D) \geq minSupp$*, avec* $minSupp$ *une valeur fixée par l'utilisateur, alors le motif séquentiel* ms *est un* ***motif séquentiel fréquent****.*

Definition 3.11 *Soit une base de données* BD*, l'ensemble* LBD *des* ***motifs séquentiels fréquents maximaux*** *est donné par :*
$LBD = \{ms | supp(ms, BD) \geq minsupp \wedge \nexists ms' | supp(ms', BD) \geq minsupp \wedge ms \prec ms'\}$.

Dans la définition 3.11 LBD est constitué de tous les motifs séquentiels fréquents tels que pour chaque motif séquentiel ms dans LBD, ms n'est inclus dans aucun autre

motif séquentiel fréquent de LBD. Il s'agit des motifs séquentiels fréquents maximaux.

Nous utiliserons dans la suite les définitions d'extraction de motifs séquentiels approximatifs tel que définies par Kum (2004).

Definition 3.12 *Soit dist une mesure de distance normalisée de deux séquences avec des valeurs dans* $[0, 1]$. *Une séquence* s_1 *est* ***plus similaire*** *à une séquence* s_i *qu'une autre séquence* s_2 *si* $dist[s_i, s1] < dist[s_i, s2]$.

Definition 3.13 *Etant donné une distance minimale* min_{dist}, *le* ***support approximatif*** *d'une séquence appartenant à une base de données D est défini comme :* $\widetilde{sup}(s_p) = \| s_i \mid (s_i \in D) \wedge (dist(s_i, s_p) \leq min_{dist}) \|$

Definition 3.14 *On appelle* ***motif séquentiel approximatif*** *un motif séquentiel dont le support approximatif est suppérieur à la valeur minSupp (un support minimum spécifié par l'utilisateur).*

3.2 Exploitations possibles de ces connaissances

Les techniques d'extraction qui permettent d'obtenir les itemsets et les motifs séquentiels fréquents peuvent trouver leurs applications dans diverses situations. Cette section en donne quelques exemples.

3.2.1 Itemsets et règles d'association

A l'aide des itemsets fréquents, un supermarché peut connaître les corrélations fréquentes dans ses ventes d'articles, un site peut découvrir ses pages les plus visitées (et éventuellement augmenter les prix des annonces publicitaires de ces pages ou proposer des promotions ou de la publicité de manière adéquate) ou bien une compagnie de téléphonie peut dépister les destinations des appels de ses clients les plus corrélées. Il est possible de calculer le degré d'implication entre des sous-ensembles d'un itemset fréquent. Selon Agrawal *et al.* (1993), par **règle d'association** on comprend une implication de forme $X \Longrightarrow Y$, où X est un ensemble d'items dans I et I_j est un item simple dans I et qui n'apparait pas dans X. La règle $X \Longrightarrow I_j$ est satisfaite dans l'ensemble de transactions T avec un facteur de confiance $0 \leq c \leq 1$ si et seulement si au moins $c\%$ des transactions de T qui satisfont X, satisfont I_j aussi". X est appelé le *corps* ou

l'*antécédent* de la règle et Y la *tête* ou la *conséquence* de la règle. Le **support** d'une règle d'association $X \Longrightarrow Y$ est le support de $X \cup Y$. Si le support dépasse une certaine valeur spécifiée (le seuil), la règle d'association est appelée **fréquente**. Si on note avec T l'ensemble de transactions sur I, la confiance d'une règle d'association $X \Longrightarrow Y$ peut s'exprimer comme la probabilité conditionnée que la transaction contient Y, étant donné qu'elle contient X :

$$confidence(X \Rightarrow Y, T) := P(Y|X) = \frac{support(X \cup Y, T)}{support(X, T)}$$

Une règle est appelée *confiante* si $P(Y|X)$ dépasse la valeur d'un certain seuil de confiance minimale γ, avec $0 \leq \gamma \leq 1$.

Exemples d'applications réelles possibles

Pour un supermarché il serait utile de connaître les règles d'associations qui ont comme *conséquent* le foie gras, cela afin de savoir quelles décisions il faut prendre pour renforcer la vente du celui-ci, comme par exemple faire des promotions sur l'Armagnac ou les figues.
Connaître l'*antécédent* aide à savoir l'impact de celui-ci sur le conséquent. Par exemple, l'impact des ventes de vin sur les ventes de fromages : si on augmente/diminue la quantité de vin du supermarché, alors il faut augmenter/diminuer celle des fromages aussi.
Si on veut s'assurer de vendre les éléments du conséquent, on peut tout simplement faire des promotions pour les éléments de l'antécédent.
Les éléments liés dans une règle d'association (vin-fromage, foie gras-Armagnac, chips-Coca Cola, bière-cacahuètes) donnent des indices que ces éléments doivent être vendus ensemble, rangés à proximité etc.

3.2.2 Motifs séquentiels

La découverte des motifs séquentiels Agrawal *et al.* (1993) consiste à trouver des motifs inter transactions comme par exemple un ensemble d'items suivi par un autre item dans un ensemble ordonné de transactions et avec des contraintes de temps. Autrement dit, les motifs séquentiels ont pour objectif la découverte d'enchaînements fréquents dans les bases de données, avec des contraintes temporelles. Une application typique

est la découverte de comportements des clients dans un supermarché (« 15% de clients achètent une télévision puis un magnétoscope et enfin un caméscope quelques jours plus tard »).

On trouve dans la littérature actuelle de nombreux travaux sur l'extraction des motifs séquentiels à partir de grandes quantités de données stockées dans des bases de données ou des fichiers Agrawal & Srikant (1994c); Masseglia *et al.* (1998); Pei *et al.* (2001a). La découverte de ces relations temporelles a été largement utilisée dans des domaines variés comme le marketing sélectif, l'aide à la décision, le management, la bioinformatique, l'analyse des performances des systèmes, l'analyse des réseaux de télécommunication etc. Un secteur important d'applications pour l'extraction de relations temporelles est l'analyse du panier de la ménagère, qui étudie les comportements des clients en cherchant des séries d'articles qui sont fréquemment achetés dans un intervalle de temps donné.

Exemples d'applications réelles possibles

L'historique des visites des utilisateurs Web est gardé dans les fichiers Log. Le fait que chaque transaction gardée dans le fichier Log soit associée à une estampille temporelle ouvre largement la porte aux chercheurs de méthodes d'extraction de motifs séquentiels et les avantages sont nombreux : la possibilité de prédire les comportements des utilisateurs, de proposer aux utilisateurs des liens appropriés, etc.
Un système de fouille de Web peut découvrir des comportements comme :

- 10% des personnes qui ont visité "/company/toys" ont fait dans un délai d'une semaine une recherche google d'après le mot "Disney" ;
- 15% des personnes qui ont acheté le livre "*/amazon/Theorie_de_la_relativite*" ont acheté dans les 10 jours d'après les livres "*/amazon/Bibliographie_de_Einstein*" et "*/amazon/Bibliographie_de_Stephen_Hawking*".

On peut aussi être intéressé par des inter-corrélations entre des données dans un certain intervalle de temps. Par exemple, on pourrait savoir :

- quelles caractéristiques communes ont les personnes qui ont continué d'acheter des voitures ou d'investir en bourse dans un intervalle d'une semaine après l'annonce de la crise mondiale ;
- les points communs des personnes qui achètent un jour avant les soldes.

3.3 Extraction de motifs séquentiels versus extraction d'itemsets et de règles d'association

La *différence théorique* entre les itemsets et les motifs séquentiels réside dans les contraintes temporelles. Alors que {foie gras, Armagnac, figues} représente un itemset, {foie gras suivi par l'achat de l'Armagnac et des figues dans les trois jours suivants} exprime un motif séquentiel.
Mais il y a une *différence pratique* très importante qui complique l'adaptation d'un algorithme d'extraction d'itemsets à l'extraction de motifs séquentiels : si dans un itemset les items sont uniques, dans le cadre d'un motif séquentiel on peut avoir des répétitions d'items.

Exemple 3 *Une page Web /.../devenirRiche.pdf souvent visitée peut donner un motif séquentiel de ce type {/.../devenirRiche.pdf, /.../devenirRiche.pdf, ..., /.../devenirRiche.pdf }.*
Un client qui raffole du chocolat (comme moi) peut acheter {chocolat, chocolat, chocolat, ..., chocolat}.

Appliquer les méthodes classiques d'extraction d'itemsets qui utilisent une étape de génération de candidats conduirait à une certaine longueur de la séquence et en conséquence, au blocage de la mémoire, d'où la difficulté d'extraire les motifs séquentiels (voir plus de détails dans la section 8.2.2).

En d'autres termes, l'extraction des règles d'association s'intéresse aux motifs intra-transactions, alors que l'extraction de motifs séquentiels s'intéresse à la recherche des motifs inter-transaction.

La différence entre l'extraction de règles d'associations et l'extraction de motifs séquentiels consiste dans le fait que la deuxième s'applique juste sur une base de données dont les transactions ont été préalablement triées en ordre croissant en fonction du temps associé à chaque transaction.

Dans sa forme la plus simple, l'extraction de motifs séquentiels n'impose pas de contraintes temporelles (intervalle de temps entre deux transactions), tant que les transactions sont faites par le même client.

Alors que dans l'extraction de règles d'association, les transactions sont utilisées seulement pour compter le support, dans l'extraction des motifs séquentiels, un client

peut contribuer à un motif candidat seulement quand on compte les supports des motifs candidats. Plus simplement, dans le premier cas on compte le pourcentage de transactions, alors que dans le deuxième on compte le pourcentage de clients.

Après cette traversée rapide et générale des notions utilisées dans l'analyse des motifs fréquents, le chapitre suivant analyse les particularités du traitement des flux de données.

CHAPITRE 4

A LA RENCONTRE DES FLUX DE DONNEES

DEPUIS DE NOMBREUSES ANNEES, on assiste à un développement technologique de plus en plus rapide : les technologies hardware changent d'un jour à l'autre et dans beaucoup de domaines le volume de données est si important qu'il est devenu impossible de le stocker sur les machines. Beaucoup de systèmes d'aujourd'hui sont bombardées en continu avec de grandes quantités de données générées à de grandes vitesses. Ajoutons à tout cela le fait que ces systèmes doivent être capables de répondre aux interrogations en temps réel. En quatre mots : grande vitesse, grande quantité. Que faire ? Donc Cendrillon n'était pas la seule à devoir traiter beaucoup de données (à séparer deux plats de lentilles des cendres) en peu de temps (une heure). En outre, l'histoire de Cendrillon de nos jours a plusieurs versions encore plus compliquées :

- la belle mère continue de renverser des lentilles au fur et à mesure que Cendrillon les sépare des cendres (les données arrivent en continu) ;
- la belle mère renverse beaucoup de plats de lentilles, mais ne donne à Cendrillon qu'un plat vide pour qu'elle ramasse les lentilles les plus grandes, sans que Cendrillon sache auparavant leur taille maximale (besoin de retenir les informations importantes avec une quantité de mémoire limitée) ;

- la belle mère mélange dans les lentilles des insectes et Cendrillon doit les trouver rapidement car ceux-ci mangent les lentilles (détection d'attaque et d'intrusion) ;
- etc.

Ce chapitre traite des problèmes posés par le traitement des flux de données. Après en avoir donné une définition et précisé les différences entre bases de données statiques et dynamiques, les contraintes et les défis du traitement de flux sont analysés. Des exemples de flux et des domaines d'applications sont ensuite fournis pour ancrer les analyse dans une réalité plus concrète.

4.1 Définition

Ce type de données qui arrivent en grande quantité et en continu est connu sous le nom de flux de données (data stream) et reçoit un intérêt croissant. A notre connaissance il n'y a pas dans la littérature de définition stable pour ce concept de flux de données. En fait, cette notion est récente et provient de contextes applicatifs particuliers. Nous proposons ici une définition pour ce concept.

Definition 4.1 *Un flux de données se présente sous la forme d'une séquence de signaux numériquement codés qui est utilisée pour transmettre de l'information. Cette séquence est continue, illimitée, et produite à une très grande vitesse.*

Dans la définition 4.1, on entend par *grande vitesse* le fait que les outils de traitement existants sont insuffisants en comparaison de la rapidité de production des données.

4.2 Statique versus dynamique

Comme Motwani le synthétise dans le tutoriel de PODS Motwani (2002), les différences entre les bases de données statiques et dynamiques sont multiples :

SGBD	SGFD
Relations persistantes	Flux temporaires
Interrogations ponctuelles (one time)	Interrogations continues
Accès aléatoire	Accès séquentiel
Espace de stockage sans limite	Mémoire principale limitée
Seul l'état actuel est important	Importance de l'historique des données
Pas de services en temps réel	Besoin en temps réel
Niveau de mises à jour relativement bas	Possibilité d'arrivée de multi-gigabytes[1]
Données à toute granularité	Données à de granularité fine
Résultat précis	Résultat approximatifs
Design de base de données physique, plan d'accès établie par le processeur de requêtes	Arrivée imprévisible/variable de données et caractéristiques

4.3 Contraintes et défis dans le traitement des flux de données

Contrairement aux données statiques, les données dynamiques lèvent beaucoup de défis dans leur traitement. Dans le modèle des flux de données, les données arrivent séquentiellement et elles sont traitées en utilisant un espace de mémoire insuffisant pour stocker toutes les données. Premièrement, comme les données arrivent en continu, il serait naïf de penser à garder les données dans la mémoire principale. En conséquence, les algorithmes qui traitent des flux de données gardent un résumé des données traitées qui est capable de répondre aux questions sur ces données avec un certain degré d'approximation. Quant aux calculs, tout doit être fait en utilisant la mémoire principale. Deuxièmement, comme les données passent une seule fois, une méthode dans le style des méthodes statiques traditionnelles qui parcourent les données plusieurs fois, n'est pas envisageable. Tous les algorithmes statiques qui font du backtracking sur les données sont donc inadaptés. Troisièmement, afin de garder le rythme, il faut implémenter des algorithmes rapides sur le plan du temps de traitement des données, mais aussi sur le plan du temps de réponse aux interrogations. La situation change si rapidement que le résultat de la requête évolue pendant l'exécution de la requête ; les algorithmes doivent s'adapter aux changements qui ont lieu pendant l'exécution. Même le taux de génération des données n'est pas constant dans le temps, cela conduisant à une condition appelée

burtyness. La caractéristique principale des flux de données est leur instabilité et les algorithmes qui traitent les flux de données doivent s'y adapter. Les flux de données nécessitent un traitement en temps réel. Souvent, les interrogations sont continues, et donc, il est nécessaire d'avoir une évaluation en continu des données qui arrivent et, implicitement, une mise à jour continue. Quatrièmement, le problème combinatoire qui intervient dans le calcul des item(set)s et des motifs séquentiels dans le cas statique, ajoute à son tour de nouvelles difficultés. Cinquièmement, si dans un cas statique, un algorithme arrive au blocage, on n'a qu'à relancer l'interrogation et tout ce qu'on perd est le temps de relancer l'interrogation. Mais dans le cas dynamique, la situation se complique car on risque de perdre les données qui sont passées pendant le blocage. Imaginons seulement le cas d'une banque qui a perdu une partie de ses transactions (et les clients chanceux qui ont fait des achats pendant cette période-là) ! Eviter à tout prix les blocages rend impossible l'utilisation des opérations de jointure qui sont la base d'une majorité d'algorithmes d'extraction de motifs. La nature dynamique des flux de données impose toutes ces contraintes ; en conséquence, les algorithmes existants pour le cas statique ne peuvent plus être appliqués. Afin de satisfaire ces conditions, le traitement des flux de données sacrifie l'exactitude du résultat de l'analyse en permettant quelques approximations (le plus souvent bornées). Il est donc reconnu que l'approximation et l'adaptabilité sont des éléments clés pour exécuter des requêtes et extraire des connaissances dans les flux de données. C'est un défi de trouver les motifs séquentiels dans un flux de données, avec un espace de mémoire limité et dans un environnement dynamique où l'on permet des opérations comme l'insertion et la suppression massives des données.

4.4 Exemples de flux de données

Les flux de données les plus actifs sont probablement issus des satellites et des télescopes. Un tel exemple est le nouveau télescope Effelsberg ajouté le 1 avril 2008 au e-EVN, le Réseau Européen VLBI (Very Long Baseline Interferometry), et qui a presque doublé sa capacité d'observation en arrivant à *6 Gbit de données astronomiques par seconde*. e-EVN est ainsi devenu le réseau interférométrique radio le plus rapide au

monde[1].

Une caméra orbiteur de reconnaissance lunaire LROC (Lunar Reconnaissance Orbiter Camera) de la NASA a été lancée en mai 2009. Dans le cadre de cette opération, *30 GB de données par jour* seront envoyés via un satellite à Arizona State University's (ASU) Fulton School of High Performance Computing. D'autres données seront envoyées par le Goddard Space Flight Center de la NASA et au total, la NASA estime accumuler environ 330 TB de données[2].

Un autre exemple de flux de données appartient à Siemens et il s'agit d'une turbine à gaz. En février 2009, Siemens a mis en place la plus grande turbine à gaz comptant 2 838 capteurs et *une heure de tests produit 90 GB de données* [3]

4.5 Domaines d'applications

Les domaines d'application des flux de données sont nombreux et variés. Selon Koudas & Srivastava (2005) on peut diviser les flux de données en deux grandes catégories :

– *flux de données transactionnels* ;

– *flux de données de mesure.*

La première catégorie s'intéresse aux interactions entre des entités, alors que la deuxième surveille leurs évolutions dans le temps.

Flux de données transactionnels

Un premier élément du groupe des flux de données transactionnels est représenté par les transactions bancaires. Les opérations bancaires forment un flux de données continu et parmi les requêtes typiques nécessitant des réponses en temps réel on peut énumérer : la détection des retraits massifs, l'évaluation des fluctuations des comportements des clients (ou d'un seul client), la détection d'intrusions, l'étude du comportement d'un client afin de lui proposer des services spécialisés etc. Les télécommunications représentent un deuxième exemple de flux de données transactionnel. Les appels des clients sont étudiés afin d'adapter les services aux besoins des clients. Le Web aussi fait partie

[1] $http://www.euindiagrid.eu/newsroom/news/effelsberg-telescope-inaugurated-as-member-ofthe-e-evn-network/$, $http://www.expres-eu.org/Effelsberg_inauguration.html$, $http://news.bbc.co.uk/2/hi/technology/3093294.stm$

[2] $http://searchdisasterrecovery.techtarget.com/news/article/0,289142,sid190_gci1350630,00.html$

[3] $http://www.controleng.com/article/ca6637328.html?nid=2488\ \&\ rid=1768760$

du même groupe que les précédents. Les logs Web et les flux de clics de pages Web sont des exemples de flux de données. Les accès des clients aux ressources forment un flux de données et l'analyse des comportements des clients a de nombreux avantages (propositions adaptées à chaque client, étude comparative des comportements des clients, évaluation des intérêts des clients, choix des produits en promotion, proposition des produits similaires, classement des liens les plus visités, etc).

Flux de données de mesure

Les flux de données de mesure surveillent l'évolution des états des entités.

Un premier domaine d'application est la surveillance des réseaux IP. En octobre 2006 SoftScan annonçait que les spams représentaient 89,07% du volume total d'e-mails reçus dans le mois. Comme si cela n'était pas suffisant, en 2008 le directeur de la recherche AntiSpam Bitdefender annonçait une hausse du nombre de spams suite à la crise financière. La détection des spams fait partie des applications des flux de données tout comme la détection d'intrusions. Dans le cadre de la surveillance des réseaux, on a souvent besoin de réponses à des questions d'estimation et d'analyse de trafic du type : "Quelle est la quantité de données transférées entre deux adresses IP ? Combien d'adresses IP sont inactives ? A-t-on des problèmes de déni de services ?". Assurer la sécurité des réseaux implique des analyses de type : "Quelles sessions ont des durées beaucoup plus longues que la durée moyenne ? Quelle session transmet beaucoup plus de données que la quantité moyenne ? Y a-t-il des sessions qui enregistrent des pics et dans ce cas quels sont les IPs correspondants ? Y a-t-il des IPs communs à plusieurs sessions ?"

Un deuxième exemple de domaine d'application réside dans les réseaux de capteurs. Dans cette catégorie entrent d'abord les satellites, qui offrent les flux de données présentant la plus grande quantité de données à traiter. Les systèmes de surveillance des réseaux reçoivent des informations de plusieurs sources et doivent détecter les situations critiques comme par exemple une surcharge des réseaux ou une situation bloquante. La détection d'anomalies et d'intrusions fait partie aussi des applications. Le trafic routier, les phénomènes physiques, les satellites, les télescopes, le climat, les systèmes de prévision de la météo ou encore des phénomènes séismiques sont d'autres exemples qui appartiennent à cette même catégorie.

Après cette présentation des particularités du traitement des flux de données, le chapitre suivant aborde un domaine qui reçoit un intérêt croissant du fait de la diversité de ses domaines d'application, il s'agit de la détection d'anomalies.

CHAPITRE 5

A LA RENCONTRE DE LA DETECTION D'ANOMALIES

5.1 Introduction

L'INTERET CROISSANT pour la détection d'anomalies tient à la variété de ses applications dans des domaines très diversifiés. Ce chapitre se propose de faire un point sur cette fonctionnalité de la fouille de données en traitant successivement les points suivants :

- Forme des résultats ;
- Types de méthodes de détection d'anomalies ;
- Une discussion pour analyser les méthodes exposées.

On retrouve la détection d'anomalies sous plusieurs noms : détection de nouveautés (novelty detection) Ratle *et al.* (2007), détection d'outliers/anomalies Barnett & T. Lewis (1994); Ben-Gal (2005), détection de déviations Andreas Arning (1996)[1], détection de bruit Raphisak *et al.* (2004) ou encore fouille d'exceptions (exception mining) Luo *et al.* (2008). Les anomalies portent également différentes appellations selon les domaines : outliers, anomalies, exceptions, observations aberrantes, surprises, contaminants etc, mais les plus utilisées restent anomalies ou outliers.

[1] Elle est liée à la détection du bruit, mais ce n'est pas identique

5. A LA RENCONTRE DE LA DETECTION D'ANOMALIES

Les anomalies sont soit les pépites les plus intéressantes, soit les plus nuisibles. Leurs causes d'apparition sont complexes, variant d'erreurs de saisie, fautes mécaniques, erreurs d'instruments de mesure aux actions malveillantes. Les causes que j'appelle « neutres (involontaires) » (telles que les erreurs de saisie) mènent à des résultats incorrects, modèles erronés, paramètres mal estimés etc. Les conséquences néfastes des causes que j'appelle « négatives » (tel que les attaques pirates) sont plus faciles à imaginer (comme le vol ou la perte d'information). Tout cela explique l'intérêt croissant accordé à la détection d'anomalies.

Dans plusieurs domaines, la détection d'anomalies est devenue une des premières étapes dans le processus de prétraitement des données. Les études montrent qu'entre 0.01 et 0.5% Lazarevic & Kumar (2005) des données sont des anomalies (quelquefois arrivant à 10% ou plus Wang *et al.* (1995)) et si on rapporte ces valeurs aux grandes quantités de données manipulées aujourd'hui, on comprend bien pourquoi l'étape de détection d'anomalies fait partie du prétraitement des données et c'est pourquoi elle est si importante aujourd'hui. Pyle (1999); Williams *et al.* (2002) décrivent l'importance d'avoir des données « propres » pour tout type d'analyse.

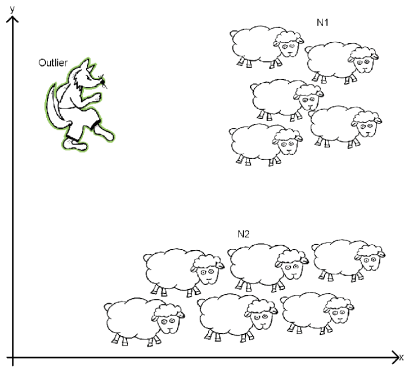


FIG. 5.1: Représentation d'une anomalie en 2D

Un exemple de représentation d'anomalie en 2D est illustré dans la figure 5.1 où les groupes N1 et N2 représentent les comportements normaux, alors que le comportement isolé, "Outlier", est une anomalie. Des algorithmes capables de tracer une frontière entre les anomalies et les enregistrements normaux doivent être développés. Les conséquences de la découverte des anomalies sont multiples : des corrections (saisies erronées), des

suppressions (mesures instrumentales erronées), des alertes (intrusion dans le système), des découvertes/nouveautés (éléments inconnus avant) etc.

La découverte d'éléments inconnus/nouveaux a été étudiée par Markou & Singh (2003a,b); Singh & Markou (2004). Quant aux éléments découverts, Singh & Markou (2004) indiquent que "la nouveauté peut être définie comme ces situations nouvelles qu'on n'aurait pas pu prédire à partir des expériences précédentes et qui représentent vraiment des événements nouveaux" et ils donnent des exemples du domaine de la reconnaissance d'objets dans la vidéo. Un autre cas d'anomalie concerne l'élimination du bruit des données Rao & Pisharam (1990).

Les débuts des études sur la détection d'anomalies datent du XIX-e siècle Edgeworth (1887) et sont issus de travaux de statisticiens. Les premières méthodes n'étaient applicables qu'aux cas des données unidimensionnelles, par exemple la méthode développée par Grubbs (1969). Ensuite, plusieurs chercheurs ont commencé à faire des études spécifiques pour leur propre domaine et maintenant on les retrouve presque partout : systèmes de sécurité, activités militaires, imagerie médicale (détection de cancers, irrégularités dans les monitorages des fonctionnements de différent organes), irrégularités dans les votes, changements climatiques, activités terroristes, analyses des performances des athlètes, étude des images des satellites, détection de données erronées dans des ensembles d'apprentissage, détection de mauvais fonctionnement des appareils, analyses des performances des réseaux etc.

Vu cette multitude de domaines d'applications, on comprend pourquoi il n'existe pas une seule définition consacrée/universelle pour les anomalies, mais plusieurs adaptées à chaque domaine. De plus, les définitions portent aussi l'empreinte de la méthode utilisée ou du type de données. Une des premières définitions est celle de Grubbs (1969) : *"une observation anormale, ou un outlier, est celle qui dévie nettement des autres membres de l'échantillon dont elle fait partie"*. Une autre définition liée aux statistiques est donnée par Mendenhall & Reinmuth (1993) où les anomalies sont des valeurs improbables *"très éloignées du milieu de la distribution en toute direction"*. Les anomalies sont aussi *"des observations (ou des sous-ensembles d'observations) qui semblent être inconsistantes avec le reste de l'ensemble de données"*. Knorr & Ng (1998a) considèrent qu'« un objet O dans une base de données T est une anomalie DB(p, D) si au moins une fraction p des objets de T se trouvent à une distance plus grande de D de O »., alors que Angiulli

& Pizzuti (2005) propose une définition qui prend en compte la somme des distances aux plus proches k éléments voisins.

5.2 Formes des résultats

Il existe deux manières de présenter les résultats :

- avec des *étiquettes* (élément normal/anomalie) ;
- avec des *marquages/notes (ranking)* indiquant leur degré d'anomalie.

Le deuxième type est préféré car il apporte le grand avantage de pouvoir distinguer un degré d'anomalie parmi les données et d'établir un seuil au-dessus duquel on considère que c'est une anomalie. Hu & Sung (2003) les appellent classification *soft* (ranking) et *hard* (étiquettes).

5.3 Types de méthodes de détection d'anomalies

Il n'existe pas une classification universelle des méthodes de détection d'anomalies. Cependant, pour les résultats présentés de façon étiquetée, on peut en distinguer trois types :

- *des méthodes non-supervisées* : Ces méthodes ne disposent d'aucune connaissance préalable et la séparation des données se fait en considérant les éléments les plus éloignés comme des anomalies. Ce type de méthodes capte bien les nouveautés, mais par contre, le taux de fausses alarmes peut être très élevé ;
- *des méthodes supervisées* : Dans ce cas, on dispose d'un ensemble de données labellisées et la classification de nouveaux éléments se fait par des comparaisons avec les anomalies connues. Ce type de méthode a l'avantage d'avoir très peu de fausses alarmes, mais le désavantage de ne pas découvrir les nouvelles anomalies ;
- *des méthodes semi-supervisées* : Dans ce cas nous avons une connaissance partielle des anomalies ou des comportements normaux. La majorité de ces méthodes dispose des connaissances sur les comportements normaux. La méthode de Dasgupta & Majumdar (2002) est une des rares méthodes à travailler avec des anomalies labellisées.

Les méthodes de détection d'anomalies peuvent aussi se diviser en méthodes univariable et multi-variables. Une anomalie peut ne pas se révéler si on utilise une ap-

proche uni-variée, mais en considérant une approche multi-variée Ben-Gal (2005). Zhang & Selinus (1998) proposent une comparaison des méthodes de détection d'anomalie uni-variée et multi-variée. Ben-Gal (2005) contient une classification détaillée des méthodes de détection d'anomalies en fonction de ce critère. Ben-Gal (2005) décrit les effets de masquage (masking) et de couverture (swamping). On appelle un effet de masquage quand une anomalie empêche une autre d'être repérable, et donc celle-ci ne peut être détectée comme anomalie qu'en la considérant toute seule (donc pas en présence de la première anomalie). Une anomalie de couverture "couvre" une autre, c'est à dire que la deuxième ne se révèle comme anomalie que dans la présence de la première. Une minutieuse étude des méthodes de détection d'anomalies est faite par Ch *et al.* (2007). Selon Ch *et al.* (2007) on a neuf types de techniques de détection d'anomalies : basées sur le clustering, sur la classification, sur la méthode du plus proche voisin, sur les techniques statistiques, sur la théorie de l'information, sur la décomposition spectrale, sur les techniques visuelles, sur les techniques manuelles contextuelles et sur les techniques manuelles collectives. Williams *et al.* (2002) parle des méthodes de détection d'anomalies paramétriques et non-paramétriques, alors que Lazarevic & Kumar (2005) trouve que la majorité des techniques de détection d'anomalies peuvent se grouper en quatre types : approches statistiques, approches basées sur des distances, méthodes qui utilisent des profils et méthodes basées sur des modèles. Enfin pour Hodge & Austin (2004) les méthodes de détection d'anomalies ne sont dérivées que de trois domaines : des statistiques (basées sur la proximité, paramétriques, non-paramétriques et semi-paramétrique), des réseaux neuronaux (supervisées et non-supervisées) et de l'apprentissage automatique.

Dans la suite, je présente les principales caractéristiques de quelques types de ces méthodes.

5.3.1 Approches basées sur des algorithmes de clustering

Considérées par certains (Ben-Gal (2005)) comme faisant partie des méthodes de détection d'anomalies basée sur la fouille de données, les approches basées sur des méthodes de clustering sont très populaires. Dans le cas des méthodes de détection d'anomalies basées sur un algorithme de clustering, la sélection des anomalies se fait sur la base de l'idée que les clusters de taille petite sont des anomalies. La limitation parmi les clusters se fait par des critères de taille, de densité, de pourcentage, de distance jusqu'au centre des clusters etc. Un **point faible** de ces méthodes est lié à leur grande complexité

de calcul (beaucoup de calculs de distance). Un **point fort** est qu'elles peuvent fonctionner de manière non-supervisée. Quelques exemples des méthodes utilisant le clustering sont : Eskin *et al.* (2002); Kaufman & Rousseeuw (1990); Otey *et al.* (2003).

5.3.2 Approches basées sur des distances

Ce type d'approche a été proposé pour la première fois par Knorr & Ng (1997, 1998b, 1999); Knorr *et al.* (2000). Une étape de calcul des distances entre les éléments voisins précède la détection des anomalies. La condition de détection d'anomalies est basée sur un calcul d'une fraction d'objets qui sont plus éloignés de l'élément considéré. Différentes méthodes de calculs des distances existent selon le type des données : distance Euclidienne, distances basées sur des graphes, distances basées sur la plus longue sous-séquence commune etc. Il y a des méthodes qui détectent les anomalies selon la densité et les régions de faible densité contiendront les anomalies. Ce type de méthode est considéré comme un sous-type des méthodes basées sur les distances par Ch *et al.* (2007), alors que Papadimitriou *et al.* (2003) les considère comme une approche distincte. Breunig *et al.* (2000a) ont été les premiers à proposer ce type de méthode. Breunig *et al.* (2000b); Chawla & Sun (2006) illustrent un **désavantage** des méthodes basées sur le calcul des distances qui contiennent en même temps des données denses et rares : si la distance "d" entre les éléments d'un cluster est grande, alors on ne va pas dépister les anomalies qui sont éloignées d'une distance plus petite ou égale à "d". Un **avantage** est que par contre ce type de méthodes fonctionne de manière non-supervisée et ne suppose aucune distribution sur les données Ch *et al.* (2007).

5.3.3 Approches statistiques

Les méthodes statistiques se basent sur la construction d'un modèle statistique (une distribution, généralement la distribution normale) et par des comparaisons à ce modèle. D'habitude, le modèle décrit les comportements normaux et les éléments qui sont conformes à ce modèle seront considérés comme normaux, alors que les autres seront pris pour des anomalies. Un **point négatif** de ces méthodes est qu'elles sont trop liées à un modèle de représentation des données et souvent, il est difficile de trouver un modèle qui soit en parfaite adéquation avec les situations réelles. En plus, ces méthodes ne sont pas appropriées pour les ensembles de données multidimensionnelles. Un exemple de résultat erroné est illustré par Chawla & Sun (2006) le milieu d'un ensemble de données

est une anomalie. Ord (1996); Rousseeuw & Leroy (1996) ont des exemples d'approches statistiques. Walfish (2006) offre une étude intéressante sur les méthodes de détection d'anomalies.

5.3.4 Approches basées sur des projections

Certaines méthodes de détection d'anomalies projettent les éléments dans un autre sous-espace afin de trouver les anomalies. Les projections se font en utilisant un nombre réduit d'attributs. La décomposition en composantes principales (PCA) est la méthode la plus utilisée (Parra *et al.* (1996); Saha *et al.* (2009); Ye *et al.* (2009)). Cette méthode permet d'obtenir une liste de composantes principales et pour détecter le comportement général des éléments il suffit de garder les composantes principales associées aux plus grandes valeurs propres. Un **avantage** de ces méthodes concerne l'analyse des données multidimensionnelles, en permettant d'étudier les anomalies dans un espace réduit. Par contre, le **désavantage** est que, selon l'espace de projection, la distribution des anomalies change aussi.

5.4 Discussion

Que chercher ? Et... quel chemin suivre pour arriver à une destination inconnue ? ! Chercher quelque chose d'inconnu soulève déjà des interrogations... Mais à cela s'ajoutent encore d'autres défis. Vu le grand nombre de domaines où détecter les anomalies, on imagine facilement la grande variété des contraintes imposées par chaque domaine dans la détection d'anomalies. Les algorithmes de détection doivent donc être plus ou moins adaptés à chaque domaine. Cela implique l'intégration des connaissances d'ordre sémantique, les données seules n'étant pas suffisantes. Les mêmes données peuvent représenter du bruit pour une application alors que pour une autre elles constituent de l'information importante. Par exemple, si on cherche à dépister les endroits où vivent les ours pandas, sur une image satellite on sera intéressé par des chemins créés dans le bambou ; alors que si on veut cartographier les forêts de la Chine, on ne va pas prendre en compte les traces créées à l'intérieur des forêts par les pandas, mais d'autres informations sur la lisière de la forêt. Les attaques répétitives sur un site représentent les données d'entrée pour ceux qui travaillent dans la sécurité, alors que pour les autres, elles ne représentent que du bruit. Le

traitement des anomalies malveillantes doit intégrer des contraintes temporelles afin de stopper/diminuer leurs dégâts (détecter un virus après 5 minutes ou après 5 heures n'aura pas le même impact). S'il s'agit d'anomalies malveillantes, elles peuvent également essayer de passer inaperçues en simulant les caractéristiques d'une donnée normale, ce qui complique davantage le processus de détection. Quant aux anomalies par fautes de saisie, leurs valeurs peuvent être très proches des valeurs normales et en conséquence difficiles à distinguer. Dans la détection d'anomalies on cherche à trouver les comportements normaux et les anomalies ; mais si on fait un tri des données selon leur degré d'anomalie, comment être sûr que le seuil choisi séparera parfaitement les anomalies des données normales ? Dans cette logique binaire, on dira que les données au-dessus d'un seuil sont normales et les autres sont des anomalies. Mais une logique trivalente donne une réponse plus adaptée à la réalité car elle considère les données proches de la valeur du seuil comme inconnues. Pour aller encore plus loin, la logique floue décrit les objets à une granularité plus fine par une palette de valeurs entre 0.0 et 1.0, selon le degré d'anomalie, augmentant ainsi la difficulté pour trouver le bon seuil. Quant aux données labélisées, elles induisent de l'ambigüité dans les résultats, parce qu'on ne peut pas distinguer les anomalies entre elles. Enfin, la quantité de données dans laquelle il faut fouiller les anomalies est d'habitude immense, ce qui ajoute un défi aux algorithmes qui s'aventurent à détecter les anomalies dans ces données. On peut conclure qu'avec le progrès technologique, des types et des sources d'anomalies vont apparaître spontanément, augmentant ainsi le besoin de nouveaux algorithmes de détection.

La première partie du mémoire de thèse, le Tour d'Horizon du domaine, a abordé successivement les différentes notions requises pour aborder l'extraction de motifs séquentiels dans les flux de données et présenter les travaux réalisés dans le cadre de cette thèse.

La seconde partie est consacrée à la présentation des algorithmes originaux qui ont été développés lors de cette thèse et à leurs validations expérimentales.

Deuxième partie

FIL D'ARIANE

CHAPITRE 6

DANS LE LABYRINTHE

 ORS DE CETTE THESE, nous avons cherché les réponses à une question principale et à plusieurs questions adjacentes ou dérivées. Nos questions semblent faire partie d'une histoire de Cendrillon en version XXIème siècle où la tâche difficile de Cendrillon de séparer les lentilles de la cendre seulement en une heure est devenue extraire les motifs séquentiels des flux de données en temps réel. Ceci représente le thème centrale de ma thèse : trouver une méthode d'extraction de motifs séquentiels approximatifs dans les flux de données. Plus spécifiquement, notre objectif a été de trouver la réponse à cette question : *Etant donnée une quantité fixe d'espace de mémoire, trouvez une méthode d'extraction de motifs séquentiels à partir des flux de données en temps réel, tout en étant capable d'interroger l'historique des motifs séquentiels à tout moment de temps et de donner les réponses avec une erreur bornée et petite.*

Pour donner la solution à cette question, nous avons dû répondre aux multiples défis. D'abord on a dû satisfaire les défis des flux de données. Vu la vitesse de production des données d'un flux, on distingue les défis suivants :

- Examiner les données très rapidement, car d'autres données sont générées continuellement et il faut avoir terminé le traitement des données qui viennent de passer pour examiner les prochaines ;
- Ne pas scanner les données plusieurs fois, car on ne les stocke pas et qu'elles sont très volatiles ;
- S'adapter à des taux de données différents d'un moment à l'autre. Le taux de génération des données d'un flux varie continuellement (condition de "burstyness") ;
- Diminuer le nombre d'opération nécessaires pour des opérations de mise à jour, de recherche d'information etc ;
- Accepter l'approximation au détriment de l'exactitude et mettre en place des algorithmes d'approximation efficaces et rapides.

Une solution de traitement des données qui prend en compte ces défis de vitesse est présentée dans le chapitre 13.

Un deuxième grand défi des flux concerne les aspects quantitatifs des données. Ceci ne nous permet pas de regarder les données en totalité, mais seulement via une fenêtre d'observation limitée :

- Traiter une quantité de données potentiellement infinie en disposant d'une quantité de mémoire limitée ;
- Répondre à la difficulté d'adaptation des algorithmes conçus pour le cas statique, principalement en raison de leur problème de complexité et d'opérateurs qui ont besoin de connaître tout l'ensemble de données ;
- Vu la rapidité de l'évolution des événements d'un flux, choisir un composant capable d'être mise à jour rapidement et de manière incrémentale ;
- Mettre en place un composant capable de compresser sa taille selon les besoin de mémoire et d'être interrogée dans de courts délais.

Ce défi est omniprésent tout au long du mémoire. Le chapitre 11 propose une stratégie de gestion de la mémoire adaptée à ce défi d'espace limité est traitée. Le chapitre 10 s'intéresse à une méthode de compression de données. Une structure de données capable de répondre à ce défi et au premier défi de vitesse est présentée dans le chapitre 12.

Un autre défi de ce sujet de thèse est lié à la période limitée de temps pendant laquelle on peut voir un élément. Si on rate le moment quand l'élément passe, on ne peut plus rien faire pour le revoir. Donc, un blocage conduirait à une perte permanente de données. En conséquence, on doit tout faire pour éviter le blocage. Le défi est de mettre

en place des méthodes qui soient capables de faire face à des problèmes de blocage ou de perte d'une certaine partie de données (par exemple dans le cas d'une panne électrique). La difficulté de respecter dans un environnement dynamique est présenté en détail dans la section 8.2.2.

Une conséquence du défi de vitesse de traitement a engendré un autre lié au regroupement de données similaires. Ce sujet est traité dans le chapitre 13. En voici quelques sous-éléments de ce défi : .

- Quel critère de similitude des séquences choisir afin de classer les séquences (implicitement de minimiser le nombre de comparaisons à faire) assez rapidement pour répondre aux contraintes des flux ?
- Quelle étiquette associer afin qu'elle soit représentative de tout le groupe de séquences ?
- Quel critère de similitude des séquences choisir afin de permettre des comparaisons successives et de mis à jour de l'information (voir les sections 13.2.2.2 et 10.3) ?
- Comment décider quels groupes de séquences garder et quels groupes effacer ? On ne peut pas avoir une vision globale sur un flux, donc on ne sait pas quelles informations sont importantes. On ne connaît que l'information qui passe en ce moment et un résumé des données passées. Une séquence peut ne pas être importante (fréquente) en ce moment ou dans le passé, mais elle peut le devenir.

Un sujet d'intérêt connexe au sujet principal, l'extraction de motifs séquentiels dans les flux, est la détection des anomalies. Trouver une méthode de détection d'anomalies dans un flux de données a un défi majeur : arriver à séparer rapidement les anomalies, afin de traiter correctement seulement les bonnes données. Ce sujet est traité dans le chapitre 14.

Lors de ce mémoire, on découvrira pas à pas comment on a contourné ces défis. Afin de faciliter la lecture, le prochain chapitre contient une carte.

CHAPITRE 7

CARTE DU LABYRINTHE

DANS CE CHAPITRE, j'explique l'organisation générale des chapitres qui suivent. Afin de faciliter la lecture et la compréhension de notre méthode, des cartes avec les principales étapes parcourues sont présentées dans les figures 7.1, 7.2 et 7.3. Au début de chaque chapitre qui suit, la carte est reprise avec l'indication exacte de l'endroit traité.

Voici dans la suite une description globale de la méthodologie développée lors de cette thèse. Toutes les étapes seront présentées en détails dans les chapitres suivants. Deux grands sujets ont été abordés lors de l'élaboration de cette méthodologie : l'extraction de motifs séquentiels approximatifs dans les flux de données et la détection d'anomalies dans un flux de données. Ces deux sujets commencent par deux étapes communes (illustrées dans les figures 7.1 et 7.2 par les étapes 1 et 2). Les étapes suivantes sont distinctes (2 étapes pour l'extraction de motifs séquentiels et 4 étapes pour la détection d'anomalies).

Le traitement du flux démarre avec son fractionnement en batches. Nous avons choisi de traiter le flux de données par ***batches de transactions***. Plus simplement

dit, le flux de données est "coupé" dans des paquets d'une taille fixe et identique et, à chaque instant, on ne traite que le batch le plus récent (étape numérotée 1 dans les figures 7.1 et 7.2). Plus de détails sur les approches de traitement des flux de données sont donnés dans le chapitre 9.

Le processus de traitement d'un flux de données débute avec une étape d'initialisation qui est suivie par le ***traitement du batch courant***. Les batches se succèdent l'un après l'autre, selon l'avancement du flux. Plusieurs étapes sont effectuées lors du traitement du batch courant selon l'objectif poursuivi.

Ensuite, on a deux chemins qui fonctionnent en parallèle, selon l'objectif : extraction de motifs séquentiels approximatifs dans les flux de données ou détection d'anomalies dans un flux de données :

D'abord les séquences sont regroupées grâce à un algorithme de ***clustering*** (étape illustrée dans les figures 7.1 et 7.2 par étape 2 est présentée en détails dans le chapitre 13).

1. Pour *l'extraction de motifs séquentiels approximatifs* (5 étapes au total) :
   - On continue avec une ***sous-étape d'élagage*** où les petits clusters sont supprimés, car ils ne peuvent pas contenir des motifs fréquents.[1]

   - A la fin de l'étape de clustering, on a obtenu un ensemble de clusters qui ont un représentant (un centroïde) par cluster[2]. Un représentant d'un cluster donne un résumé du contenu du cluster correspondant et, tous les représentants des clusters forment ***un résumé*** du batch correspondant. Ces représentants des clusters constituent les motifs séquentiels approximatifs (étape illustrée dans la figure 7.1 par étape 3 et présentée en détail dans le chapitre 10).

   - Afin de garder les motifs séquentiels approximatifs pendant l'exécution du flux, on a mis en place ***une structure de stockage*** (étape illustrée dans la figure 7.1 par étape 4) . Cette structure est composée principalement d'***un arbre***

[1] En fait, ces petits clusters peuvent contenir des motifs fréquents dans le temps, mais pour des raisons de mémoire on est obligé de ne pas garder les éléments avec les plus faibles probabilités.

[2] Pour la version I de clustering, on résume les séquences des clusters après avoir fini le clustering. Les versions de clustering II et III ont un centroïde pour chaque cluster qui est déjà calculé pendant l'étape de clustering.

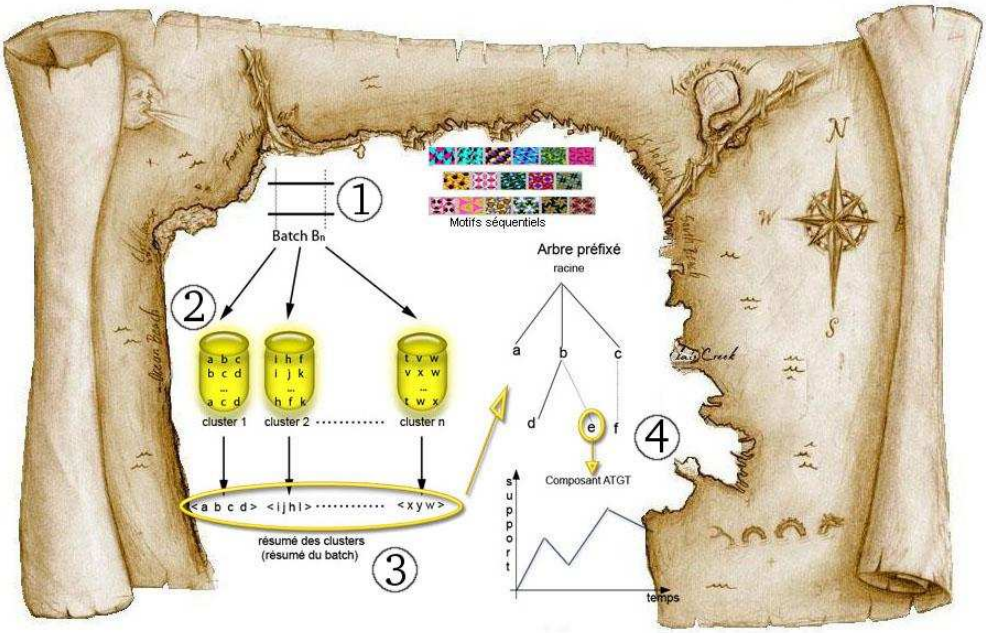


FIG. 7.1: Etapes effectuées lors du le traitement d'un batch de transactions pour l'extraction de motifs séquentiels approximatifs

préfixé qui a ***des tableaux*** (des composants ATGT[1]) attachés à ses nœuds. Chaque chemin de la racine vers les nœuds représente un motif séquentiel approximatif et le composant ATGT attaché au nœud permet d'interroger le motif séquentiel approximatif sur son historique. A la fin du traitement du batch courant, cette structure est ***mise à jour*** avec les nouveaux comportements découverts dans le batch courant. Une ***étape d'élagage*** suit, afin de renoncer aux éléments les moins importants et de se limiter à l'espace de mémoire disponible. Cette étape est détaillée dans les chapitres 12 et 11.

2. Pour *la détection d'anomalies* (6 étapes au total) :
   - On continue avec ***une étape de tri*** des clusters en fonction de leur taille, ce qui donne une distribution des clusters selon l'indice d'ordre et la taille (étape illustrée dans la figure 7.2 par étape 3 et présentée en détail dans le chapitre 14).

[1] Adaptative Time Granularity Table

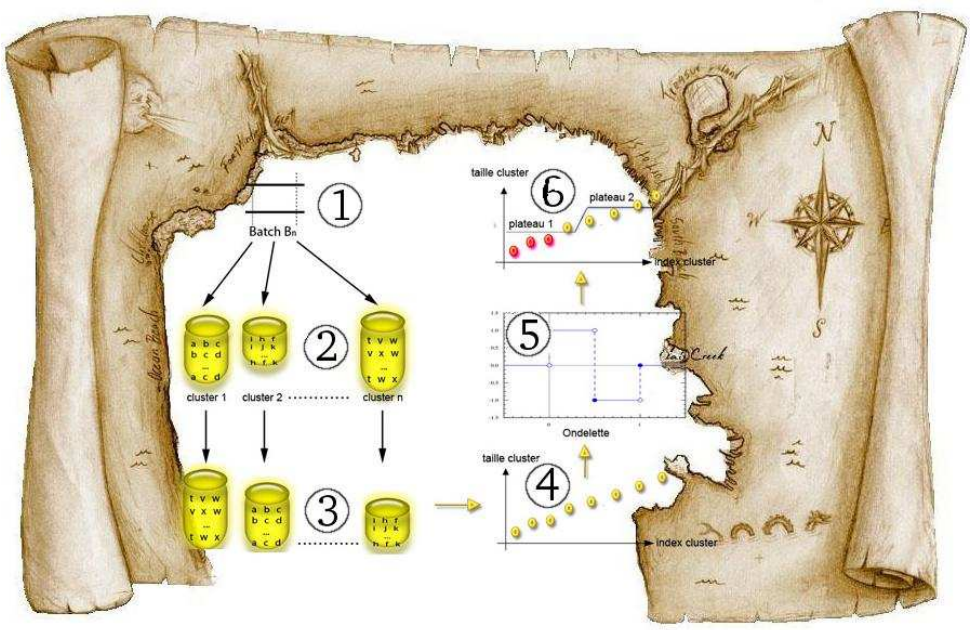


FIG. 7.2: Etapes effectuées lors du traitement d'un batch de transactions pour la détection d'anomalies

- Les clusters triés selon la taille forment une distribution (illustrée dans la figure 7.2 par étape 4). Sur la distribution des clusters, on applique ***un algorithme de détection d'anomalies basé sur les ondelettes*** (illustré dans la figure 7.2 par étape 5). Cet algorithme calcule l'indice qui donne la séparation des clusters en anomalies et normaux (résultat illustré dans la figure 7.2 par étape 6).

Ces opérations se répètent pour chaque batch à traiter. La figure 7.3 illustre ce processus. L'alignement et le résumé obtenu sont remplacés par l'algorithme de détection d'anomalies basé sur des ondelettes dans le cas de la détection d'anomalies.

Afin d'introduire toutes les notions avant leur utilisation et d'éviter des sauts vers les chapitres suivants, l'organisation de cette partie ne respecte pas l'ordre chronologique illustré dans les figures 7.1 et 7.2. Voici l'ordre des chapitres contenus dans cette partie :

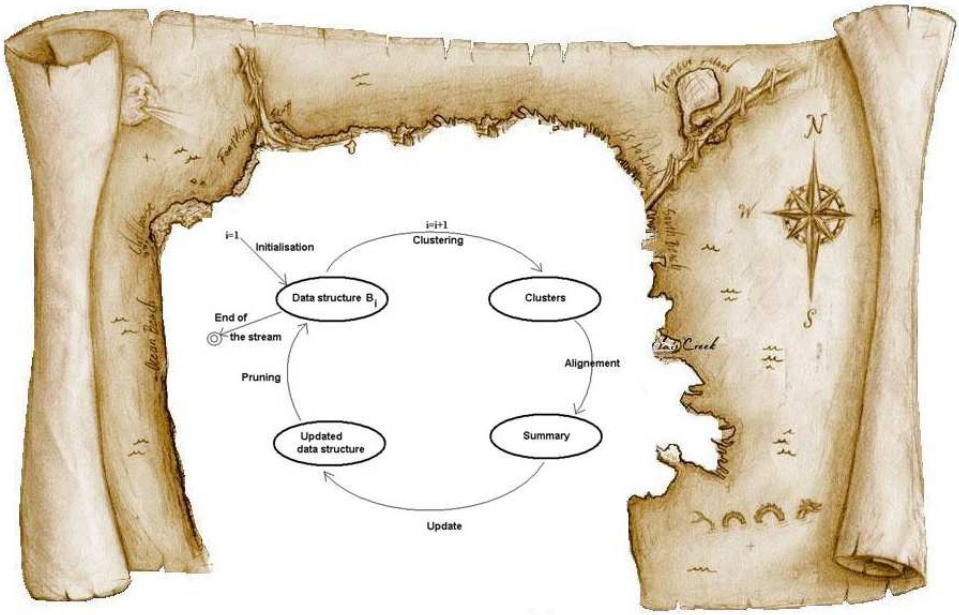


FIG. 7.3: Schéma général : traitement successif des batches de transactions

– Chapitre 8 - Ce chapitre présente les principaux algorithmes existants dans la littérature qui se sont intéressés à l'extraction de motifs séquentiels.

– Chapter 9 - Ce chapitre présente les principaux types d'approches de traitement des flux de données et positionne l'approche que nous avons choisie.

– Chapter 10 - Une condition nécessaire des algorithmes qui visent de traiter des flux de données est l'utilisation d'une méthode capable de faire des résumés des données. Une méthode existante dans la littérature capable de résumer des séquences est présentée.

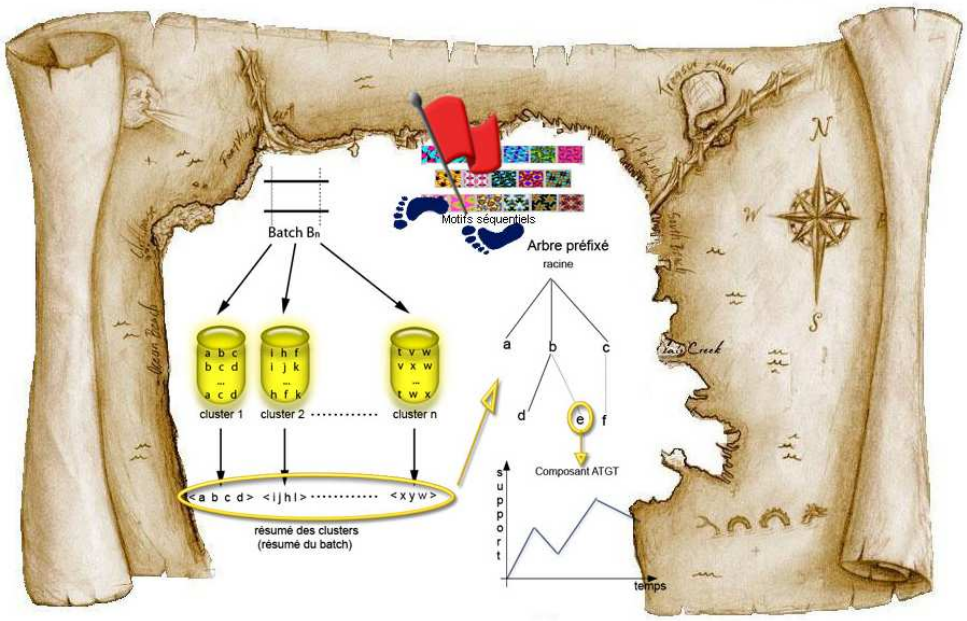
– Chapter 11 - Lors de l'exécution du flux, l'historique des données retenues doit être géré. Ce chapitre contient la description d'une stratégie de gestion de l'historique des motifs séquentiels lors de l'exécution du flux.

– Chapter 12 - Ce chapitre décrit la structure de stockage de données mise en place.

– Chapter 13 - Dans ce chapitre nous présentons des méthodes de clustering qui regroupent les séquences similaires dans des clusters (afin de mieux les résumer ensuite par l'application de la méthode présentée dans le chapitre 10.

– Chapter 14 - Ce chapitre contient la présentation d'une méthode de détection d'anomalies dans les flux de données.

CARTE GENERALE



Le chapitre suivant traite l'extraction des motifs séquentiels. Je débute avec la présentation de l'algorithme pionnier d'extraction d'itemsets et règles d'association, Apriori, j'enchaine avec une synthèse des algorithmes existants qui traitent les items et les itemsets et je finis avec la présentation des méthodes existantes qui traitent les motifs séquentiels.

CHAPITRE 8

EXTRACTION DE MOTIFS SEQUENTIELS

8.1 Algorithmes principaux

LA PREMIERE PAGE de l'histoire de l'extraction de motifs fréquents a été écrite par Agrawal *et al.* (1993) et contient le premier algorithme d'extraction d'itemsets et règles d'association, *l'algorithme AIS*[1]. Peu après, l'algorithme a été amélioré en utilisant la propriété d'anti-monotonicité du support des itemsets et la confiance des règles d'association. Cette nouvelle version de l'algorithme a été publiée sous le nom d'Apriori Agrawal & Srikant (1994a); Srikant & Agrawal (1995). Indépendamment, une méthode similaire a été développée par Mannila *et al.* (1994), avec pour objectif d'extraire des motifs répétés dans une série de valeurs. Plus tard, les auteurs de ces travaux ont publié leurs résultats en commun dans Agrawal *et al.* (1996).

Ce chapitre présente les principaux algorithmes existants dans la littérature qui se sont intéressés à l'extraction de motifs séquentiels. La première section est consacrée à la présentation de l'algorithme APRIORI, pionnier dans l'extraction d'itemsets et de règles d'association. Dans la section suivante, une synthèse des algorithmes

[1] Agrawal, Imielinsky, Srikant

existants traitant des items et des itemsets est proposée. La dernière section présente les méthodes existantes pour les motifs séquentiels.

8.1.1 Apriori

L'algorithme Apriori utilise une méthode bottom-up dans laquelle, à chaque étape, les sous-ensembles fréquents sont élargis d'un item. L'idée de base d'Apriori est qu'un itemset est fréquent si tous ses sous-ensembles sont fréquents. En conséquence, un itemset devient un candidat si tous ses sous-ensembles propres sont fréquents.
D'abord la base de données est scannée afin de trouver tous les supports de tous les 1-itemsets ; les 1-itemsets avec un support plus élevé que le support minimum[1] sont gardés. Ensuite, dans l'étape k, à partir des k-itemsets, on génère les k+1-itemsets ; la base de données est scannée afin de trouver le support des k+1-itemsets. Les k+1-itemsets dont le support est plus grande que le support minimum sont gardés pour l'étape suivante. L'algorithme s'arrête quand il ne trouve plus de nouveaux itemsets.
L'inconvénient d'Apriori est le grand nombre de sous-ensembles de candidats générés et les multiples scans de la base de données.
Évidemment Apriori est un algorithme significatif, car il marque le début de l'histoire de l'extraction de motifs, mais il a connu beaucoup d'améliorations.

8.1.2 Après Apriori

Une fois Apriori proposé, un grand nombre de nouveaux algorithmes ou d'améliorations d'algorithmes existants ont été publiés. Mais découvrir tous les itemsets fréquents reste toujours une tâche difficile car l'espace de recherche est exponentiel en fonction du nombre d'items de la base de données. Parmi les modifications les plus significatives d'Apriori qui ont été proposées, citons DHP Park *et al.* (1995b), Partition Savasere *et al.* (1995), DIC Brin *et al.* (1997) et l'algorithme d'échantillonnage de Toivonen (1996). Le premier algorithme à générer tous les candidats par une approche en profondeur d'abord (approche de type Depth-First), Eclat, a été publié en 1996, Zaki (2000); Zaki *et al.* (1997). Ensuite, plusieurs approches de type Depth-first sont apparus Agarwal *et al.* (2000a,b); Han *et al.* (2000b), parmi lesquels FP-Growth Han *et al.* (2000b, 2004) est

[1] La valeur du support minimum est un paramètre spécifié à l'entrée par l'utilisateur.

largement reconnu par la communauté. Cet algorithme a été ensuite amélioré et publié sous le nom de H-mine Pei *et al.* (2001b).

8.1.2.1 Structures de données

Dans un environnement statique, les structures de données impliquées ont souvent pour but de retenir l'information relative à l'extraction en cours (*e.g.* les motifs fréquents ou les structures intermédiaires permettant d'obtenir ces motifs). Pour retenir les données pendant l'exécution, les structures de données mises en place utilisent plusieurs techniques comme les tableaux de hachage Agrawal *et al.* (1996); Jin *et al.* (2003a), les treillis de surveillance Chang & Lee (2003b) ou encore les arbres préfixés Amir *et al.* (1997); Bayardo (1998); Borgelt & Kruse (2002); Chang & Lee (2003a,b); Giannella *et al.* (2003); Manku & Motwani (2002).
Agrawal *et al.* (1996)utilise un arbre de hachage pour garder les candidats. Dans une telle structure un nœud contient une liste d'itemsets (un noeud feuille) ou une table de hachage (noeud intérieur). Mais l'arbre préfixé reste la structure préférée. Dans un arbre préfixé (ou trie), chaque nœud correspond à un k-itemset, sauf la racine. Chaque nœud contient un item et chaque chemin de la racine à une feuille représente un itemset. Les branches d'un noeud sont implémentées à l'aide des structures comme par exemple les tableaux de hachage, les arbres binaires de recherche, les vecteurs etc.

Dans un environnement dynamique, un élément supplémentaire doit être pris en compte : l'historique des événements (ou des motifs). Datar *et al.* (2002) a introduit pour la première fois un histogramme appelé histogramme exponentiel, parce que les paquets de l'histogramme augmentent exponentiellement. Donc, les paquets sont de dimensions $1, 2, 4, \ldots, 2^m$. Cette structure est reprise et étendue dans Giannella *et al.* (2003) sous le nom de "tilted time window" (TTW). La structure introduite par Giannela est appelée FP-Stream. Elle est composée par un arbre FP-Tree Han *et al.* (2000b) et des TTWs attachées à chaque feuille de l'arbre. Ces TTWs permettent de gérer l'historique des itemsets fréquents à différents degrés de granularités en utilisant une échelle logarithmique.

8.1.2.2 Environnement statique

Comme Goethals (2003) le synthétise, les auteurs d'Apriori ont proposé plus tard une amélioration d'Apriori appelée AprioriTid qui réduit le temps de calcul du support

en remplaçant chaque transaction de la base de données par l'ensemble des itemsets candidats qui sont apparus dans la transaction. Une combinaison des méthodes Apriori et AprioriTid Agrawal & Srikant (1994b); Agrawal *et al.* (1996) a été proposée sous le nom de AprioriHybrid et donne généralement de meilleurs résultats qu'Apriori. DHP Park *et al.* (1995a) propose une autre manière d'améliorer le temps d'exécution par une réduction du nombre d'itemsets candidats, alors que Srikant (1996) propose un algorithme qui utilise un tableau triangulaire pour le calcul du support des 2-itemsets candidats. Une méthode similaire a été proposée plus tard par Orlando *et al.* (2001, 2002) avec les algorithmes DCP et DCI. Au lieu de générer les 2-itemsets candidats, Goethals (2003) propose de scanner toute la base de données et d'enlever de chaque transaction les items qui ne sont pas fréquents et, ensuite, pour chacune de ces transactions Goethals (2003) augmente le support de tous les 2-itemsets candidats contenus dans cette transaction. De cette façon, il évite de générer les 2-itemsets qui n'apparaissent pas et il réduit ainsi le nombre de 2-itemset générés. Un premier algorithme utilisant le support de tous les sous-ensembles pour calculer le support d'un itemset a été proposé par Bayardo (1998). L'algorithme DIC proposé par Brin *et al.* (1997) arrive à réduire le nombre de passes sur la base de données en divisant la base de données en plusieurs intervalles d'une taille spécifique. Toivonen (1996) propose un algorithme d'échantillonnage qui commence par une recherche de tous les motifs fréquents dans un échantillon aléatoire pour ensuite vérifier les résultats sur toute la base de données. Dans le même but, Savasere *et al.* (1995) propose un algorithme de partition où la base de données est stockée dans un format vertical et les supports des itemsets sont calculés par des opérations d'intersection des couvertures de deux de ses sous-ensembles.

8.1.2.3 Environnement dynamique

Avec l'apparition des flux de données, les algorithmes d'extraction d'item(set)s fréquents ont dû s'adapter aux contraintes des flux de données. Pour traiter les séquences d'un flux, certains algorithmes (Lossy Counting Manku & Motwani (2002), DSM-FI Li *et al.* (2008), estDec Chang & Lee (2003b), FP-Stream Giannella *et al.* (2003)) ont choisi d'utiliser les fenêtres landmark (les connaissances sont extraites depuis le début du flux et maintenues en temps réel à l'état courant du flux), d'autres Chang & Lee (2003a); Chi *et al.* (2006) ont choisi les fenêtres glissantes (les connaissances sont extraites sur une fenêtre plus courte que le flux et dont la fin correspond à l'état courant

du flux). Une structure de données intéressante est proposée par Charikar *et al.* (2002). Cette structure a la propriété majeure d'être additive en permettant ainsi de calculer la différence entre deux structures sketch [1] de deux flux de données. Ceci donne une méthode de calcul des items avec le plus grand changement de fréquence entre les flux de données. C'est le seul algorithme connu jusqu'à présent qui permet cette opération. On peut aussi appliquer cette propriété pour calculer les items avec des fréquences très différentes entre deux intervalles de temps. Dans un contexte aussi dynamique que les flux de données, retenir tout l'historique des comportements de motifs fréquents pose des problèmes de mémoire. Un parallèle entre la segmentation relaxation Teng *et al.* (2003) et la fonction d'âge Giannella *et al.* (2003) peut être fait, car tous les deux ont le même objectif : diminuer progressivement l'importance des événements dans le temps passé et ainsi, la taille de l'historique des fréquents. L'algorithme estDec Chang & Lee (2003b) utilise dans un même but une échelle de vieillissement (decay) qui est appliquée à chaque item pour lui affecter un poids selon son âge. Afin de relaxer la structure de stockage des comportements fréquents, différentes techniques d'élagage ont été élaborées Chang & Lee (2003c); Giannella *et al.* (2003). Certains algorithmes garantissent que l'erreur du résultat obtenu est bornée Charikar *et al.* (2002); Giannella *et al.* (2003); Jin *et al.* (2003b). Alors que FP-Stream de Giannella *et al.* (2003), Lossy Counting de Manku & Motwani (2002), FTP-DS de Teng *et al.* (2003), hCount de Jin *et al.* (2003b) et eDec de Chang & Lee (2003b) peuvent être interrogés pendant l'exécution, CountSketch de Charikar *et al.* (2002) et StickySampling de Manku & Motwani (2002) permettent cela seulement à la fin du traitement.

8.2 Motifs séquentiels

8.2.1 Environnement statique

La majorité des algorithmes qui se sont occupés de l'extraction de motifs séquentiels peut être divisée en deux grands groupes : similaires à Apriori et similaires à FP-growth.

[1] Un sketch ou un synopsis est une structure de données qui utilise peu d'espace de mémoire et qui est capable de répondre rapidement aux interrogations, mais avec des approximations.

Méthodes basées sur le principe de "générer-élaguer"

Dans le cas du premier groupe, les algorithmes se sont basés sur la propriété d'anti-monotonicité d'Apriori (tout sous-ensemble d'un ensemble fréquent est fréquent) Agrawal & Srikant (1994a). Parmi les algorithmes qui ont utilisé ce concept nous pouvons citer : Aprioriall, AprioriSome, DynamicSome Agrawal & Srikant (1995) et plus tard, GSP Srikant & Agrawal (1996) et SPADE Zaki (2001). L'algorithme GSP scanne plusieurs fois la base de données afin de générer à chaque passe des candidats d'une taille plus grande qu'à la passe précédente (1-séquences, ensuite 2-séquences etc). Il y a deux étapes dans cet algorithme : l'étape de génération de candidats et l'étape de comptage du support. Une différence importante de l'application du principe d'Apriori aux motifs séquentiels réside dans l'ensemble de candidats générés. Si à partir de $1 \rightarrow 2$ et $1 \rightarrow 3$ dans l'étape de génération de 2-séquences, Apriori donne (1, 2, 3) comme 3-itemset, GSP donne $1 \rightarrow 2 \rightarrow 3$, $1 \rightarrow 3 \rightarrow 2$ et $1 \rightarrow 23$. Une autre caractéristique de GSP est qu'il permet de prendre en compte des "gaps" (ou intervalles de temps entre les événements). Une autre méthode basée sur le principe d'anti-monotonicité d'Apriori est PSP Masseglia *et al.* (1998). Il s'agit d'un algorithme basé sur GSP, mais avec une structure de données qui retient les candidats et les séquences fréquentes d'une manière plus efficace. En 2000, Zaki (2001) a proposé l'algorithme SPADE avec pour but de découvrir tous les séquences fréquentes. SPADE utilise un algorithme level-wise (approche par niveaux, comme dans GSP) et une recherche verticale. L'algorithme utilise une base de données verticale dans laquelle une liste est associée à chaque item. L'avantage de cette structure est qu'en utilisant des opérations simples de jointure ou d'intersection, on obtient toutes les séquences fréquentes. Afin d'optimiser l'espace de mémoire utilisée, SPADE décompose l'espace de recherche (le treillis) dans plusieurs espaces plus petits (sous-treillis) qui peuvent être traités indépendamment. Cette décomposition se base sur une relation d'équivalence de préfixes. SPADE commence avec plus petit élément du treillis et graduellement, il génère dans une approche bottom-up toutes les séquences fréquentes. Le nombre de passes de la base de données varie de un (pour les informations prétraitées) à trois. Une autre méthode qui utilise une représentation verticale de la base de données est SPAM Ayres *et al.* (2002). Dans ce cas, les auteurs représentent les séquences à l'aide de vecteurs de bits et utilisent des opérateurs binaires pour générer et tester les candidats.

Méthodes basées sur la projection

Basés sur des projections, les algorithmes les plus cités du deuxième groupe sont : FreeSpan Han *et al.* (2000a) et PrefixSpan Pei *et al.* (2001a). FreeSpan a été proposé en premier. Un an plus tard les auteurs ont proposé des améliorations et publié le résultat sous le nom de PrefixSpan. A partir des items fréquents de la base de données, PrefixSpan génère des projections de base de données. Une telle projection de base de données contient des suffixes de séquences de la base de données originale, groupés par préfixes. Ce processus est récursivement répété jusqu'au moment où il n'existe plus d'item fréquent dans la base de données projetée. Dans cet algorithme, le motif séquentiel est le chemin des items fréquents vers la base de données projetée. D'autres versions d'algorithmes proposés sont MEMISP Lin & Lee (2002) (méthode basée sur l'indexation de la mémoire) et SPIRIT Garofalakis *et al.* (1999) (permet de spécifier des contraintes en utilisant des expressions régulières).

Motifs séquentiels fermés

Un motif séquentiel est appelé fermé quand il n'est pas inclus dans d'autres motifs séquentiels avec le même support. Le premier algorithme qui a proposé l'extraction de motifs séquentiels fermés est Xifeng Yan (2003). Cet algorithme génère les séquences fréquentes de taille 2 selon la règle "A apparaît toujours avant/après B". Une extension de cet algorithme se trouve dans l'algorithme BIDE Wang & Han (2004). Celui-ci utilise un nouveau schéma de vérification de la fermeture de la séquence appelée extension bidirectionnelle et diminue l'espace de recherche en utilisant une méthode d'élagage appelée BackScan et une technique d'optimisation Scan-Skip. L'idée principale de cette méthode est d'éviter les extensions inutiles en détectant à l'avance des extensions contenues dans les séquences.

Extraction incrémentale de motifs séquentiels

L'extraction incrémentale peut s'avérer nécessaire dans toutes les techniques d'extraction de connaissance puisque toutes les bases de données sont susceptibles être mises à jour dans le temps. Cela est encore plus important pour l'extraction de motifs séquentiels en raison du changement des données séquentielles en continu dans le temps. Dans Zhang *et al.* (2002) une approche basée sur GSP est proposée, alors que Parthasarathy

et al. (1999) propose une approché basée sur SPADE appelée ISM. ISM est une méthode capable de maintenir l'ensemble de motifs séquentiels pendant les opérations de mise à jour de la base de données et, de plus, elle permet la modification des contraintes (support minimal, items inclus et exclus) grâce à une interface. ISM garde toutes les séquences fréquentes et leurs supports dans la base de données et dans un treillis les séquences contenues dans la bordure négative [1] associées à leurs supports. D'après les auteurs, cet algorithme est plus rapide que la majorité des algorithmes d'extraction de motifs séquentiels. Alors que cette approche permet juste d'ajouter de nouvelles séquences, ISE Masseglia *et al.* (2003) rend également possible des opérations d'insertion dans les séquences existantes. Une autre caractéristique d'ISE est qu'il minimise le coût de calcul par une réutilisation de l'information minimale des anciennes séquences fréquentes, c'est à dire le support. Ainsi, l'ensemble des séquences candidates est significativement réduit. Zheng *et al.* (2002) propose une méthode appelée IUS. Alors que ISE prend seulement en compte l'extension des suffixes des séquences fréquentes, IUS étend les préfixes et les suffixes des anciennes séquences fréquentes.

Différences entre les flux et l'aspect incrémental

Les caractéristiques des méthodes incrémentales peuvent, dans un premier temps, sembler suffisantes pour répondre aux besoins exprimés dans le cadre des flux de données. Le principal point commun est en effet la nécessité de maintenir une connaissance qui est devenue obsolète suite à des mises à jour dans les données. Les connaissances sont calculées au temps t sur la base des données disponibles et doivent être révisées au temps $t + n$ suite à des modifications. Cependant les différences entre le calcul incrémental et le calcul sur les flux sont plus importantes que les ressemblances :

- La version incrémentale d'une méthode résout le même problème que la version statique mais en prenant en compte la mise à jour. Dans un problème de flux de données, il est généralement admis qu'on ne peut pas résoudre le même problème que sur les données statiques à cause des contraintes liées au flux. Par exemple l'extraction d'itemsets sur un flux impose une approximation dont on peut s'affranchir avec les données statiques. La définition d'un problème connu sur les données statiques se verra donc adaptée dans sa version flux de données ;

[1] Une bordure négative est la collection de toutes les séquences pas fréquentes, mais dont les sous-séquences sont fréquentes.

- La version incrémentale d'une méthode prend en compte les connaissances obtenues avant la mise à jour pour calculer les nouveaux résultats (ce qui permet d'obtenir les résultats plus rapidement). Dans les flux de données, l'historique des connaissances extraites est potentiellement infini. Il est donc impossible d'en tenir compte dans le calcul des nouveaux résultats car ces connaissances sont obligatoirement oubliées (avec un facteur de vieillissement la plupart du temps) ;
- Dans les flux les mises à jour sont massives et rapides, ce qui n'a pas de commune mesure avec les mises à jour que l'on connaît dans l'aspect incrémental des problèmes de fouille de données. Ainsi, l'approximation est utilisée pour contrebalancer ce volume et cette rapidité. De plus, "incrémental" ne signifie pas "temps réel" alors que dans les flux, les résultats doivent être obtenus au minimum en temps réel.

Analyse des motifs périodiques

L'analyse des motifs périodiques a pour but de trouver les motifs qui apparaissent avec une certaine périodicité dans des bases de séries temporelles. Elle peut être vue aussi comme une extension de l'extraction de motifs séquentiels en considérant la durée comme un ensemble de séquences partitionné. Il y a trois types d'algorithmes pour cette analyse : 1) les algorithmes qui extraient des motifs périodiques complets (tous les points de la série temporelle contribuent au comportement cyclique) Han & Kamber (2001) ; 2) les algorithmes qui recherchent des motifs partiellement périodiques (seulement certains points de la série temporelle sont pris en compte) Bettini *et al.* (1998); Elfeky (2000); Han *et al.* (1999); Yang *et al.* (2000) et 3) les algorithmes qui extraient des règles d'association périodiques ou cycliques (des règles qui associent un ensemble de comportements qui ont une apparition périodique).

8.2.2 Environnement dynamique

Limites d'une approche intégrant une méthode classique

Dans cette section nous présentons la motivation de ce travail, une analyse des limites à l'adaptation des méthodes statiques d'extraction de motifs séquentiels dans le contexte des flux de données et, ensuite, la méthode que nous proposons.

Idée générale

Dans les applications concernant les flux les données peuvent entrer et sortir (suite à des opérations d'élagage) de manière dynamique et en très grandes quantités. De plus, on a souvent besoin d'obtenir les résultats en temps réel. Un exemple typique d'application se trouve dans la sécurité qui implique une détection et une réaction rapides (dans le cas de la détection d'une attaque pirate sur un site, la détection d'une fraude, etc). Les applications qui concernent la détection des motifs séquentiels dans les flux de données sont donc nombreuses et néanmoins, ce problème ne connaissait pas d'algorithme approprié en 2005. En effet, des algorithmes comme GSP, PSP, PrefixSpan extraient les motifs séquentiels à partir des données stockées statiques (base de données ou fichiers). Leur idée générale est de chercher les sous séquences fréquentes de taille n et puis de trouver celles de taille n+1 etc (méthode « générer-élaguer »). Puisqu'il existe des méthodes d'extraction de motifs séquentiels pour le cas statique, notre première piste a été d'intégrer une telle méthode dans le cas des flux de données. Après une étude nous sommes arrivés à la conclusion que cette intégration soulèverait un certain nombre de problèmes et, par conséquence, ne serait pas réaliste. Ces problèmes sont traités dans la section suivante.

Analyse des limites

L'intégration des méthodes d'extraction de motifs séquentiels à partir des données stockées statiques (GSP, PSP, PrefixSpan, Spam) dans un environnement dynamique soulève un certain nombre de problèmes. Je vais illustrer ces problèmes en prenant le cas de l'algorithme PSP Masseglia *et al.* (1998). Si on essaie d'intégrer l'algorithme PSP dans le contexte des flux, on va constater un problème potentiel de limite de mémoire. La cause de ce problème réside dans la manière de générer les candidats. Pour obtenir un fréquent de taille "n", on doit générer tous les sous-ensembles de ce fréquent. Par exemple, avec une séquence fréquente de taille 100 (i.e. $< (1)(2)\dots(99)(100) >$), PSP va générer $2^{100} - 1$ candidats. De cette manière, et avec un support petit, on risque une complexité en mémoire exponentielle. Les expérimentations faites montrent ce phénomène. On a expérimenté l'algorithme PSP avec une base de données composée seulement de deux séquences identiques où la séquence était une répétition de l'itemset $< (1)\ (2) >$. Par exemple pour 11 répétitions on a les séquences

$s_1 =< (1)(2)(1)(2)\ldots(1)(2) >$, et $s2 = s1$ avec longueur $(s1) = 22$. Pour différentes longueurs de séquences et pour chaque étape de génération, on a obtenu les temps de réponses illustrés par la figure 8.1 (l'abscisse donne la longuer des candidats). Par exemple, avec une séquence fréquente de longueur 26, PSP génère 190000 candidats de longueur 19. Pour une longueur des séquences de 28 (donc 14 répétitions d'itemset (1)(2)) la mémoire disponible a été dépassée et le programme a été arrêté. Le même phénomène a été remarqué avec l'algorithme PrefixSpan où, pour d'autres types de séquences, l'espace disque permettant les re-écritures de la base peut se trouver rapidement saturé. On doit envisager que de telles séquences existent dans un des batches de transactions et l'exécution normale du programme en serait alors compromise. Ce phénomène n'est pas pénalisant dans les cas classiques du data mining (ou les motifs fréquents sont souvent courts), mais risque de bloquer une machine gérant des flux de données si un tel cas est présent (ce qui serait contraire aux contraintes imposées par les flux). Cela peut se produire dans le cas (connu) de requêtes répétées pour une URL sous la forme de document PDF ou PHP par exemple.

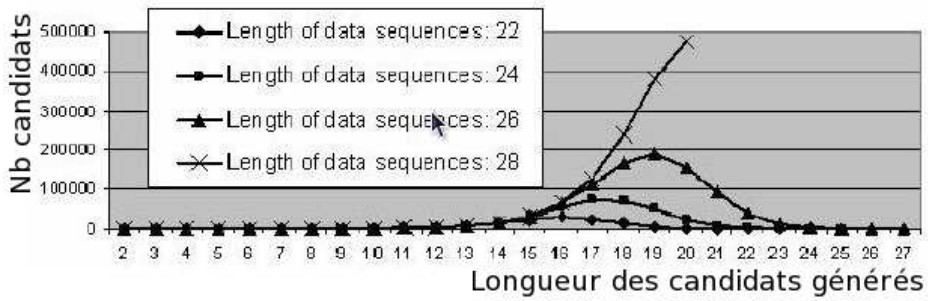


FIG. 8.1: Limites de l'algorithme PSP

Approches existants à présent dans la littérature

Nous présentons ici quatre approches proposées dans la littérature pour l'extraction de motifs séquentiels dans les flux de données.

Dans Raïssi *et al.* (2005); Raissi *et al.* (2006) les auteurs proposent SPEED, une méthode d'extraction de motifs séquentiels dans les flux de données. SPEED considère des batches de séquences constitués à partir du flux (*i.e.* un batch contient n séquences

de tailles variables). SPEED organise l'espace mémoire avec une structure de données qui associe un arbre (afin de représenter les séquences) avec des régions qui regroupent les séquences du flux en fonction de leurs inclusions. En d'autres termes, si une séquence α est incluse dans une séquence β alors α et β seront dans la même région et α sera un noeud descendant de β dans l'arbre. Quand une nouvelle séquence arrive, l'arbre est mis à jour afin de la stocker en mémoire dans la bonne région. L'espace mémoire étant limité, SPEED utilise comme méthode d'approximation un élagage des séquences les moins fréquentes et des items les plus anciens. Pour cela, SPEED utilise une structure de fenêtre logarithmique permettant de savoir quels items supprimer.

Dans Raïssi & Poncelet (2007), les auteurs présentent une méthode d'échantillonnage pour extraire les motifs séquentiels d'un flux de données. Les données traitées par Raïssi & Poncelet (2007) sont des séquences d'itemsets. Cette méthode est basée sur le principe de l'échantillonnage par réservoir. La technique d'échantillonnage est biaisée pour donner plus d'importance aux événements les plus récents. Le principal avantage de ce travail est de prendre en considération l'aspect incrémental de la construction des séquences dans un flux. En effet, les séquences ne sont pas organisées en batches de manière naturelle dans le déroulement du flux et il peut être préjudiciable de "couper" une séquence (par exemple avec une fenêtre temporelle) pendant qu'elle se constitue. Pour répondre à ce problème, les auteurs ont mis en place une liste noire (BL) contenant les identifiants des séquences qui sont sorties du réservoir suite au tirage aléatoire. Quand une séquence est sortie du réservoir, elle reste dans la BL pendant un nombre d'étapes égal à sa taille. L'extraction des motifs séquentiels est basée sur la notion de support minimum. Cette extraction repose sur un appel à prefixSpan Pei *et al.* (2001c).

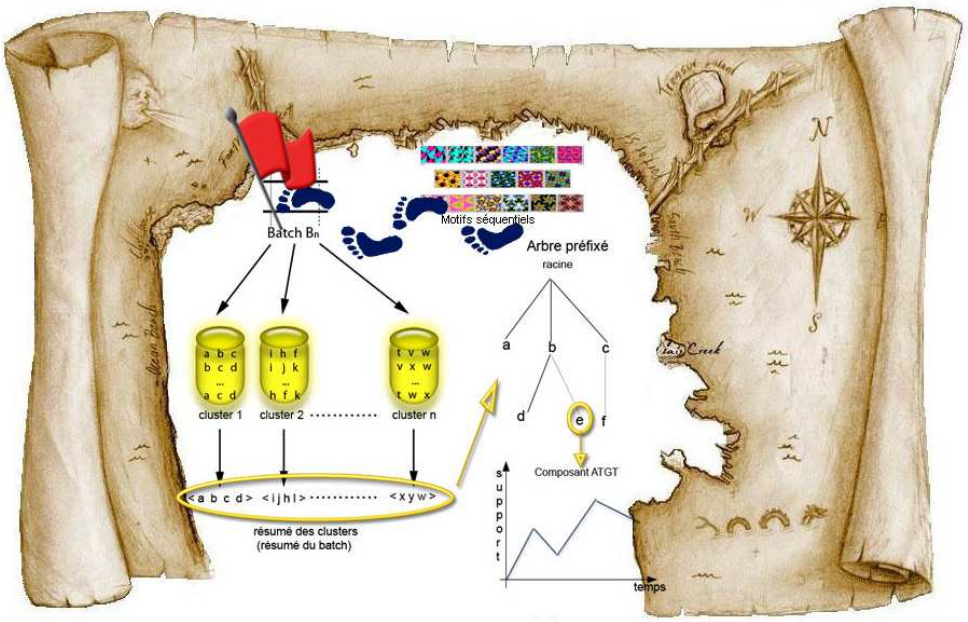
Raïssi & Plantevit (2008) propose une technique d'extraction de motifs séquentiels multidimensionnels. Un motif séquentiel multidimensionnel est composé d'une suite d'itemsets multidimensionnels. Un itemset multidimensionnel contient des informations prenant leurs valeurs sur plusieurs domaines possibles. Par exemple, le motif $< \{(CD, NY), (CD, NY)\}\{(DVD, LA)\} >$ s'interprète comme l'achat d'un CD à New-York répété deux fois dans la journée suivi, plus tard, par l'achat d'un DVD à Los-Angeles. La méthode MDSDS Raïssi & Plantevit (2008) traite le flux en le découpant en batches de taille fixe. Chaque batch contient donc un ensemble de séquences d'itemsets multidimensionnels. Un itemset multidimensionnel étant lui-même composé d'items multidimensionnels. Un item multidimensionnel est un $m-$uplet défini sur sa

dimension d'analyse. Par exemple (CD, NY) est un item à deux dimensions (nature de l'achat et lieu de l'achat). Pour répondre au problème de l'explosion combinatoire lié à la nature de ces données, MDSDS propose de réaliser l'extraction de motifs fréquents en deux temps. Dans un premier temps, les items les plus spécifiques sont extraits. Ils serviront de base pour une extraction de motifs ultérieure. Ces items sont obtenus en fonction du support donné par l'utilisateur. Ensuite, une extraction de motifs séquentiels est obtenue par un appel à PrefixSpan Pei *et al.* (2001c). Enfin les motifs fréquents sont gérés par une structure d'arbre préfixé dans laquelle chaque noeud est associé à une table temporelle (de manière similaire à l'algorithme présenté dans la séction 13.2.2 et à celui de Giannella *et al.* (2003)).

Enfin, Chen *et al.* (2005) considère un flot sous la forme de plusieurs séquences produisant des données. Un itemset y est défini comme l'ensemble des items à l'intersection de toutes les séquences pour une estampille temporelle précise. Les séquences sont découpées en plusieurs fenêtres de taille fixe et la fréquence d'un motif est calculée en fonction du nombre de fenêtres qui supportent ce motif. L'extraction des motifs est réalisée grâce à une modification de l'algorithme PrefixSpan.

Après cette revue des algorithmes qui se sont intéressés à l'extraction de motifs séquentiels, le chapitre suivant fait un survol des principales approches de traitement de flux de données existantes dans la littérature et justifie le choix du modèle utilisé.

CARTE GENERALE



Dans le prochain chapitre, je fais un survol des principales méthodes de traitement des flux de données existantes dans la littérature et j'explique le choix de notre modèle.

CHAPITRE 9

APPROCHES DE TRAITEMENT DES FLUX DE DONNEES

9.1 Méthodes de traitement des flux de données

CE CHAPITRE s'intéresse à la présentation des approches de traitement des flux de données existantes dans la littérature. Après une introduction des principaux travaux relatifs aux résumés de grandes quantités de données en contexte statique ou dynamique, j'explique le choix de notre modèle basé sur un découpage par batches et sur la classification non-supervisée.

Il existe un attribut, le temps, qui définit l'ordre des données dans un flux de données. À chaque élément est associé une estampille temporelle et en fonction de cette information, on peut distinguer trois types d'approches de traitement des flux de données :

- *agglomératif (ou itératif)* : les données sont traitées par ordre d'apparition, une après l'autre ;
- *par paquets* : les données sont groupées dans des paquets d'une même taille fixe et ensuite les paquets seront traités l'un après l'autre. Quelques exemples d'algorithmes appliquant cette stratégie se trouvent dans Chang & Lee (2003b); Cheng

et al. (2008); Giannella *et al.* (2003); Li *et al.* (2008); Manku & Motwani (2002) ;

– *à l'aide des fenêtres glissantes* : une fenêtre de taille fixe dont le premier point glisse dans le temps ou à l'arrivée de nouvelles données. Chang & Lee (2003a, 2004); Chi *et al.* (2006) sont des exemples d'algorithmes qui utilisent ce type de fenêtres.

Selon la manière dont la taille de la fenêtre (ou du paquet) est établie, les deux derniers types se divisent à leur tour dans deux sous-types :

– *taille établie selon le nombre de données* ;

– *taille établie selon l'intervalle de temps.*

La figure 9.1 illustre cette classification.

Les flux de données peuvent être classifiés en fonction de la stratégie de mise en jour de la manière suivante :

– *mise à jour par transaction* : L'opération de mise à jour se fait après le traitement de chaque transaction ;

– *mise à jour par paquet* : D'abord on traite les données de tout le paquet (ou fenêtre) et ensuite on fait les mises à jour.

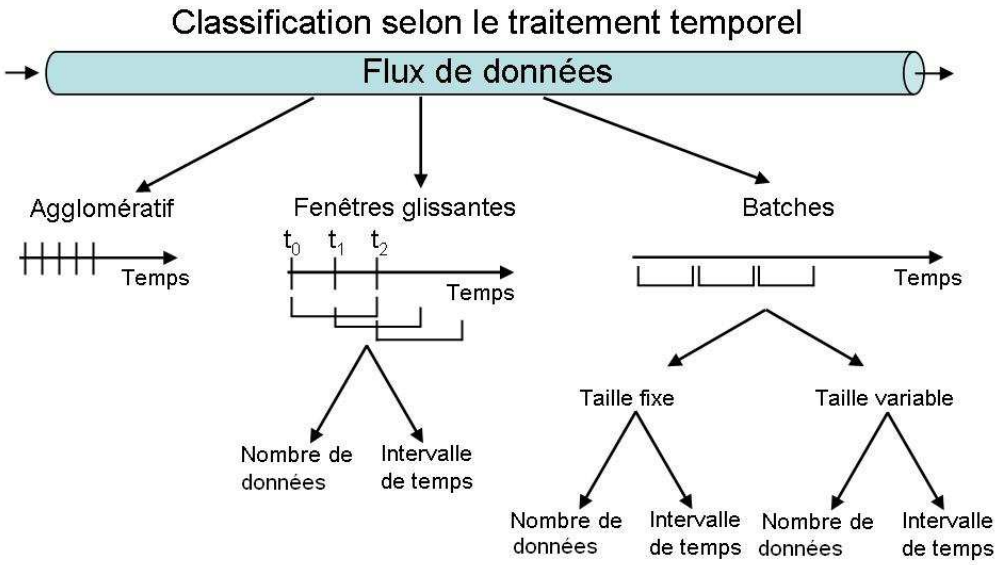


FIG. 9.1: Classification des méthodes de traitement des flux de données selon l'estampille temporelle

9.2 Résumés de grandes quantités de données : du statique au dynamique

Barbar'a *et al.* (1997) offre une étude intéressante des méthodes de réduction concernant les grandes bases de données. Ils divisent les techniques de réduction de données en deux classes :

- techniques paramétriques : Ces techniques assument l'existence d'un modèle pour les données et elles estiment les paramètres du modèle ;
- techniques non-paramétriques : Ces techniques n'assument l'existence d'aucun modèle.

Barbar'a *et al.* (1997) cite comme exemple de technique paramétrique les techniques basées sur la décomposition en valeurs singulières, sur la transformée discrète en ondelettes, les modèles basés sur la régression et les modèles log-linaires. En ce qui concerne les techniques non-paramétriques, on trouve les approches basées sur les histogrammes, sur des algorithmes de clustering et des arbres d'indices.

Pour prendre en compte les données dynamiques, ces méthodes conservent une grande partie des propositions faites sur les données statiques ; toutefois, les contraintes des flux laissent leurs empreintes. L'impossibilité de scanner plusieurs fois les données fait son apparition et on note surtout un degré d'approximation accru. Garofalakis *et al.* (2002); Yang (2008)divise les techniques de traitement flux de données selon la caractéristique approximative de leurs réponses : avec des limites déterministes et avec des limites probabilistes. Celles qui ont des limites déterministes, font une estimation bornée de l'erreur de calcul de la réponse, alors que celles dont les limites sont probabilistes calculent les réponses avec des probabilités. Cheng *et al.* (2008) s'est intéressé à une classification des algorithmes d'extraction d'itemsets fréquents dans les flux et proposent une classification en algorithmes exacts et approximatifs. Ils classifient ensuite les algorithmes approximatifs en faux-positifs et faux-négatifs.
La grande majorité des méthodes qui traitent les flux de données calcule un résumé (synopsis en anglais) des données et ensuite, sur la base de ce résumé, elles calculent des réponses approximatives (Han & Kamber (2001)). En fonction de ces critères, Garofalakis *et al.* (2002); Han & Kamber (2001); Yang (2008) classifient les méthodes selon qu'elles utilisent : de l'échantillonnage (aléatoire), des histogrammes, des ondelettes, des

sketches, des tableaux de hachage, des fenêtres glissantes, des modèles multi-résolutions et des algorithmes aléatoires. A cette liste s'ajoute aussi le clustering, une méthode très utilisée dans le traitement des flux. Dans la suite, je propose un survol de certaines de ces méthodes.

Histogramme

Un histogramme offre une image très suggestive de la distribution des fréquences des données en utilisant des rectangles. La hauteur d'un rectangle donne la fréquence d'apparitions des données correspondantes à sa largeur. Un histogramme est composé d'une suite de tels rectangles. En fonction des caractéristiques des rectangles, il existe plusieurs types d'histogrammes, dont les plus utilisés sont les histogrammes V-optimaux et les histogrammes à largeurs égales. Kooi (1980); Poosala (1997) sont quelques exemples de méthodes qui utilisent les histogrammes.

Echantillonnage

Ce type de méthodes garde un échantillon des données considérées comme représentatives pour l'ensemble entier des données. Une méthode fréquemment utilisé de ce type est le "Reservoir Sampling" de Vitter (1985). Cette méthode considère un réservoir de taille fixe, n ; la taille totale du flux est supposée inconnue. Quand les éléments du flux passent, ils remplacent les éléments du réservoir avec une probabilité de n/N. Gibbons & Matias (1998) propose une forme d'échantillon concis, "concise sampling", qui garde un compteur pour les valeurs qui apparaissent plus d'une fois. Ces éléments ont la forme suivante : $< valeur, compteur >$. Pendant l'exécution, si un nouvel élément est déjà contenu dans l'échantillon, on incrémente son compteur. L'algorithme utilise un seuil τ et chaque nouvel élément a la probabilité 1τ d'être ajouté à l'échantillonnage ; pendant l'exécution, une fois atteinte la valeur du seuil, on augmente à une nouvelle valeur τ' du seuil et on décrémente les compteurs des éléments contenus dans l'échantillonnage avec une probabilité de $\tau\tau'$. Gibbons (2001); Manku & Motwani (2002) proposent également des algorithmes basés sur l'échantillonnage.

Ondelettes

La transformée en ondelette est présentée en détail dans la section 14.2.2. Matias *et al.* (1998) propose une méthode qui utilise les ondelettes afin de construire des histogrammes. Dans une étape de prétraitement, ils forment une distribution des données cumulative étendue à partir des données d'origine ou d'un échantillon des données d'origine. Ensuite, ils utilisent les ondelettes et gardent seulement les coefficients les plus significatifs qui, liés aux indices, forment un histogramme. Cet histogramme est utilisé dans le processus de reconstruction des données d'origine avec un certain degré d'approximation.

Sketches

Alon *et al.* (1996) introduit le concept de résumé (sketch) aléatoire. L'idée est de garder un résumé des données en projetant chaque donnée dans un certain espace via des fonctions de hachage et de maintenir seulement les composants les plus définitoires. Certains exemples de méthodes qui utilisent des sketches sont : Alon *et al.* (1996, 1999); Cormode & Muthukrishnan (2004); Dobra *et al.* (2002); Gilbert *et al.* (2001).

Clustering

La diversité des méthodes de classification est tellement importante que, pour la décrire, il faudrait lui consacrer beaucoup plus d'espace que cette section. Nous voulons présenter ici le cadre général et les principaux types de méthodes de classification. L'objet de la classification est de grouper les données dans des classes qui contiennent des données similaires ; un tel groupe porte le nom de « cluster ». Trois des principales causes de cette grande diversité des méthodes de classification sont : les multiples domaines d'applications, les nombreuses formes que peuvent prendre les clusters (impliquant beaucoup de types de méthodes de traitement afin de traiter chaque type de cluster) et l'intégration des méthodes de classement dans d'autres cadres, plus généraux (comme une étape dans des algorithmes).

Les racines de la classification sont représentées par les statistiques, l'analyse numérique et la théorie de la décision. Parmi les domaines d'application on peut citer : la

reconnaissance des objets, la segmentation des images, la fouille de texte, la recherche d'information, le marketing ou encore les diagnostics médicaux.

Les méthodes de classifications sont divisées dans deux grandes catégories :

- les méthodes de classification non-supervisées (clustering) : ce type de méthodes ne dispose d'aucune information apriori sur les classes à obtenir. Jain *et al.* (1999) synthétise plusieurs appellations plus au moins exactes sous lesquelles le clustering apparaît dans la littérature : taxonomie numérique Sneath & Sokal (1973), quantification vectorielle Oehler & Gray (1995) ou apprentissage par observation Michalski & Stepp (1983) ;
- les méthodes de classification supervisées : dans ce cas, on utilise des informations connues à priori sur les classes. Ce type de classification porte aussi le nom d'analyse discriminante.

Catégoriser les méthodes de classification est un sujet très étudié, mais étant donnée la multitude de types d'algorithmes de classification, il est très difficile d'arriver à les classer. De plus, chaque domaine impose son contexte sur les notions utilisées ce qui augmente la difficulté liée à l'écriture d'un résumé de ces méthodes trouvant leurs origines dans des domaines distincts. Pour une lecture approfondie sur les méthodes de classification nous recommandons Berkhin (2002); Han & Kamber (2001); Jain *et al.* (1999); Xu & Ii (2005).

Avec la classification supervisée, la difficulté est liée à la production ou l'obtention de données labélisées. Outre son indépendance vis-à-vis des données labélisées, la classification non-supervisée présente quelques avantages dans notre contexte : adaptation aux événements nouveaux, meilleur traitement des classes qui évoluent dans le temps, adaptable dans un contexte dynamique (où on ne peut pas avoir des connaissances apriori sur les classes finales à obtenir). Dans le cas des flux, la classification non-supervisée (le clustering) paraît donc appropriée. En effet, dans un flux on ne connaît qu'un résumé des données passées et les données actuelles ; on ne dispose pas de connaissance sur les données suivantes et en conséquence, on ne peut pas faire des suppositions sur les classes finales (les clusters). Toutefois, il est possible de bénéficier des connaissances sur les comportements passés car il y a une forte probabilité qu'ils se répètent dans le futur. Une de nos méthodes, proposée dans cette thèse, utilise cette idée.

9.3 Motivation du modèle de traitement choisi

Afin de résoudre le problème d'extraction de motifs séquentiels dans les flux de données, nous avons choisi dans le cadre de cette thèse de couper le flux dans des paquets de taille fixe.

Premièrement, parce qu'on est souvent intéressé par le résultat à la fin de certains intervalles temporels : évaluer les informations sur les achats d'un supermarché chaque jour le soir, les informations collectées par un satellite chaque heure, les comportements des utilisateurs d'un site chaque demi-journée etc. La valeur de ces intervalles peut changer dynamiquement, par exemple, en cas d'un passage d'une comète on est intéressé par les informations reçues à des granularités de temps très fines et afin de répondre à cette contrainte il suffit d'ajuster la valeur de la taille du paquet.

Deuxièmement, en traitant plusieurs données à la fois, si on fait un groupement des données similaires contenues dans un paquet, on gagne du temps en comparaison du traitement des données une par une.

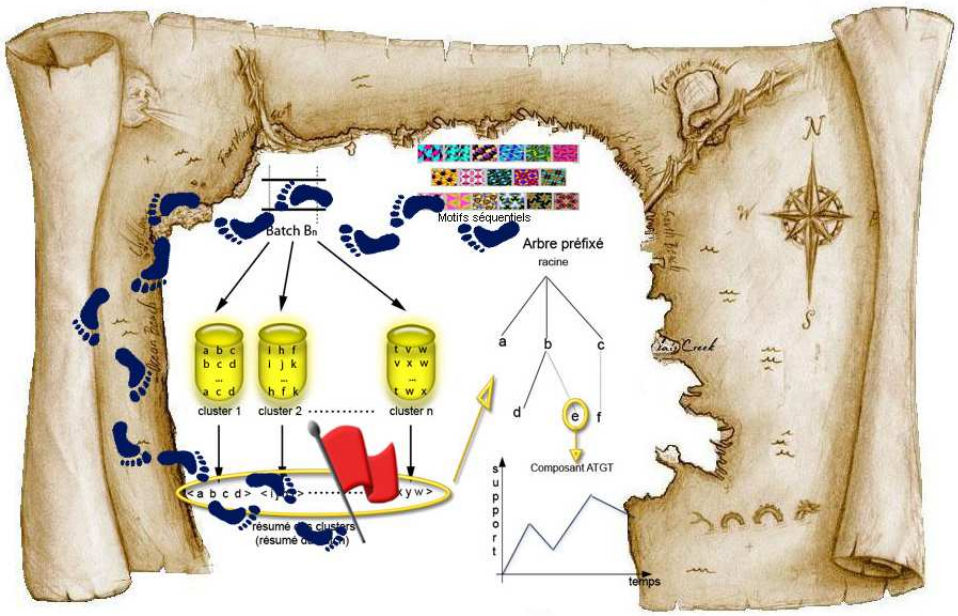
Troisièmement, il y a beaucoup plus de possibilités d'adaptation d'une méthode mathématique existante pour le cas statique, en raison du fait que la grande majorité des méthodes n'a pas été conçue pour prendre en compte le cas incrémental.

Quatrièmement, le coupage en batches nous permet de simuler certaines des conditions du cas statique.

Comme méthode de traitement, nous avons choisi d'utiliser la classification non-supervisée. Ce choix est motivé par sa parfaite adaptation aux autres parties de notre méthode globale. En effet, notre méthode a besoin de s'appliquer sur des groupes de données similaires, ce qui peut être obtenu suite à l'application d'une classification non-supervisée. On a opté pour la classification non-supervisée face à celle supervisée, pour une meilleure prise en compte des nouveautés.

On a vu lors de ce chapitre les différentes approches de traitement des flux de données, notre choix d'utiliser le découpage des batches et d'utiliser la classification non-supervisée. Notre but est de grouper, pour chaque batch, les données similaires via une classification non-supervisée et ensuite de les résumer cluster par cluster. Afin de pouvoir résumer les données d'un cluster, le chapitre suivant passe à la présentation d'une méthode capable de résumer des séquences.

CARTE GENERALE



Dans le chapitre qui suit, nous proposons une méthode permettant de résumer les données d'un flux de données. Cette méthode est basée sur un algorithme d'alignement des séquences.

CHAPITRE 10

ALIGNEMENT DE SEQUENCES ET MOTIFS SEQUENTIELS APPROXIMATIFS

10.1 Description du problème et motivation du choix

ON A VU l'importance de résumer les données d'un flux et les différentes méthodes qui ont été proposées dans ce but jusqu'à présent. Nous proposons lors de cette thèse une méthode qui permet de résumer les données grâce à la programmation dynamique. Notre objectif est de proposer une méthode qui puisse s'adapter dans cet environnement dynamique et qui soit capable de résumer de telles séquences de manière incrémentale, avec une bonne qualité de résultat et tout en respectant les contraintes temporelles. Nos travaux s'inspirent d'un algorithme d'alignement de séquences proposé par Birgit *et al.* (2003); Kum (2004) pour la fouille de bases de données statiques. On a choisi cet algorithme d'alignement parce qu'il permet d'obtenir un résumé de plusieurs séquences en alignant les séquences deux par deux et en raison de sa bonne complexité temporelle et spatiale. En effet, l'alignement de deux séquences de tailles m et n se fait avec une complexité spatiale de O(nm) et temporelle de O(nm). En ce qui concerne la complexité pour un batch de séquences, considérons que toutes les séquences d'un cluster C ont une longueur m et $|C| = p$. La complexité de l'alignement pour un batch

est de $O(p.m^2)$.

Ce chapitre commence avec une présentation générale du principe d'alignement très connu de la programmation dynamique. Ensuite, on peut voir une illustration de cet algorithme sur un ensemble de séquences et enfin, une discussion concernant les avantages de l'utilisation de cette méthode.

10.2 Principe général de l'alignement de deux séquences

```
GAATTCAG
|  |  | |  |
GGA–TC–G
```

FIG. 10.1: Alignement de deux séquences

La programmation dynamique est une méthode de construction graduelle de la solution en appliquant des principes récurrents Cormen *et al.* (2001). Un grand nombre d'algorithmes utilisent la programmation dynamique, parmi lesquels les algorithmes d'alignement de séquences.

Definition 10.1 *Un alignement (global) de deux séquences S_1 et S_2 est obtenu par l'insertion d'abord d'espaces (ou de tirets) soit à l'intérieur soit à la fin de S_1 et S_2 et ensuite par le positionnement des deux séquences obtenues l'une au-dessus de l'autre afin que chaque caractère ou espace dans chaque séquence soit en opposition avec un seul caractère ou espace de l'autre chaîne.Gusfield (1997)*

Donc, aligner deux séquences de tailles potentiellement différentes suppose de les imaginer écrites l'une sous l'autre, de les séparer dans les éléments composants et d'insérer des espaces vides entre les éléments composants (au début, au milieu ou à la fin) afin que les éléments identiques soient alignés sur les mêmes positions verticales. L'insertion d'espaces vides sur une même position verticale dans les deux séquences n'est pas considérée. A la fin de ce processus d'alignement, les deux séquences auront la même taille. Un exemple d'alignement des séquences peut être suivi dans la figure 10.1.

Le problème de l'alignement peut être vu comme un problème de transformation d'une séquence A dans une autre séquence B. Sous une forme récursive, une séquence A connait une édition optimale dans la séquence B par :

- L'insertion du premier élément de B et la continuation de l'alignement de A et du reste de B ;
- L'insertion du premier élément de A et la continuation de l'alignement du reste de A et de B ;
- Le remplacement du premier élément de A avec le premier élément de B et la continuation de l'alignement du reste de A avec le reste de B.

Le pseudo-code de l'algorithme d'alignement que nous utilisons est transcrit dans la figure 10.2. L'alignement est obtenu étape par étape ; les résultats intermédiaires étant retenus dans une matrice. Le calcul de l'alignement de deux séquences comporte trois étapes :

- *Initialisation* : Dans cette étape une matrice de $M+1$ colonnes et $N+1$ lignes est initialisée, où M est la taille de la première séquence et N la taille de la deuxième séquence. L'algorithme que nous utilisons fait une initialisation de la première ligne et de la première colonne avec des valeurs qui augmentent de 1 à chaque pas, en commençant à 0.
- *Remplissage de la matrice* : En commençant avec le coin en haut à gauche, on remplit les cellules de la matrice selon la relation de récurrence illustrée dans la figure 10.2. Dans la figure 10.2, v_1 et v_2 sont les éléments courants dont on étudie l'alignement.
- *Traçage en arrière* : En commençant avec l'élément de la matrice de coordonnées M+1 et N+1, on calcule le chemin en arrière en revenant vers le point d'origine (en haut à gauche). Pour cela on regarde les valeurs des cellules voisines (à gauche, en diagonale et en haut) et on choisit celle qui minimise le chemin parcouru. Pour le résultat de l'alignement, chaque pas ainsi effectué signifie soit un espace vide dans une des séquences, soit un regroupement des éléments correspondants.

10.3 Résumé des séquences

Comme on l'a vu dans les sections 4.1 et 3.1 un flux de données est un flux de séquences où chaque séquence est une liste ordonnée non vide, d'itemsets notée

$for(i = 0 \quad to \quad length(S_1)$
$\quad D_{i,0} \leftarrow i$
$for(j = 0 \quad to \quad lenght(S_2)$
$\quad D_{0,j} \leftarrow j$
$fori = 1 \quad to \quad length(S_1)$
$forj = 1 \quad to \quad length(S_2)$
{
$\quad Choix_1 \leftarrow D_{i-1,j} + 1$
$\quad Choix_2 \leftarrow D_{i,j-1} + 1$
$\quad Choix_3 \leftarrow D_{i-1,j-1} + 1 - 2 * \frac{v_1 \cap v_2}{|v_1|+|v_2|}$
$\quad D(i,j) \leftarrow min(Choix_1, Choix_2, Choix_3)$
}

FIG. 10.2: Algorithme d'alignement

$< s_1 s_2 ... s_r >$, avec s_j un itemset. Pour un ensemble de k séquences, l'algorithme d'alignement sera appliqué successivement k-1 fois. La séquence résultant de l'alignement de deux séquences a reçu le nom de *séquence pondérée*, selon Kum (2004). Le résultat de l'alignement d'une séquence pondérée avec une séquence simple porte aussi le nom de séquence pondérée.

L'alignement des séquences renvoie une séquence pondérée du type : $SA =< I_1 : n_1, I_2 : n_2, ..., I_r, n_r >: m$. Dans cette représentation, m représente le nombre total de séquences impliquées dans l'alignement. I_p $(1 \leq p \leq r)$ est un itemset représenté sous la forme $(x_{i_1} : m_{i_1}, ...x_{i_t} : m_{i_t})$, où m_{i_t} est le nombre de séquences qui contiennent l'item x_i à la p^{eme} position dans la séquence pondérée. Enfin, n_p est le nombre d'occurrences de l'itemset I_p dans l'alignement. L'exemple 4 décrit le processus d'alignement de quatre séquences. À partir de deux séquences, l'alignement commence par insérer des itemsets vides (au début, au milieu ou à la fin des séquences) jusqu'à ce que les deux séquences contiennent le même nombre d'itemsets.

Exemple 4 *Considérons les séquences suivantes :* $S_1 =<$ *(a,c) (e) (m,n)* $>$, $S_2 =<$ *(a,d) (e) (h) (m,n)* $>$, $S_3 =<$ *(a,b) (e) (i,j) (m)* $>$ *et* $S_4 =<$ *(b) (e) (h,i) (m)* $>$. *Les étapes conduisant à l'alignement de ces séquences sont détaillées dans la figure 10.3. Tout d'abord, un itemset vide est inséré dans* S_1. *Ensuite* S_1 *et* S_2 *sont alignées dans le but de produire* SA_{12}. *Le processus d'alignement est alors appliqué entre* SA_{12} *et* S_3. *La*

méthode d'alignement continue à traiter les séquences deux par deux jusqu'à la dernière séquence.

À la fin du processus d'alignement, la séquence pondérée (SA_{14} dans la figure 10.3) est un résumé de toutes les séquences initiales. Le **motif séquentiel approximatif** peut être obtenu en spécifiant k : le nombre minimum d'occurrences d'un item pour que celui-ci soit considéré comme fréquent. Par exemple, avec la séquence SA_{14} de la figure 10.3 et $k = 2$, la séquence pondérée filtrée sera : <(a,b)(e)(h,i)(m,n)> (ce qui correspond aux items qui ont un nombre d'occurrences supérieur ou égal à k).

Etape 1 :				
S_1 :	<(a,c)	(e)	()	(m,n)>
S_2 :	<(a,d)	(e)	(h)	(m,n)>
SA_{12} :	(a :2, c :1, d :1) :2	(e :2) :2	(h :1) :1	(m :2, n :2) :2
Etape 2 :				
SA_{12} :	(a :2, c :1, d :1) :2	(e :2) :2	(h :1) :1	(m :2, n :2) :2
S_3 :	<(a,b)	(e)	(i,j)	(m)>
SA_{13} :	(a :3, b :1, c :1, d :1) :3	(e :3) :3	(h :1, i :1, j :1) :2	(m :3, n :2) :3
Etape 3 :				
SA_{13} :	(a :3, b :1, c :1, d :1) :3	(e :3) :3	(h :1, i :1, j :1) :2	(m :3, n :2) :3
S_4 :	<(b)	(e)	(h,i)	(m)>
SA_{14} :	(a :3, b :2, c :1, d :1) :4	(e :4) :4	(h :2, i :2, j :1) :3	(m :4, n :2) :4

FIG. 10.3: Etapes de l'alignement de séquences

10.4 Discussion

Les avantages de cette méthode sont multiples. Premièrement, cette méthode permet la spécification du nombre d'occurrences d'un item pour qu'il soit considéré comme fréquent et la séquence pondérée résumée adapte sa forme à cette valeur. Ceci présente l'avantage majeur de cette méthode, car cette propriété sera exploitée au maximum pendant l'exécution du programme dans le cadre d'un flux de données. Plus les données du flux sont abondantes, plus la valeur de paramètre augmente en filtrant plus les

données et en accélérant la vitesse de traitement. Bien sûr, plus grande est la valeur de ce paramètre, moins les résultats sont précis, mais le risque de blocage du flux est plus important car il mènerait à une perte irrécupérable de données.

Deuxièmement, sa complexité temporelle de O(nm), avec n et m les tailles des séquences à aligner, permet son intégration dans un environnement dynamique. Afin d'améliorer cette complexité temporelle, d'autres versions d'algorithmes d'alignement moins coûteux temporellement existent et leur intégration est plausible dans notre cadre général.

Troisièmement, une séquence pondérée permet d'être alignée de nouveau avec une nouvelle séquence. Ce côté incrémental convient très bien au contexte des flux, où les séquences arrivent au fil de l'eau. Toutefois, l'ordre des séquences met une empreinte sur le résultat, une technique d'optimisation de la qualité du résultat sera présentée dans la section 13.2.2.2.

Il existe plusieurs solutions alternatives de l'alignement, toutes avec un même score d'alignement. En conséquence, le résultat n'assure pas l'unicité de la solution. Il serait intéressant d'étudier des pistes qui puissent arriver à une optimisation du résultat trouvé, pas seulement du point de vue du score, mais aussi du point de vue de la signification du résultat comme motif séquentiel approximatif et selon certaines relations sémantiques entres les éléments dépendants du domaine d'application. Par exemple, dans une application agro-alimentaire, on peut définir une classe pomme={golden, gala, granny}. Dans une telle application, le taux d'acceptation de ces fruits dans un même item serait plus faible. Une version d'alignement sémantique trouverait de nombreuses applications.

```
Global   FTFTALILLAVAV
         F--TAL-LLA-AV

Local    FTFTALILL-AVAV
         --FTAL-LLAAV--
```

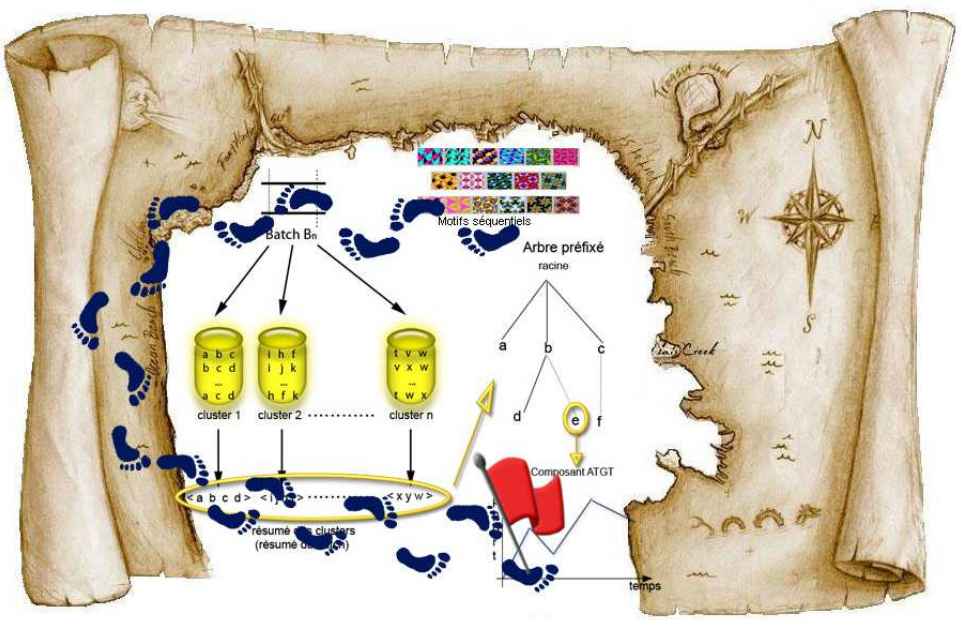
FIG. 10.4: Alignement global/local (source Wikipedia)

Les algorithmes d'alignements peuvent se grouper en deux parties : ceux qui créent un alignement global et ceux qui créent un alignement local. Un alignement global essaie d'aligner *tous* les éléments de la première séquence avec *tous* les éléments de la

deuxième séquence, alors que l'alignement local s'intéresse plus aux correspondances optimales locales (figure 10.4). L'algorithme de Needleman-Wunsch est représentatif du premier cas et l'algorithme Smith and Waterman du deuxième. Selon les applications, une des versions peut être préférée.

Après la présentation d'un algorithme d'alignement, on passe dans le chapitre suivant au traitement de l'historique des motifs séquentiels approximatifs.

CARTE GENERALE



Le chapitre suivant contient la description d'une nouvelle stratégie de gestion de l'historique des motifs extraits dans un flux - REGLO (Résumé à Erreur Globale Optimisée). Tout d'abord on définit les problèmes ; chaque historique est vu comme une série temporelle et les séries temporelles sont mises en compétition afin de gagner plus d'espace de mémoire. Ensuite on présente les travaux existants dans la littérature. Toujours en train de mieux répondre aux contraintes de temps et d'espace, on continue avec la présentation de nos contributions. La première contribution consiste dans une nouvelle stratégie de gestion de l'historique des motifs extraits dans un flux. La deuxième contribution réside dans une simplification significative des formules classiques de calcul de la régression linaire et des calculs des erreurs de compression. Cette simplification se traduit par un gain important en vitesse. Enfin, la troisième contribution est une nouvelle approche rapide pour la fusion des segments et des calculs des erreurs de compression - AMi (Approximation par les Milieux). Afin de valider nos contributions, multiples expérimentations et comparaisons y sont présentées. On conclut ce chapitre par une discussion.

CHAPITRE 11

UNE NOUVELLE STRATEGIE DE GESTION DE L'HISTORIQUE DES MOTIFS EXTRAITS : REGLO ET AMI

11.1 Introduction

LES CONNAISSANCES extraites sur un flux ne peuvent pas être stockées en totalité et doivent être (au mieux) résumées. Il n'est alors pas toujours possible de retracer avec exactitude les variations (par exemple la valeur du support) des connaissances passées. Néanmoins, une réponse approximative très proche de la valeur demandée peut-être satisfaisante. Produire des résumés de séries temporelles, permettant de répondre à ce problème, est un sujet très étudié ces dernières années Chang & Lee (2003c); Chen *et al.* (2002); Cohen & Strauss (2003); Papadimitriou *et al.* (2005); Teng *et al.* (2003); Zhu & Shasha (2002). Considérons M, la quantité de mémoire disponible. Imaginons que nous souhaitions résumer le comportement de n séries temporelles et considérons que nous disposons d'une mémoire organisée sous la forme d'une matrice à n lignes. Nous pouvons alors stocker en mémoire au maximum $s = M/n$ valeurs pour chaque série. Après s itérations du flux la mémoire est pleine et nous devons appliquer une méthode pour résumer les séries et quand s devient très grand ce résumé ne peut pas se faire sans introduire un degré d'approximation. Ce degré d'approximation se tra-

duit alors par une erreur locale à chaque série et une erreur globale (cumul des erreurs locales) du modèle. Les questions qui se posent sont alors les suivantes : "quelle série doit être résumée pour minimiser l'erreur globale ?", "comment résumer une série pour minimiser l'erreur locale ?", "quel algorithme utiliser pour calculer l'approximation aussi vite que possible ?".

Exemple 5 *De nombreux sites Web souhaitent mesurer la fréquence des requêtes sur leurs URLs et la variation de cette fréquence dans le temps. Ces données ne peuvent toutefois pas être stockées étant donnés leur volume et la durée d'observation. Ainsi, un résumé fiable de ces données doit être calculé. Ce résumé doit permettre de montrer, pour chaque URLs observée, la fréquence des requêtes et son évolution dans le temps avec une qualité d'approximation aussi bonne que possible.*

Il existe de nombreuses techniques capables de répondre au problème illustré par l'exemple 5. Une première classe de techniques consiste à minimiser l'erreur locale. Plus précisément, quand la mémoire allouée à une séquence est saturée, cette série doit être compressée. Disons que cette séquence contient plus de s segments (observations) alors une compression de deux (ou plusieurs) segments doit être appliquée. Une autre classe de techniques utilise un facteur d'ancienneté des données. Dans ce cas, les segments les plus anciens sont considérés comme moins importants et, en conséquence, doivent être choisis pour la compression plus souvent que les segments récents. La majorité des solutions existantes dans les flux de données est basée sur la deuxième classe de méthodes (utilisant l'ancienneté)

La condensation des séries s'est faite jusqu'à présent indépendamment pour chaque séquence. Les algorithmes du domaine se sont concentrés sur des résumés séparés et indépendants Chen *et al.* (2002); Giannella *et al.* (2003); Zhu & Shasha (2002), en accordant à chaque série le même espace en mémoire. Toutefois, la gestion de cet espace mémoire est un sujet très important pour les flux de données et une stratégie accordant la même quantité de mémoire à chaque série n'est pas forcément appropriée. Nous proposons une vision nouvelle qui lance une compétition des séries vis à vis de l'espace de mémoire selon leur besoin de précision. La méthode proposée dans ce but est capable :

- de se restreindre à l'espace mémoire disponible ;
- de calculer une approximation efficace et fiable d'une série temporelle.

Il s'agit à la fois d'une stratégie d'optimisation de l'erreur globale des résumés de plusieurs séries et d'une méthode rapide et fiable de compression des séries.

Notre contribution se situe également au cœur du problème de la condensation d'un série temporelle. Nous abordons la fusion de segments avec un nouveau point de vue. Avec la régression linaire (RL), la fusion de deux segments peut être calculée en O(1) (ainsi que l'erreur résiduelle). Toutefois, les articles existants dans le domaine sont basés sur des formules complexes impliquant des variables et des opérateurs nombreux. On propose tout d'abord de simplifier ces formules. Ensuite, compte tenu de la vitesse à laquelle les flux doivent être traités, nous proposons une technique d'approximation rapide qui donne des résultats similaires à ceux de la RL tout en utilisant beaucoup moins de variables et d'opérateurs. AMi (Approximation par les Milieux), notre nouvelle technique d'approximation, utilise les milieux des segments pour les résumer.

Les principales contributions de ce chapitre sont :

1. Une technique d'approximation rapide et fiable basée sur une observation intuitive des propriétés de la RL (présentée en section 11.6.1).
2. Une mesure rapide et fiable de l'erreur due à cette approximation sur deux segments d'une série temporelle (présentée en section 11.6.2).
3. Une stratégie de minimisation de l'erreur globale pour allouer la mémoire aux résumés des séries temporelles en fonction des mesures données par les points (1) et (2).

Ce chapitre est organisé de la manière suivante. La section 11.2 donne les définitions liées à notre problème et la section 11.3 présente les techniques existantes. En section 11.4 nous présentons notre méthode minimisant l'erreur globale. Les nouvelles équations pour la RL sont présentées en section 11.5 et la section 11.6 expose notre nouvelle technique d'approximation. Enfin, la section 11.7 donne nos résultats d'expérimentations.

11.2 Définitions des problèmes

Cette section présente les concepts d'approximation optimisant l'erreur globale et définit nos objectifs de recherche.

11.2.1 Résumés optimisant l'erreur globale

Soit $T[1..n]$, un ensemble de n séries temporelles à observer. Soit $T[j]$ la j^{eme} série de T alors à l'étape s on a $T[j] = (T[j][1], ..., T[j][s]]$ comme ensemble des observations (valeurs) de $T[j]$. Soit M la quantité de mémoire disponible (en d'autres termes, on peut stocker M segments ou valeurs en mémoire). Nous considérons le cas de temps discrétisé. A chaque étape nous avons une valeur pour chaque série (la valeur par défaut étant zéro). La valeur maximale de s permettant de stocker les mesures sans compression est $s = M/n$. Quand $(s \times n) > M$, la représentation des séries doit être compressée et la perte d'information est liée à la différence entre ces deux nombres $(s \times n)$ et M.

Une technique de compression de séries temporelles bien connue se trouve dans la régression linéaire (RL). Soit $R[1..n]$, l'ensemble des représentations des n séries et $S[i]$ la taille de $R[i]$ (*i.e.* le nombre de segments utilisés par la représentation $R[i]$). La représentation $R[i]$ de la série i est une approximation de cette série en $S[i]$ segments avec comme contrainte globale $\sum_{i=1}^{n} S[i] = M$.

Soit $E[i]$, l'erreur de $R[i]$ par rapport aux données d'origine de la série i et $E(R)$, l'erreur globale de $R[1..n]$. Alors $E(R) = \sum_{i=1}^{n} E[i]$.

Les travaux existants se sont concentrés sur :

1. L'optimisation de l'erreur d'une seule représentation par la RL (*i.e.* optimiser $E[i]$ en fusionnant les segments de $R[i]$).
2. Ou bien sur l'importance accordée aux événements récents.

L'approximation obtenue par ces méthodes est telle que $\forall i, j \in [1..n]$, $S[i] = S[j]$ (la taille des représentations est identique d'une série à l'autre). À notre connaissance, il n'existe pas de travaux pour résumer un ensemble de séries temporelles en optimisant l'erreur globale (*c.à.d* optimisant $E(R)$ en fusionnant et en allouant les segments de R à chaque étape s). *Le premier problème traité dans ce chapitre porte sur la recherche d'une distribution optimale des M segments disponibles afin de minimiser l'erreur globale $E(R)$ dans un traitement en ligne.*

11.2.2 Fusion des segments

Comme nous l'expliquons en section 11.5, la RL appliquée sur deux segments peut-être calculée en O(1). En nous basant sur la RL, nous proposons une formule impliquant un nombre considérablement réduit de variables et d'opérateurs. *De plus, nous réduisons*

le nombre d'opérations nécessaires à la fusion de deux segments. Notre proposition pour une technique rapide d'approximation est expliquée en section 11.6.

11.3 Travaux existants

Résumer une série temporelle permet de réduire les dimensions des données d'origine. La qualité des résultats, obtenus suite à une requête ou une fouille appliquée aux résumés, dépend de la technique de réduction de dimensions utilisée. Ce sujet a été étudié dans de nombreux domaines comme la finance Zhu & Shasha (2002), l'extraction de motifs fréquents Giannella *et al.* (2003); Teng *et al.* (2003), la surveillance de réseaux Airoldi & Faloutsos (2004); Borgne *et al.* (2007) ou encore dans des applications comme la gestion des capteurs d'un réseau de distribution d'eau potable Papadimitriou *et al.* (2005). Les méthodes d'approximation proposées sont basées sur les ondelettes Chan & Fu (1999); Popivanov (2002), la transformée de Fourier Faloutsos *et al.* (1994); Rafiei & Mendelzon (1997), la RL Keogh & Pazzani (1998); Shatkay & Zdonik (1996), etc.

Dans Lin *et al.* (2003), les auteurs proposent une représentation symbolique des séries temporelles avec l'algorithme SAX. SAX permet de discrétiser les données d'origine à l'aide de chaînes de symboles. Les manipulations se faisant ensuite sur les symboles plutôt que sur les données. Lin *et al.* (2003) présente une fonction qui renvoie la distance minimum entre les données d'origine discrétisées sous la forme de deux représentations symboliques.

Dans Yi & Faloutsos (2000), les auteurs proposent *segmented-means*, une méthode d'extraction du vecteur de caractéristiques des série temporelles. *segmented-means* découpe chaque série en plusieurs segments de longueur égale. Ensuite, pour chaque segment, les caractéristiques sont obtenues en calculant la moyenne des valeurs de ce segment. Un avantage important de *segmented-means* est son adaptation à n'importe quelle norme $\mathcal{L}_p$.

Une approche incrémentale basée sur la transformée de Fourier est proposée dans Ogras & Ferhatosmanoglu (2006). Les auteurs proposent de trouver les coefficients qui minimiseraient l'erreur carrée entre la séquence d'origine et sa représentation, le tout dans un contexte dynamique. Tout d'abord, ils proposent de mettre à jour de manière incrémentale les M coefficients les plus significatifs d'un flux de données. Ensuite, ils étendent leur méthodologie vers un modèle qui correspond à une fenêtre glissante de

taille N. Ce modèle est calculé de manière récursive avec une complexité dans le temps de $O(M)$.

Chen *et al.* (2002) et Palpanas *et al.* (2008) proposent des techniques pour un calcul incrémental de la RL sur une série temporelle. Dans Chen *et al.* (2002) les auteurs proposent un principe intéressant de fusion des segments en temps linéaire. Dans la mesure où leur objectif est de résumer les séries en respectant des contraintes de l'utilisateur, ils ne donnent pas d'information sur le calcul de l'erreur. Dans Palpanas *et al.* (2008) les auteurs reprennent le principe de fusion des segments donné par Chen *et al.* (2002) et proposent un théorème qui montre que l'erreur entre l'approximation obtenue sur deux segments et les points d'origine peut être calculée comme la somme des erreurs entre (a) les deux segments et les points d'origine et (b) les deux segments et l'approximation sur ces deux segments. Ensuite, Palpanas *et al.* (2008) propose une étude sur l'intégration du facteur d'ancienneté dans le calcul de la RL. Chen *et al.* (2002) propose également l'utilisation d'un facteur d'ancienneté en stockant des résumés de séries temporelles dans des cubes. Le modèle d'ancienneté utilisé est celui des fenêtres logarithmiques (le facteur d'oubli des données augmente de manière logarithmique dans le temps). Teng *et al.* (2003) et Giannella *et al.* (2003) proposent de gérer l'historique des fréquents extraits à partir d'un flux en appliquant également un facteur d'ancienneté. Dans Giannella *et al.* (2003), le principe des fenêtres logarithmiques est appliqué afin d'accorder une plus grande importance à l'historique récent. Dans Teng *et al.* (2003), l'historique est représenté sous la forme d'une série temporelle et l'approximation est appliquée en priorité sur les segments anciens.

Citons également Zhu & Shasha (2002) qui propose de gérer plusieurs séries temporelles simultanément. Les auteurs proposent StatStream, une méthode permettant de gérer un ensemble de séries et de détecter les paires de séries ayant un coefficient de corrélation supérieur à un seuil donné. Les comparaisons sont faites sur les représentations condensées (basées sur les ondelettes) des données d'origine.

11.4 Contribution : Résumé à Erreur Globale Optimisée (REGLO)

La sous-section 11.4.1 présente le principe général de notre approche afin de résumer un flux de séries temporelles. Nous discutons des avantages de la stratégie proposée en

sous-section 11.4.2.

11.4.1 Principe général

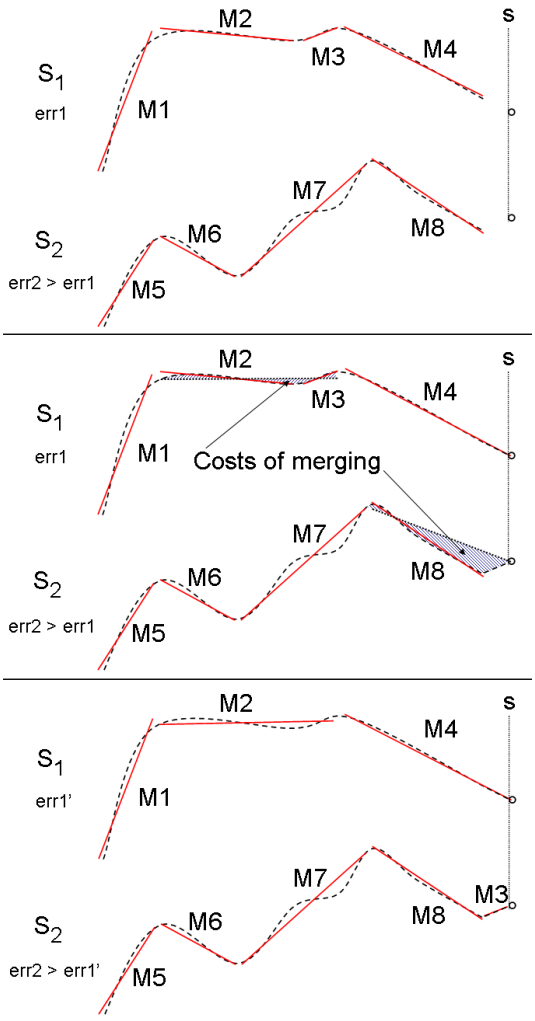


FIG. 11.1: Principe général. Ici, les représentations partagent 8 blocs en mémoire qui sont alloués en fonction de l'erreur sur chaque représentation. La représentation de S1 n'a besoin que de 3 blocs alors que S2 en nécessite 5.

Considérons $T[1..n]$, l'ensemble des séries donné en section 11.2. A chaque étape s (chaque série est mise à jour avec une nouvelle valeur ou avec la valeur zéro si la série n'est pas concernée) nous voulons mettre à jour les représentations et les erreurs

associées. Afin de maintenir un taux d'erreur global satisfaisant, nous devons choisir les représentations qui minimisent leur erreur locale. L'idée motivant ce choix est basée sur le fait qu'une représentation avec une erreur locale faible va mieux tolérer la fusion de deux segments (par rapport à une représentation avec une erreur déjà plus élevée). Quand $(s \times n) > M$ il faut libérer des blocs en mémoire afin de prendre en compte les nouvelles valeurs. Ainsi, à chaque étape s ; pour chaque nouvelle valeur :

1. Soit r la représentation ayant la plus faible erreur locale. Alors r est compressée (par la fusion de deux segments).
2. $E(r)$ l'erreur locale de r est mise à jour.
3. $E(R)$, l'erreur globale, est mise à jour.

Exemple 6 *La figure 11.1 donne une illustration de ce principe avec $n = 2$ (S_1 et S_2) et $M = 8$. Soit R_i la représentation de S_i. Deux nouvelles valeurs (à l'étape s de la figure 11.1) sont produites par le flux. R_1 et R_2 disposent chacune de 4 blocs (un segment étant représenté sur un bloc). Les trois étapes suivantes sont alors décrites par la figure 11.1 :*

1. *Suite à la production de ces deux valeurs il faut libérer deux segments. Étendre le segment $M4$ avec la nouvelle valeur de $S1$ n'a aucun coût en terme d'erreur. $M4$ est donc étendu avec la nouvelle valeur.*
2. *Le coût d'extension de $M8$ avec la nouvelle valeur de S_2 n'est pas négligeable. L'erreur résiduelle serait plus grande que celle résultant de la fusion de $M2$ et $M3$. Les deux erreurs sont décrites dans la figure 11.1. Ainsi, il est préférable (en terme d'erreur) de fusionner $M2$ et $M3$ et de réallouer le segment $M3$ à la représentation R_2.*
3. *$M2$ et $M3$ sont fusionnés en $M2$. $err1$ est mis à jour en $err1'$. $M3$ (maintenant libre) est alloué à R_2 et il est utilisé pour stocker la nouvelle valeur de S_2.*

Après ces étapes, le processus peut continuer à mettre à jour le modèle en fonction des nouvelles valeurs du flux.

11.4.2 Motivation et algorithme général

Les représentations basées sur un facteur d'ancienneté ont des avantages qui dépendent du contexte applicatif. Un de ces avantages est de donner plus d'importance aux événements récents. Toutefois, ce principe peut avoir des limites comme par exemple

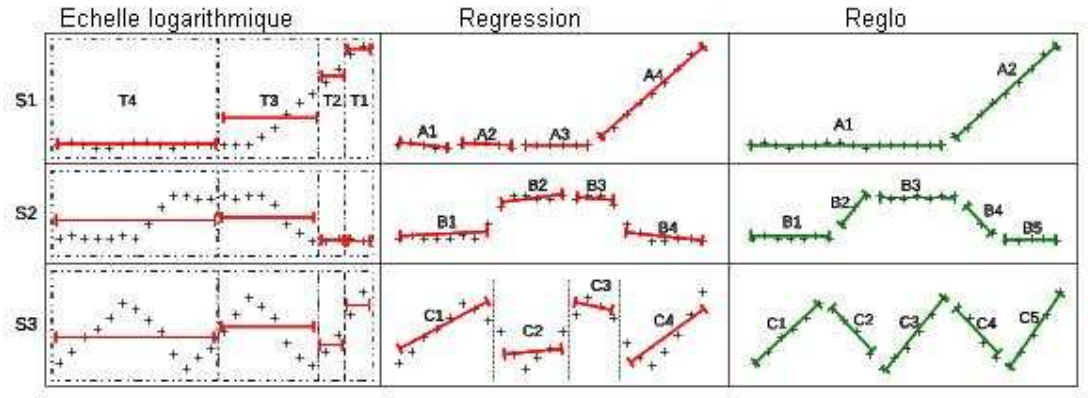


FIG. 11.2: Un ensemble de séries temporelles et leur compression selon différentes techniques dans un espace mémoire de 12 blocs. Le graphique de gauche utilise une échelle logarithmique et un nombre constant de segments par série. Le graphique de milieu utilise la RL avec un nombre constant de segments par série. Le graphique de droite utilise une approche optimisant l'erreur globale (l'algorithme REGLO).

l'indifférence face aux changements importants dans les valeurs du flux. En effet, le facteur d'ancienneté est le seul critère utilisé lors de la compression rendant ce principe inadéquat pour certains types d'applications. La détection de fraudes, par exemple, peut nécessiter de trouver des anomalies dans l'historique du flux. Une anomalie est un événement souvent atypique (indépendamment de son ancienneté). Nous pensons que les requêtes sur l'historique d'un flux (stocké sous forme de représentation condensée) doivent se faire sur un modèle qui distribue la précision de la représentation en fonction de critères autres que l'ancienneté. Le besoin de précision (basée sur l'erreur résiduelle) est le critère privilégié dans notre méthode.

La figure 11.2 illustre les avantages d'une régression basée sur l'erreur globale par rapport à un modèle utilisant un nombre égal de segments pour chaque représentation. Chaque modèle dispose de 12 segments en mémoire à distribuer entre les représentations. La RL ou les fenêtres logarithmiques n'essaient pas d'équilibrer cette distribution et allouent 4 segments à chacune des trois représentations. La régression optimisant l'erreur globale distribue les segments de manière adéquate. Dans cette illustration, la représentation donnée par REGLO pour S_1 dispose de 2 segments alors que S_2 et S_3 disposent de 5 segments chacune. Ainsi, avec le même nombre de segments pour chaque

modèle (RL, fenêtre logarithmiques et REGLO), l'erreur globale est significativement différente. La figure 11.3 donne l'algorithme générique de REGLO. Nous considérons que le tri des couples de segments par ordre de coût de fusion peut être géré par une pile (C). La complexité en espace de REGLO est de O(M).

Algorithme REGLO
Entrée : T, un flux composé de n séries temporelles et M le nombre de segments disponibles pour le modèle.
Sortie : à chaque étape s, $R[1..n]$, les représentations de T avec une erreur globale optimisée. $R[i]$ est la représentation de la série i. $R[i][j]$ est la j^{eme} valeur de la i^{eme} série.

1. Tant que $(s \times n) < M$, affecter un segment à R pour chaque nouvelle valeur de T.
2. Construire C, la pile gérant les coûts de fusion.
   //Chaque série temporelle a une pile P qui retient les valeurs des erreurs locales
   //triés. Les premières valeurs de ces piles sont retenues dans une pile globale C
   //(n valeurs au total).Le coût de fusion est égal à la première valeur de cette pile
   //générale.
3. Quand un nouvel ensemble E de n valeurs arrive (à chaque étape s) :
   For i = 1 to n :
   (a) Ajouter à chaque série i sa valeur correspondante de l'ensemble E, E[i] ;
   (b) Calculer l'erreur locale correspondante à l'unification entre le dernier élément et le nouvel élément ajouté ;
   (c) Mettre à jour la pile des erreurs locales P
   //Cette pile retient toutes les erreurs d'unification de deux segments
   //adjacents. On a une telle pile par série temporelle.

   On doit libérer n espaces de mémoire correspondantes à n unifications à faire :
   For i = 1 to n :
   (a) Prendre la première valeur $V(Segm_1, Segm_2)$ de la pile C ;
   (b) Faire l'unification du $Segm_1$ et $Segm_2$;
   (c) Mettre à jour la pile locale de la série correspondante ;
   (d) Mettre à jour la pile globale C.

FIG. 11.3: Algorithme générique de REGLO.

En effet, ce principe repose sur une mesure fondamentale et complexe : calculer l'erreur résiduelle de la fusion de deux segments. *Dans un environnement dynamique tel qu'un flux de données, toute optimisation de cette évaluation est nécessaire.* La première optimisation est de rendre récursif le calcul de l'erreur résiduelle c'est-à-dire que l'erreur résiduelle de la fusion de deux segments se calcule en fonction de l'erreur résiduelle de chaque segment et du coût de fusion de ces deux segments. Dans ce chapitre nous comparons deux techniques possibles pour évaluer le coût de fusion d'un couple de segments :

1. La régression linéaire.
2. Notre nouvelle technique d'approximation, basée sur les milieux des segments à fusionner.

La section 11.5 donne les formules simplifiées du calcul de la RL sur deux segments en O(1) et du calcul de la nouvelle erreur en O(1). La section 11.6 donne les détails de notre approche pour la seconde technique d'approximation (basée sur les milieux) et explique comment évaluer l'erreur résiduelle avec cette technique. Chacune des méthodes présentées en sections 11.5 et 11.6 peut donc être utilisée pour calculer la fusion de deux segments et l'erreur résiduelle.

11.5 Contribution : nouvelles équations pour la RL

Comme mentionné en section 11.3, Chen *et al.* (2002) donne un ensemble de formules permettant d'obtenir la RL sur deux segments en O(1) et Palpanas *et al.* (2008) donne la formule permettant de mettre à jour en O(1) l'erreur suite à cette fusion. Dans ce chapitre, nous donnons des versions simplifiées de ces formules avec un nombre de variables et d'opérateurs significativement réduit. Nos formules permettront également une comparaison plus facile avec notre approche (présentée en section 11.6).

Tout d'abord, nous proposons une écriture simplifiée des formules permettant de fusionner deux segments en fonction de la pente et de la moyenne des valeurs des séries.

Nous considérons le cas du modèle de régression linéaire $Y = \alpha.X + \beta$, où α est la pente ou le coefficient directeur de la droite régression et β est l'ordonnée à l'origine. Afin d'estimer α et β, nous avons $((i, j_i), i = 1, \ldots, s)$ une série temporelle sur un ensemble $S = \{i | i = 1, \ldots, s\}$. Soit $\bar{j}$ la moyenne des valeurs sur $(j_1, \ldots, j_s)$, alors

$\bar{j} = \sum_{i=1}^{s} j_i/n$. L'idée de l'estimation des moindres carrés est de minimiser la somme des erreurs carrées :

$$E(S) = \sum_{i=1}^{s} \varepsilon_i^2 = \sum_{i=1}^{s} (j_i - \alpha.i - \beta)^2$$

Un ajustement linéaire pour une série temporelle j_i avec $i \in [1, s]$ est l'équation de régression linéaire $Y = \alpha.X + \beta$. Les paramètres α et β sont choisis pour minimiser $E(S)$ qui est la somme des carrés des erreurs résiduelles. Ces paramètres sont obtenus de la manière suivante :

$$\alpha = \frac{\sum_{i=1}^{s}(i - \bar{i})(j_i - \bar{j})}{\sum_{i=1}^{s}(i - \bar{i})^2}$$

et

$$\beta = \bar{j} - \alpha.\bar{i}$$

avec $\sum_{i=1}^{s}(i - \bar{i})^2 = (s-1)s(s+1)/12$ et $\bar{i} = (s+1)/2$.

La connaissance de α et de $\bar{j}$ permet de calculer les paramètres α et β de la RL. Afin d'accélérer les calculs, la mise à jour se fera uniquement sur les paramètres α et $\bar{j}$.

11.5.1 Fusion de deux segments

Considérons deux ensembles $S_1 = \{i | i = 1, \ldots, k\}$ et $S_2 = \{i | i = k+1, \ldots, s\}$ à fusionner. (S_1,S_2) est une bipartition de S. $Y = \alpha_1.X + \beta_1$ est l'équation de la régression linéaire de la série temporelle $((i, j_i), i = 1, \ldots, k)$.

Théorème 1 *Les paramètres α et $\bar{j}$ de S peuvent être dérivés des paramètres des ensembles S_1 et S_2 comme suit :*

$$\begin{aligned} S &= S_1 \cup S_2 \\ \alpha &= \frac{k^3 - k}{s^3 - s}.\alpha_1 \\ &+ \frac{(s-k)^3 - (s-k)}{s^3 - s}.\alpha_2 \\ &- \frac{6k(s-k)}{s^3 - s}.(\bar{j}_1 - \bar{j}_2) \\ \bar{j} &= \frac{k.\bar{j}_1 + (s-k)\bar{j}_2}{s} \end{aligned} \tag{11.1}$$

Remarque : l'ordonnée β peut être calculée par la formule suivante :

$$\beta = \frac{k.\bar{j}_1 + (s-k)\bar{j}_2}{s} - \frac{s+1}{2}.\alpha$$

.

Preuve

Le numérateur et la pente α peuvent être écrits comme suit :

$$\sum_{i=1}^{k}(i-\bar{i}_1+\bar{i}_1-\bar{i})(j_i-\bar{j}_1+\bar{j}_1-\bar{j})$$
$$+\sum_{i=k+1}^{s}(i-\bar{i}_2+\bar{i}_2-\bar{i})(j_i-\bar{j}_2+\bar{j}_2-\bar{j})$$

Le premier terme peut être décomposé en quatre termes :

$$\sum_{i=1}^{k}(i-\bar{i}_1)(j_i-\bar{j}_1)+k(\bar{i}_1-\bar{i})(\bar{j}_1-\bar{j})$$
$$+(\bar{j}_1-\bar{j})\sum_{i=1}^{k}(i-\bar{i}_1)+(\bar{i}_1-\bar{i})\sum_{i=1}^{k}(j_i-\bar{j}_1)$$
$$\text{on a}\sum_{i=1}^{k}(i-\bar{i}_1)=0 \text{ et } \sum_{i=1}^{k}(j_i-\bar{j}_1)=0$$

De plus, on a

$$\begin{aligned}
\bar{i}_1-\bar{i} &= \frac{s.\bar{i}_1-k.\bar{i}_1-(s-k)\bar{i}_2}{s} \\
&= \frac{(s-k)}{s}.(\bar{i}_1-\bar{i}_2) \\
\text{et} \quad \bar{j}_1-\bar{j} &= \frac{(s-k)}{s}.(\bar{j}_1-\bar{j}_2) \\
\text{comme} \quad (\bar{i}_1-\bar{i}_2) &= -s/2 \\
\text{et} \quad (\bar{i}_1-\bar{i})(\bar{j}_1-\bar{j}) &= \frac{-(s-k)^2}{2s}.(\bar{j}_1-\bar{j}_2)
\end{aligned}$$

Pour le numérateur et la pente on obtient :

$$(k^3-k).\alpha_1+((s-k)^3-(s-k)).\alpha_2$$
$$-\frac{k(s-k)^2}{2s}.(\bar{j}_1-\bar{j}_2)-\frac{k^2(s-k)}{2s}.(\bar{j}_1-\bar{j}_2)$$

Comme le dénominateur est égal à $(s-1)s(s+1)/12$ (ou $(s^3-s)/12$), le théorème est prouvé.

Concernant le calcul de β, on part de la formule :

$$y_i = \alpha \cdot x_i + \beta, i \in [1, s]$$

On additionne tous les termes :

$$\sum_{i=1}^{s} y_i = \alpha \cdot \sum_{i=1}^{s} x_i + s.\beta, i \in [1, s]$$

et on divise l'équation par s.

Comme :

$$\frac{\sum_{i=1}^{s} y_i}{s} = \frac{\sum_{i=1}^{k} y_i + \sum_{i=k+1}^{s} y_i}{s} = \frac{k.\overline{j_1} + (s-k).\overline{j_2}}{k}.$$

et

$$\frac{\sum_{i=1}^{s} x_i}{s} = \frac{\frac{s(s+1)}{2}}{s} = \frac{s+1}{2}$$

On obtient :

$$\beta = \frac{k.\bar{j}_1 + (s-k)\bar{j}_2}{s} - \frac{s+1}{2}.\alpha$$

Ce théorème permet de construire une régression linéaire sur S avec les paramètres α_1, α_2, $\bar{j}_1$ et $\bar{j}_2$, à partir des régressions de S_1 et S_2 sans les points d'origine de la série j.

11.5.2 Mise à jour de l'erreur

Dans Palpanas *et al.* (2008), le théorème 2 est important car il permet de calculer la somme des erreurs carrées sur S avec les erreurs carrées sur S_1 et S_2 et l'erreur entre les régressions locales sur S_1 et S_2 et la régression globale sur S. Ce qui s'exprime comme :

$$E(S) = E(S_1) + E(S_2) + \hat{E}(S_1 \cup S_2) \tag{11.2}$$

Selon le lemme 1 de Palpanas *et al.* (2008), l'erreur $\hat{E}(S_1 \cup S_2)$ qui est le coût de cette fusion, peut être calculée en $O(1)$. Nous proposons une autre preuve de ce lemme avec une formule proche mais condensée qui dépend des paramètres des régressions linéaires de S_1 et S_2.

$$\begin{aligned}\hat{E}(S_1 \cup S_2) &= \sum_{i=1}^{k}(\alpha_1.i + \beta_1 - \alpha.i - \beta)^2 \\ &+ \sum_{i=k+1}^{s}(\alpha_2.i + \beta_2 - \alpha.i - \beta)^2\end{aligned}$$

Ainsi, on obtient les formules suivantes :

$$\begin{aligned}\hat{E}(S_1 \cup S_2) &= k(\beta_1 - \beta)^2 + k(k+1)(\alpha_1 - \alpha)(\beta_1 - \beta) \\ &+ \frac{k(k+1)(2k+1)}{6}(\alpha_1 - \alpha)^2 \\ &+ (s-k)(\beta_2 - \beta)^2 \\ &+ (s-k)(s-k+1)(\alpha_2 - \alpha)(\beta_2 - \beta) \\ &+ \frac{(s-k)(s-k+1)(2(s-k)+1)}{6}(\alpha_2 - \alpha)^2\end{aligned}$$

La mise à jour de l'erreur $E(S)$ permet de calculer le coût de la fusion de S_1 avec S_2. Ce coût est égal à $C(S_1 \cup S_2) = E(S) - E(S_1) - E(S_2) = \hat{E}(S_1 \cup S_2)$ et est utilisé dans l'étape 2 de l'algorithme REGLO.

11.6 Contribution : une approche rapide pour la fusion des segments

Bien que les résultats précédents permettent de fusionner et d'évaluer l'erreur et le coût de fusion en O(1), nous voulons optimiser encore cette étape. En effet, ces calculs sont au coeur de notre algorithme d'approximation destiné à résumer un flux de données. Toute optimisation de ces calculs sera donc nécessaire. Si l'opération de fusion est en temps linéaire, de meilleurs temps d'exécution peuvent être obtenus en diminuant le nombre d'opérations, ce qui peut être crucial dans le contexte des flux de données. Dans cette section, nous proposons de calculer une approximation de la RL grâce à une idée innovante basée sur l'observation suivante : quand le coût de fusion de deux segments est faible alors les points de croisements entre la droite de régression et les segments sont proches des milieux de ces segments.

11.6.1 Fusion de deux segments

La RL a pour résultat une représentation d'une série temporelle telle que chaque segment minimise la somme des erreurs quadratiques entre la représentation et les données d'origine. Notre approche consiste à élaborer une approximation intuitive et efficace de ces valeurs d'origine. AMi, notre technique d'approximation, sera représentée par la ligne qui coupe les segments d'origine en leurs milieux. Cette technique présente deux avantages principaux :

1. Elle ne demande pas de nombreux calculs impliquant autant de variables que la RL comme exprimée dans le théorème 1, ce qui la rend plus rapide que cette dernière.
2. Son interprétation est naturelle. Cette technique ne minimise pas la somme du carré des erreurs, mais nous montrons que l'approximation obtenue est très fidèle aux données d'origine. En effet, la somme du carré des erreurs pénalise les représentations composées de nombreuses valeurs aberrantes. L'erreur sur ces valeurs étant rendues très importante dans le calcul global en raison de la puissance de deux utilisée. Privilégier les valeurs aberrantes n'est pas forcément le meilleur moyen d'avoir une représentation fidèle des données d'origine. Comme nous le montrerons, les représentations obtenues par AMi et la RL sont comparables. Toutefois, nous voulons être moins sensibles aux valeurs aberrantes. Ainsi, la représentation obtenue par AMi est un bon compromis entre les normes $\mathcal{L}_1$ et $\mathcal{L}_2$. En effet, nos expérimentations montrent que AMi peut avoir de meilleurs résultats quand l'erreur est mesurée dans la norme $\mathcal{L}_1$.

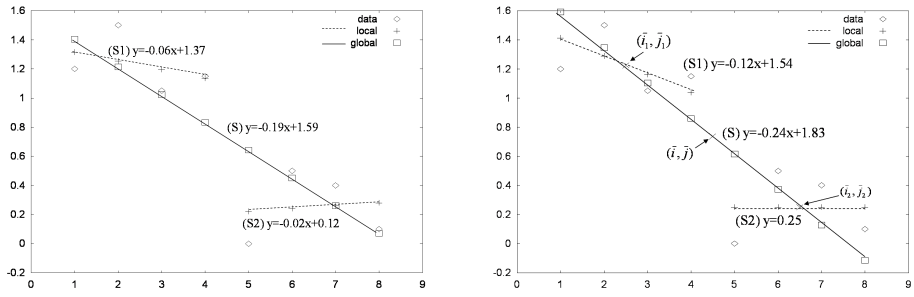


FIG. 11.4: Illustration de la RL (minimisant l'erreur quadratique) en comparaison d'AMi (ligne qui passe par les milieux des segments).

Exemple 7 *La figure 11.4 illustre la différence entre la RL (gauche) et AMi (droite). Avec AMi, un premier ensemble de 4 segments est construit entre les points 1 et 2, les points 3 et 4, les points 5 et 6 et les points 7 et 8. Ensuite, le deuxième niveau d'approximation fusionne les deux premiers segments en passant par les milieux des segments et on obtient le segment $S1$ de la figure 11.4. Le même principe est appliqué aux deux segments restants (points 5 à 8) pour donner le segment $S2$. Le troisième niveau d'approximation fusionnera les deux segments $S1$ et $S2$ avec pour résultat S, le segment qui passe par leurs milieux (i.e. $(\bar{i}_1, \bar{j}_1)$ et $(\bar{i}_2, \bar{j}_2)$).*

Soit $(\bar{i}_1, \bar{j}_1)$ et $(\bar{i}_2, \bar{j}_2)$ les milieux des segments $S1$ et $S2$ avec $\bar{i}_1 = (k+1)/2$ et $\bar{i}_2 = (s+k+1)/2$. L'équation du segment S est obtenue par résolution de l'ensemble d'équations linéaires suivant :

$$\bar{j}_1 = \alpha.\bar{i}_1 + \beta \text{ and } \bar{j}_2 = \alpha.\bar{i}_2 + \beta$$

La pente α et l'ordonnée β sont données par :

$$\alpha = \frac{\bar{j}_1 - \bar{j}_2}{\bar{i}_1 - \bar{i}_2}$$
$$\beta = \frac{\bar{j}_2.\bar{i}_1 - \bar{j}_1.\bar{i}_2}{\bar{i}_1 - \bar{i}_2}$$

et le nouveau point milieu $(\bar{i}, \bar{j})$ de S est donné par :

$$\begin{aligned} \bar{i} &= (s+1)/2 \\ \bar{j} &= \alpha.\bar{i} + \beta \end{aligned}$$

Le calcul de $\bar{j}$ est donné par :

$$\bar{j} = \frac{k.\bar{j}_1 + (s-k)\bar{j}_2}{s}$$

Dans le cas où l'hypothèse de la distribution uniforme sur S est vérifiée, le point milieu de S est le point moyen et le point médian. Cette hypothèse est vérifiée car nos observations arrivent par intervalles réguliers. Le point milieu de S vérifie les équations suivantes :

$$\bar{i} = \frac{1}{s}.\sum_{i=1}^{s} i \text{ and } \bar{j} = \frac{1}{s}.\sum_{i=1}^{s} j_i$$

De ce fait, ce point appartient à la droite de régression linéaire. Le théorème 2 montre la relation entre les pentes de la RL et de AMi, qui peut être calculée par :

$$\begin{aligned} \alpha^{RL} &= \frac{k^3-k}{s^3-s}.\alpha_1^{RL} \\ &+ \frac{(s-k)^3-(s-k)}{s^3-s}.\alpha_2^{RL} \\ &+ \frac{3k(s-k)}{s^2-1}.\alpha^{AMi} \end{aligned}$$

11.6.2 Mise à jour de l'erreur

Théorème 2 *L'erreur commise par AMi sur le segment $S = S_1 \cup S_2$ peut être calculée de manière récursive à partir des erreurs pondérées des segments S_1 et S_2 et de la pente α du segment S.*

Preuve

La somme des erreurs carrées de S peut être décomposée en deux parties :

$$\begin{aligned} E(S) &= \sum_{i=1}^{k}(j_i - \alpha.i - \beta)^2 \\ &+ \sum_{i=k+1}^{s}(j_i - \alpha.i - \beta)^2 \end{aligned}$$

La première partie de cette décomposition peut être réécrite de la manière suivante :

$$\begin{aligned} &\sum_{i=1}^{k}(j_i - \alpha.i - \beta - (\alpha_1.i + \beta_1) + (\alpha_1.i + \beta_1))^2 \\ &= \sum_{i=1}^{k}(j_i - \alpha_1.i - \beta_1)^2 + (\alpha_1.i + \beta_1 - \alpha.i - \beta)^2 \\ &\quad +2(j_i - \alpha_1.i - \beta_1)\big((\alpha_1 - \alpha).i + (\beta_1 - \beta)\big) \end{aligned}$$

La première somme des erreurs quadratiques dans la première ligne de cette réécriture représente la somme des erreurs quadratiques de S_1 obtenue avec AMi. La seconde partie représente les erreurs entre le segment S_1 obtenu par AMi et le segment S obtenu par AMi calculé sur S_1. Comme le modèle obtenu par la RL, AMi vérifie que $\sum_{i=1}^{k}(j_i - \alpha_1.i - \beta_1) = 0$ et la dernière partie peut être simplifiée par :

$$\begin{aligned}\Delta(S_1/\alpha) &= 2(\alpha_1 - \alpha)\sum_{i=1}^{k}(j_i - \alpha_1.i - \beta_1)i \\ &= 2(\alpha_1 - \alpha)\sum_{i=1}^{k} i.j_i \\ &- k(\alpha_1 - \alpha)\big(\alpha_1(2k+1)(k+1)/3 - \beta_1(k+1)\big)\end{aligned}$$

Avec AMi, la somme des erreurs quadratiques de $E(S)$ est plus complexe que la somme des erreurs quadratiques de la RL définie par l'équation 11.2, mais la complexité par rapport à s est inchangée. $E(S)$ peut être exprimé de la manière suivante :

$$\begin{aligned}E(S) &= E(S_1) + E(S_2) + \hat{E}(S_1 \cup S_2) \\ &+ \Delta(S_1/\alpha) + \Delta(S_2/\alpha)\end{aligned} \tag{11.3}$$

Le calcul de $\Delta(S_1/\alpha)$ ou $\Delta(S_2/\alpha)$ dépend de la pente du nouveau segment S et des valeurs $\delta(S_1) = \sum_{i=1}^{k} i.j_i$ et $\delta(S_2) = \sum_{i=k+1}^{s} i.j_i$. Ainsi, la relation linéaire $\delta(S) = \delta(S_1) + \delta(S_2)$ existe et δ est maintenu pour chaque segment. De plus, $\Delta(S_1/\alpha)$ peut se calculer facilement :

$$\Delta(S_1/\alpha) = -\alpha.\delta(S_1) + \Lambda(S_1)$$

Où $\Lambda(S_1)$ dépend seulement du segment S_1 et on obtient :

$$\begin{aligned}E(S) &= (E(S_1) + \Lambda(S_1)) + (E(S_2) + \Lambda(S_2)) \\ &+ \hat{E}(S_1 \cup S_2) - \alpha(\delta(S_1) + \delta(S_2))\end{aligned}$$

11.7 Expérimentations

Dans cette section, nous évaluons l'efficacité et la qualité de notre méthode d'approximation grâce à un ensemble d'expérimentations. Nous avons implémenté deux

versions de REGLO. Dans la première version, l'approximation et le calcul de l'erreur sont basés sur la RL (telles que présentées en section 11.5). Dans la deuxième version, l'approximation et le calcul de l'erreur sont basés sur AMi (C.f. section 11.6). Nous avons également implémenté une compression basée sur les ondelettes et une autre basée sur les fenêtres logarithmiques afin de comparer REGLO avec les techniques les plus importantes. Une description des ondelettes peut être trouvée en Daubechies (1992). Les fenêtres logarithmiques sont décrites en Chen *et al.* (2002); Giannella *et al.* (2003). Nous avons évalué nos algorithmes sur deux jeux de données réelles. Le premier jeu de données, "NYSE", est construit sur les données téléchargées depuis `http://icf.som.yale.edu/nyse`. Ce jeu contient 134 séries temporelles, chacune contenant 804 valeurs correspondant aux valeurs du marché de New-York de 1815 à 1925. Le deuxième jeu de données, "WEB", vient des logs d'accès Web anonymisés de l'Inria Sophia-Antipolis. Nous avons recueilli un an d'usage, pour un total de 14 Go. Un premier prétraitement a permis d'extraire les URLs les plus fréquentes de ce jeu. Avec un support minimum de 1%, nous en avons trouvé 223. Ensuite, nous avons reporté les variations du nombre de requêtes à ces URLs avec une fenêtre sautante de taille 1000. En d'autres termes, chaque série est mise à jour toutes les 1000 requêtes et la valeur affectée à la série correspond au nombre de requêtes sur l'URL pendant cet intervalle.

REGLO (versions RL et AMi) est implémenté en Java. Les expérimentations sont effectuées sur un PC équipé de 2 Gb de mémoire et un pentium à 4 processeurs cadencés à 2,2 Ghz.

11.7.1 Comparaison entre AMi et la RL

Dans ce premier ensemble d'expérimentations nous évaluons, sur les deux jeux de données :

- L'impact de AMi sur l'erreur moyenne à chaque étape.
- La différence des temps d'exécutions entre AMi et la RL.

La figure 11.5 montre l'erreur moyenne des approximations obtenues par AMi et la RL pour les données NYSE (haut) et WEB (bas) sur les 251 premières valeurs. L'erreur est calculée de la manière suivante : $\forall i \in [1..s] err[i] = averageError_{j=1}^{j=n}(R[j][i])$ où $R[j][i]$ est l'erreur entre la représentation de l'enregistrement i de la série j et la valeur réelle de i dans les données d'origine. Ainsi, pour chaque unité de temps, nous connaissons l'erreur moyenne du modèle pour les n séries temporelles. Pour NYSE les erreurs

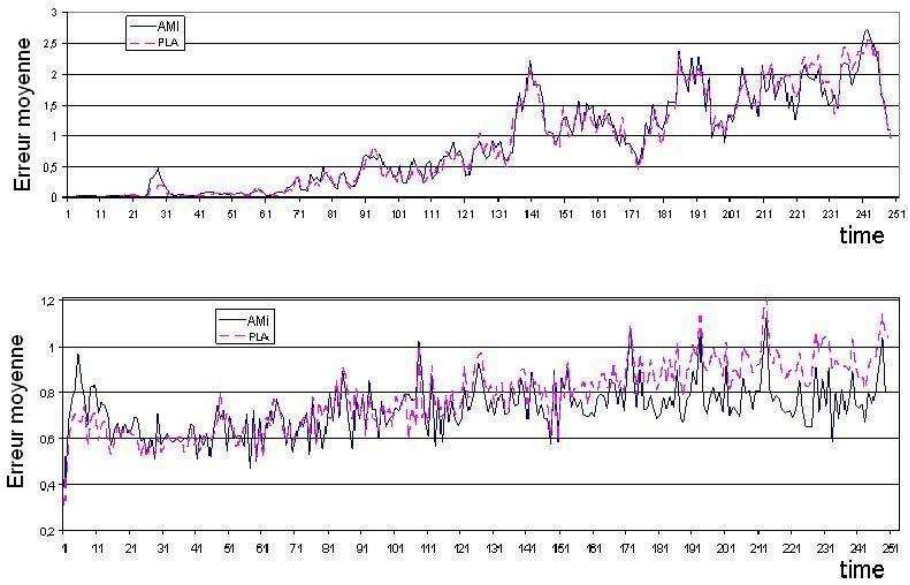


FIG. 11.5: Erreur moyenne étape par étape pour AMi et la RL sur NYSE et WEB.

moyennes commises par AMi et la RL sont très proches. La qualité de l'approximation obtenue par AMi est illustrée sur le jeu de données WEB. Pour ce jeu, AMi obtient une très bonne erreur moyenne en comparaison de la RL. La sous-section 11.7.2 donne les erreurs globales moyennes des deux méthodes d'approximation et AMi obtient de meilleurs résultats (0,87) que la RL (2,52). La figure 11.6 montre les temps d'exécution de REGLO sur les deux jeux de données, selon que l'on utilise l'approximation de AMi ou de la RL. Les temps de réponse sont systématiquement plus faible pour AMi. L'explication vient du nombre considérablement réduit d'opérations (Cf. les formules de la section 11.6) nécessaires à AMi pour calculer l'approximation et l'erreur résiduelle en comparaison des formules complexes (que nous avons pourtant simplifiées autant que nous le pouvons) de la RL (Cf. section 11.5).

11.7.2 Ondelettes et fenêtres logarithmiques

L'objectif de cette section est de comparer l'erreur commise par REGLO avec celles des ondelettes et des fenêtres logarithmiques. La figure 11.7 montre l'erreur moyenne à chaque étape (principe similaire à la figure 11.5) avec une représentation basée sur les ondelettes de Haar sur les 250 premières valeurs. Dans ces expérimentations, les modèles

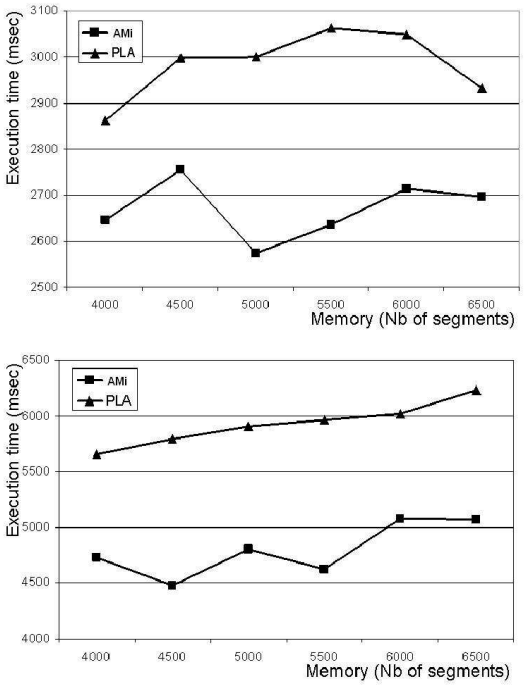


FIG. 11.6: Temps d'exécution de AMi et de la RL sur NYSE et WEB en fonction de la mémoire disponible.

obtenus par AMi et la RL disposent chacun de 5000 segments. Les modèles basés sur les fenêtres logarithmiques demandent 11 segments par série (11 étant obtenu par le logarithme de s, le nombre maximum de valeurs dans les séries du jeu de données). Le modèle basé sur les ondelettes dispose d'un nombre de segments correspondant à celui de REGLO (*c.à.d.* M/n coefficients pour chaque série, avec M le nombre de segments disponibles dans REGLO et n le nombre de séries à modéliser). Dans le cas de la figure 11.7, chaque série dispose de 34 segments. Ce nombre correspond à la mémoire disponible à la figure 11.5. En d'autres termes, pour chaque représentation par ondelettes on conserve les 34 coefficients les plus significatifs.

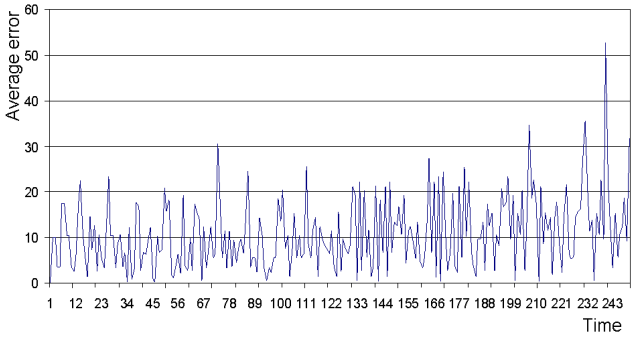


FIG. 11.7: Erreur moyenne, étape par étape, des ondelettes sur le WEB.

Concluons cette section en observant l'erreur globale commise par chaque représentation. Le tableau 11.1 donne l'erreur globale moyenne de chaque modèle en comparaison aux données d'origine. Pour chaque modèle, l'erreur moyenne est calculée à chaque étape (comme décrit à la section 11.7.1). Ensuite, la valeur moyenne des erreurs de chaque modèle est reportée dans la table 11.1. On voit que la valeur moyenne des erreurs obtenue avec AMi est beaucoup plus petite que celles obtenue avec les ondelettes ou les fenêtres.

11.8 Discussion

Nous avons proposé une nouvelle stratégie de résumé d'un flux de séries temporelles - REGLO et un nouveau principe rapide d'approximation des valeurs d'une série - AMi.

TAB. 11.1: Erreurs globales des ondelettes, des fenêtres logarithmiques, de la RL et de AMi

	WEB	NYSE
Fenêtres	17.245627	2.451278
Ondelettes	14.0588	1.55812
RL	2.529895911	0.857884432
AMi	0.879335223	0.868300858

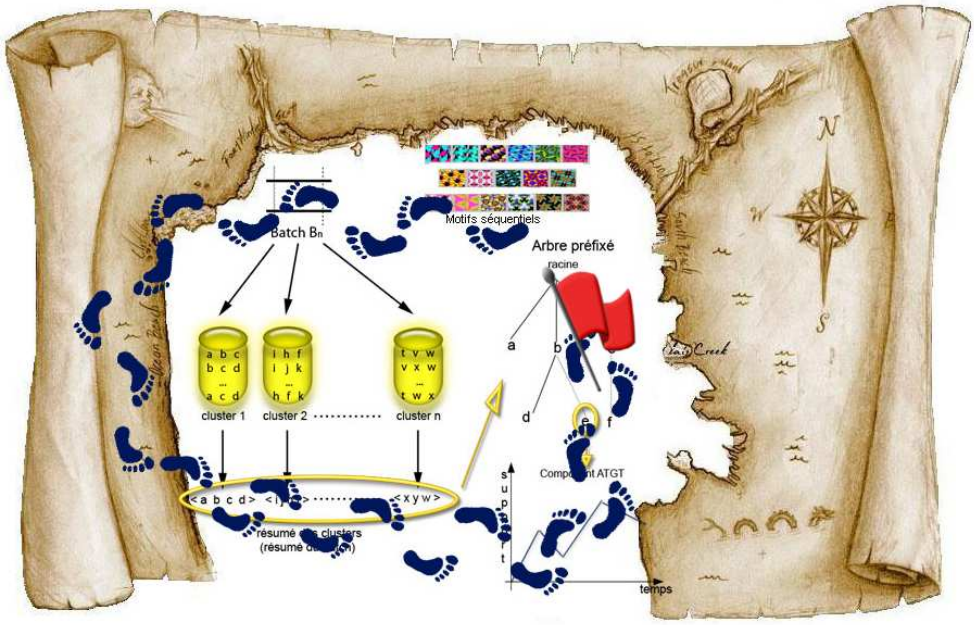
REGLO est capable de gérer de nombreuses séries temporelles sans limite de taille grâce à son principe d'équilibre et de distribution de la mémoire entre les représentations. Ce principe permet à chaque série d'obtenir ou de restituer de l'espace mémoire selon ses besoins de précision. Il n'est pas toujours facile de spécifier la bonne quantité de mémoire demandée par une application ; avec notre méthode, ce problème est résolu automatiquement car notre méthode s'adapte à une quantité de mémoire spécifiée par l'utilisateur. Le principe d'approximation rapide est comparé à la régression linéaire dans ses aspects formels, mais aussi grâce à une batterie d'expérimentations. Les résultats ont montré l'efficacité du principe d'approximation. Notre étude sur les relations entre AMi et la RL montre que AMi permet un calcul récursif de l'erreur entre la représentation et les données d'origine. Nous avons évalué nos algorithmes sur des données réelles afin de montrer les performances de notre technique d'approximation et de la distribution de l'espace mémoire. Ces expérimentations montrent que nos objectifs (diminuer l'erreur globale et accélérer les temps de calcul des résumés) sont atteints. Parmi les pistes ouvertes par ce travail citons l'étude des caractéristiques des données sur lesquelles l'approximation obtenue par AMi est meilleure que celle de la RL.

Les travaux développés lors de ce chapitre ont donné lieu aux publications suivantes :

- MARASCU, A. & MASSEGLIA, F. & LECHEVALLIER, Y. (2010). A Fast Approximation Strategy for Summarizing a Set of Streaming Time Series. In *25th Annual ACM Symposium on Applied Computing.* Sierre, Suisse. A paraître.
- MARASCU, A. & MASSEGLIA, F. & LECHEVALLIER, Y. (2010). REGLO : une nouvelle stratégie pour résumer un flux de séries temporelles. In *Extraction et gestion de connaissances,* Hammamet, Tunisie. A paraître.

Cette nouvelle stratégie de gestion de l'historique des motifs séquentiels sera utilisée par un composant de la structure de stockage que nous avons mise en place. Le chapitre suivant est dédié à la présentation de la structure de stockage que nous proposons. .

CARTE GENERALE



Dans le chapitre suivant, nous proposons une structure de stockage des résumés pendant l'exécution. Cette structure est composée par un arbre préfixé dont chacun des nœuds est lié à un composant capable de gérer l'historique de la fréquence du motif séquentiel correspondant au chemin qui va de la racine jusqu'à ce noeud.

CHAPITRE 12

STRUCTURE DE STOCKAGE

12.1 Introduction

VU LA TAILLE infinie d'un flux, le souhait de garder les données en intégralité sur les machines n'est qu'un rêve bien irréalisable. En échange, on peut aspirer à une bonne approximation de la totalité des données passées. Dans ce but, on a besoin en premier d'une structure se stockage efficace ; l'efficacité se mesurant en termes de mémoire et de temps d'accès.

12.2 Contribution : Arbre préfixé aux Tableaux de Granularités Temporelles Adaptatives (ATGT)

Lors de cette thèse, nous proposons une nouvelle structure de stockage composée par un arbre préfixé et par des tableaux de granularités temporelles adaptatives (Adaptative Time Granularity Table-ATGT) associées à chaque nœud de l'arbre. Chacun de ces composants est présentés en détail dans les sections suivantes. Cette structure est utilisée pour sauvegarder les résumés pendant l'exécution du flux. A la fin du traite-

ment de chaque batch on met à jour la structure de stockage en ajoutant les résumés correspondant au batch courant s'ils ne sont pas contenus, sinon on ajoute le support au composant ATGT du noeud correspondant et, selon le cas, on met à jour l'ATGT. Ensuite on applique une étape d'élagage où les noeuds avec des ATGTs vides sont supprimés.

12.2.1 Arbre préfixé

Les éléments constitutifs des motifs séquentiels approximatifs fréquents sont mis dans une structure d'arbre préfixé (Masseglia *et al.* (1998)). Un exemple d'arbre préfixé est montré dans la figure 12.1. Dans cet arbre chaque chemin de la racine vers une feuille représente une séquence. On a deux types d'arcs (représentés dans la figure par des lignes continues et des lignes pointillées) selon que le nœud suivant fait partie du même itemset que le nœud précédent ou non. Par exemple, l'arbre de la figure 12.1 contient les séquences suivantes : $< (a) >, < (b, d) >, < (b)(e) >, < (c)(f) >$.

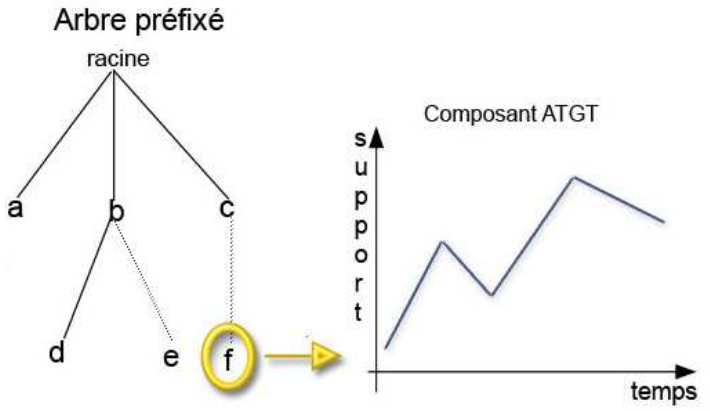


FIG. 12.1: Structure de stockage

12.2.2 Tableau de granularités temporelles adaptatives

Pour chaque motif séquentiel retenu dans l'arbre préfixé 12.2.1 on a peut-être besoin de connaître l'historique de son comportement (l'évolution de son support dans le temps). A chaque instant, on a une valeur du support de chaque motif séquentiel et

pendant la durée infinie d'un flux de données, cela génère une série de données infinie. En conséquent, il faut appliquer un algorithme afin de trouver le meilleur résumé en termes d'espace mémoire et pertinence du résultat.
Notre proposition est d'attacher à chaque nœud de l'arbre (implicitement à chaque motif représenté par le chemin de la racine vers ce nœud) un tableau. Ce tableau permet de gérer l'historique des motifs séquentiels fréquents en utilisant la technique Ami présentée dans le chapitre 11. Nous appelons un tel composant un **ATGT (Tableau de granularités temporelles adaptatives)**.

12.2.3 Exemple

La figure 12.2 illustre un exemple d'évolution des séquences au fil du temps. Suivons l'évolution de la séquence : $< c >< f >$. A l'instant t_0, elle a un support de 0,14. Donc, on attache au nœud correspondant à son dernier item (l'item f) un ATGT contenant la valeur 0,14 à l'instant t_0. Ensuite, à l'instant t_1, elle a un support de valeur 2 et on met à jour son ATGT en ajoutant cette valeur sur le graphique. De la même manière, on ajoute les valeurs 1,2 et 2,8 qui correspondent à son support pour les instants t_2 et t_3.

12.3 Discussion

On a vu qu'il est essentiel de trouver un bon compromis entre la qualité de l'approximation et le temps de réponse. Le composant ATGT vient répondre à ce problème en apportant une vision nouvelle sur les critères à considérer dans la condensation de l'information. ATGT va permettre de connaître la forme de la tendance générale au cours du temps et ne va pas afficher une tendance erronée par un facteur d'ancienneté (comme la majorité des algorithmes actuels le propose). Pour un utilisateur c'est beaucoup plus important de connaître la vraie tendance à une résolution plus basse que de connaître un résultat qui a perdu la forme générale. Par exemple, un supermarché est plus intéressé de savoir qu'il y a deux périodes de pics importantes dans les ventes de certains produits que d'avoir une approximation de la valeur moyenne des ventes il y a 8 à 12 mois (comme c'est le cas des algorithmes qui utilise un facteur d'ancienneté). De plus, l'algorithme de condensation de l'information utilisé par ATGT lui donne des avantages temporels incontestables et à un coût d'approximation très faible (voir le chapitre 11). Les avantages de la méthode d'approximation utilisée par ATGT ont

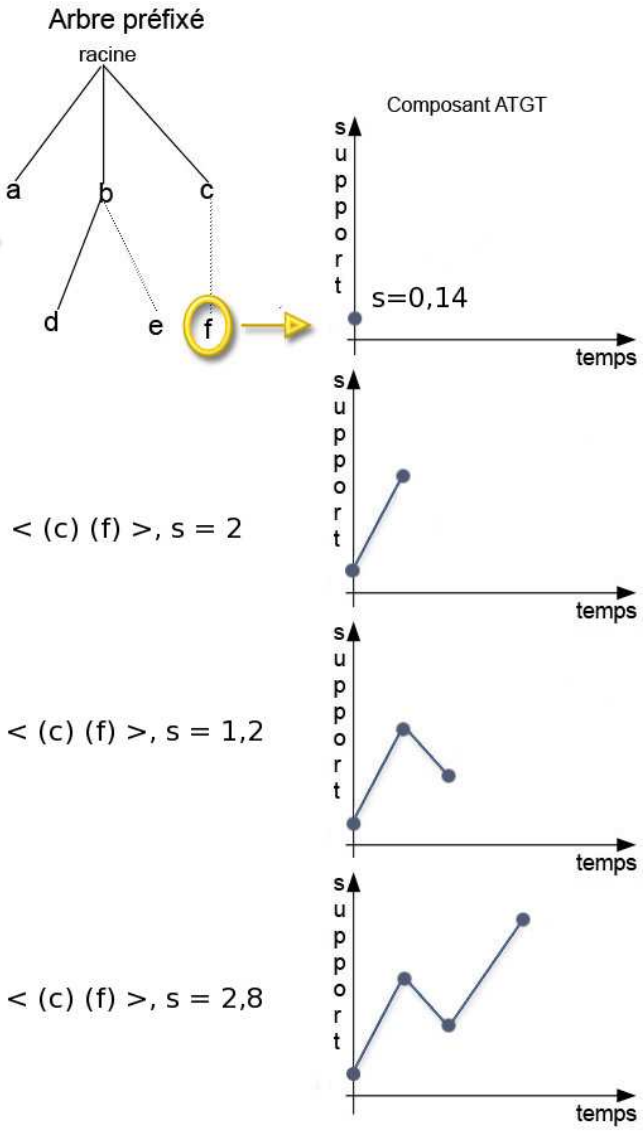
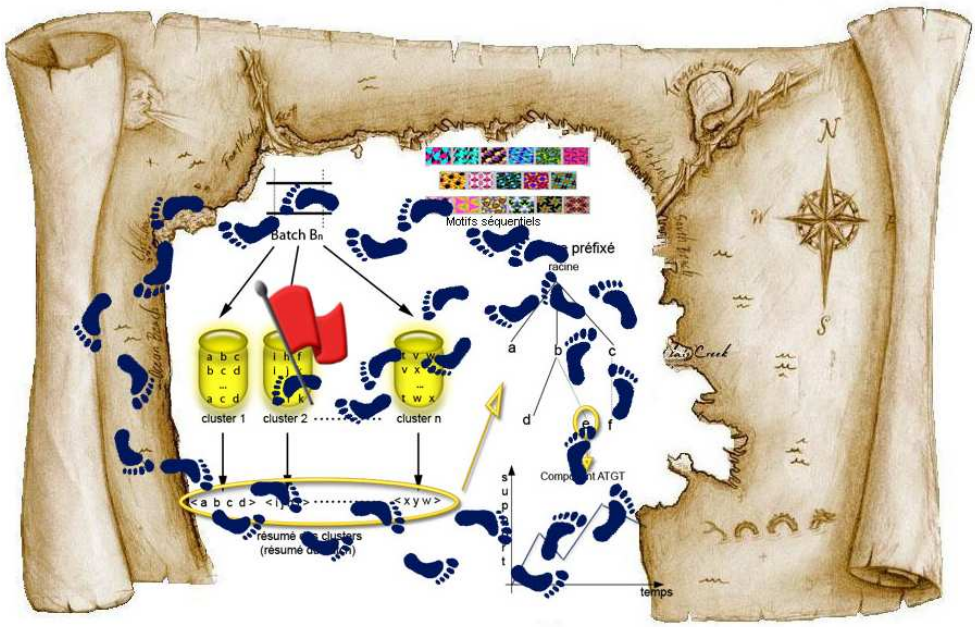


FIG. 12.2: Evolution d'une séquence dans le temps

été validés théoriquement et expérimentalement (voir le chapitre 11). En résumé, les avantages d'ATGT sont : une conservation de la forme globale de l'évolution d'une séquence dans le temps et une méthode d'approximation rapide et fiable. Un autre avantage d'ATGT est qu'en utilisant plusieurs composants ATGT concurremment, on a la possibilité d'identifier le meilleur composant à condenser et la meilleure partie à condenser de ce composant-là. Cette propriété est essentielle pour notre méthode d'extraction de motifs séquentiels dans les flux de données et on l'a beaucoup exploitée.

Après avoir vu notre choix de traitement de flux de données, une méthode qui résume des motifs séquentiels et une structure de stockage capable de gérer l'historiques des motifs séquentiels dans le temps, on a tous les éléments nécessaires pour passer à la présentation de la dernière pièce du puzzle : les méthodes de clustering. Le clustering permettra de regrouper les données similaires afin de pouvoir appliquer ensuite toutes les autres pièces de puzzle déjà présentées.

CARTE GENERALE



Le traitement des données d'un flux débute avec un regroupement des données dans des clusters. Trois algorithmes de clustering sont proposés et décrits en détail dans le chapitre qui suit.

CHAPITRE 13

CLUSTERING

13.1 Introduction

LES DONNES GENEREES par un flux de données ne sont triées d'après aucune caractéristique à part l'estampille temporelle. Elles arrivent sous forme d'amalgame de données hétérogènes. Beaucoup de techniques ont été proposées afin de trouver la meilleure forme d'organisation des données afin de permettre une extraction facile de l'information utile (voir la section 9.2). Notre idée est de commencer avec une étape d'organisation de données dans des groupes (des clusters) basée sur des mesures de similitude afin de faciliter la tâche d'extraction et d'organisation de l'information.

Nos méthodes sont basées sur un environnement de découpage du flux de données en batches de taille fixe. Soient $B_1, B_2, ...B_n$, les batches et B_n, le batch courant. Les batches sont traités l'un après l'autre. La première étape de notre méthode est une classification non-supervisée (un clustering) appliquée aux séquences contenues dans le batch courant. Cette étape est présentée en détail dans ce chapitre.

13.2 Contribution : 3 méthodes de clustering

On a mis en place trois algorithmes de clustering qui sont détaillés dans les sous-sections qui suivent.

13.2.1 Clustering I

Notre première version de schéma classificatoire est basique. Il s'agit d'un algorithme glouton qui fonctionne de la manière suivante : initialement l'algorithme ne contient aucun cluster. Pour chaque séquence s_n dans le batch courant, s_n est comparée avec chaque cluster existant c. Aussitôt que s_n est similaire à une séquence de c alors s_n est insérée dans c. Si s_n n'est insérée dans aucun cluster, alors un nouveau cluster est crée avec s_n. La similitude entre deux séquences, $sim(s_1, s_2)$, est donnée dans la définition 13.1. Une séquence s_n est insérée dans c si la condition suivante est respectée : $\exists s_c \in c / sim(s_n, s_c) \geq minSim$, avec $minSim$ la similitude minimum, spécifiée par l'utilisateur. L'algorithme de clustering permet d'obtenir des classes de séquences similaires, ce qui est un élément clé pour l'alignement des séquences qui sera appliqué aux clusters dans une étape suivante. Un nombre maximum fixe de clusters est gardé d'un batch à l'autre, ce numéro étant spécifié comme paramètre par l'utilisateur. L'utilisateur a la possibilité de changer cette valeur au parcours de l'exécution. L'algorithme basé sur cette méthode de clustering porte le nom de **SMDS (Sequence Mining in Data Streams).**

Definition 13.1 Soient s_1 et s_2 deux séquences. Soit $|PLSC(s_1, s_2)|$ la longueur de la plus longue sous-séquence commune entre s_1 et s_2. La distance $dist(s_1, s_2)$ entre s_1 et s_2 est définie de la manière suivante : $dist = 1 - \frac{2 \times |PLSC(s_1, s_2)|}{longueur(s_1) + longueur(s_2)}$. Analogiquement, la similitude $sim(s_1, s_2)$ entre s_1 et s_2 est définie comme étant $sim = \frac{2 \times |PLSC(s_1, s_2)|}{longueur(s_1) + longueur(s_2)}$.

L'algorithme SMDS parcourt les pas suivants :

Algorithme 1 (SMDS)
Input : $B = \cup_{i=0}^{\infty} B_i$: un ensemble infini de batches de transactions ; $minSize$: la taille minimum des classes à synthétiser ; $minSim$: la similitude minimum entre deux

séquences pour qu'un cluster soit mis à jour ; k : le filtre dans la méthode d'alignement[1].
Output : L'arbre préfixé mis à jour avec les séquences pondérées.
While (B) **Do**
$b \leftarrow$ NextBatch() ;
$C \leftarrow$ Clustering(b, $minSize$, $minSim$) ;// 1) Obtenir des cluster de taille $> minSize$ en appliquant l'algorithme de clustering décrit au-dessus
Foreach $(c \in C)$ **Do** // 2) Résumer chaque cluster avec un filtre k ;
$SA_c \leftarrow$ Alignment(c, k) ;//On calcule la séquence pondérée correspondante aux séquences c et k
If (SA_c) **Then** PrefixTree $\leftarrow$ PrefixTree+SA_c ; //3) Stocker les séquences pondérées
Done
// Séquences obsolètes
TailPruning(PrefixTree) ; // 4) Mettre à jour les Tableaux de Granularités Temporelles Adaptatives (ATGT). TailPruning concerne l'élagage des motifs qui n'apparaissent plus pour un certain nombre de batches (dans notre cas, 3 batches).
Done (fin Algorithme SMDS) ;

13.2.1.1 Complexité

Soit n le nombre de séquences du batch. Dans le pire des cas, l'algorithme de clustering a une complexité en temps de $O(n^2)$. En fait, dans le pire des cas, PLSC est appliquée une fois pour la première séquence, deux fois pour la deuxième, et ainsi de suite. La complexité est donc $O(\frac{n.(n+1)}{2}) = O(n^2)$. Cependant, même si cette complexité peut être améliorée (pour le pire des cas) la méthode est très adaptée pour les motifs de navigation sur le Web et nos expérimentations (C.f. section 13.2.3.2) sur des données réelles (accès logs de l'Inria Sophia Antipolis) ont montré son efficacité.

13.2.1.2 Experimentations

SMDS a été implémentée en Java sur un Pentium (2,1 Ghz) exploité par un système Linux Fedora. Nous avons évalué notre proposition sur des données synthétiques et des

[1] Il aide à filtrer les séquences pondérées et de retenir seulement les items dont le nombre d'apparitions est plus élevé que la valeur du filtre.

données réelles[1].

Temps de réponse et robustesse de SMDS

Dans le but de montrer l'efficacité de SMDS, nous reportons dans la figure 13.1 le temps nécessaire pour extraire les motifs séquentiels approximatifs les plus longs sur chaque batch correspondant à des données d'usage du Web (à gauche de la figure 13.1) et des données synthétiques (à droite de la figure 13.1). Pour le site Web de l'Inria, les données ont été collectées sur une période de 14 mois et représentent 14 Go. Le nombre total de transactions est de 3,5 millions pour 300000 navigations. Nous avons découpé le fichier log en batches de 4500 transactions (soit environ 1500 séquences en moyenne). Pour ces expérimentations, le filtre k est fixé à 30% (notons que ce filtre a un impact sur les temps d'exécution, dans la mesure où il modifie la taille des séquences à gérer dans l'arbre préfixé). De plus, nous avons "injecté" dans les données des séquences parasites. Le premier batch ne subit pas de modification. Dans le second, nous ajoutons dix séquences contenant deux répétitions de deux items. Dans le troisième batch, nous ajoutons dix séquences de trois répétitions, et ainsi de suite jusqu'au trentième batch qui contient 10 séquences de trente répétitions. L'objectif est de montrer que les méthodes d'extraction traditionnelles (PSP Masseglia *et al.* (1998), prefixSpan Pei *et al.* (2001a), ...) risquent de bloquer le data stram alors que SMDS continuera sa tâche d'extraction. Nous pouvons observer que le temps de réponse de SMDS varie de 1800 ms à 3100 ms. PSP propose des motifs avec de très bonnes performances pour les premiers batches et se trouve pénalisé par le bruit ajouté par les séquences parasites (voir le batch 19). Le test a également été fait avec prefixSpan et le comportement exponentiel est similaire. Pour PSP comme pour prefixSpan, le support minimum spécifié était le maximum possible tout en assurant que les séquences "parasites" (répétitions) seraient trouvées. Nous avons ajouté à la figure 13.1 le nombre de séquences de chaque batch pour expliquer les différences de temps d'exécution d'un batch à un autre. On peut observer, par exemple, que le batch 1 contient 1500 séquences et que SMDS demande 2700 ms pour en extraire les motifs séquentiels. Pour les données synthétiques, nous avons généré des batches de 10000 transactions (qui correspondent à environ 500 séquences en moyenne). La longueur moyenne des séquences était de 10 pour 200000 items. Le filtre k est fixe à

[1] Le générateur de données synthétiques peut être téléchargé sur http ://www.almaden.ibm.com/cs/quest.

30%. Nous indiquons dans la figure 13.1 (partie droite) les temps de réponse et le nombre de séquences correspondant à chaque batch. Nous pouvons observer que SMDS traite 10000 transactions en moins de 4 secondes (par exemple pour le batch 2).

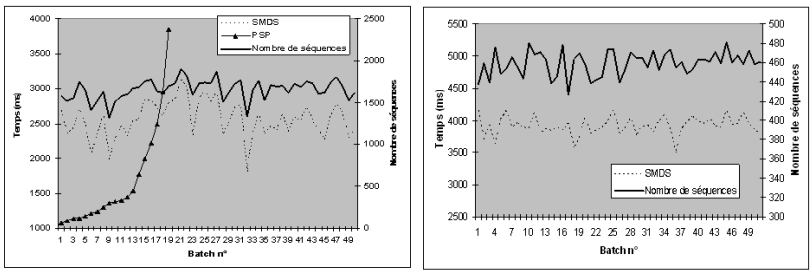


FIG. 13.1: Temps d'exécution de SMDS.

Motifs extraits sur les données réelles

La liste des comportements découverts par SMDS couvre plus de 100 objectifs de navigation (classes de séquences de navigations) sur le site Web de l'Inria Sophia Antipolis. La plupart des motifs découverts peuvent être considérée comme rare (support faible) et pertinent (haute confiance, car le filtre utilise est élevé). Nous reportons ici quelques exemples de ces comportements.

A) $k = 30\%$, taille de la classe = 13, prefixe="http ://www-sop.inria.fr/omega/" :
< (MC2QMC2004) (personnel/Denis.Talay/moi.html) (MC2QMC2004/presentation.html) (MC2QMC2004/dates.html) (MC2QMC2004/Call_for_papers.html)>
Cette séquence est fréquente en juin 2004, lors de l'organisation de la conférence MCQMC par une équipe de l'Inria Sophia Antipolis. Ce comportement était partagé par au plus 13 utilisateurs en même temps (taille de la classe = 13).

B) $k = 30\%$, taille de la classe = 10, prefixe="http ://www-sop.inria.fr/acacia/personnel/itey/Francais/Cours/" : < (programmation-fra.html) (PDF/chapitre-cplus.pdf)
(cours-programmation-fra.html) (programmation-fra.html) >
Ce comportement correspond aux requêtes faites sur un document pédagogique sur la programmation écrit par un membre d'une équipe de l'Inria Sophia Antipolis. Il correspond à la période d'avril 2004.

Pour les navigations sur le site de l'Inria Sophia Antipolis, nous avons également constaté que SMDS est capable de détecter les séquences parasites qui avaient été injectées dans les batches. Ces séquences sont simplement regroupées par SMDS dans un cluster qui ne contient qu'elles, sans impact sur les temps d'exécution.

Impact de la taille des batches

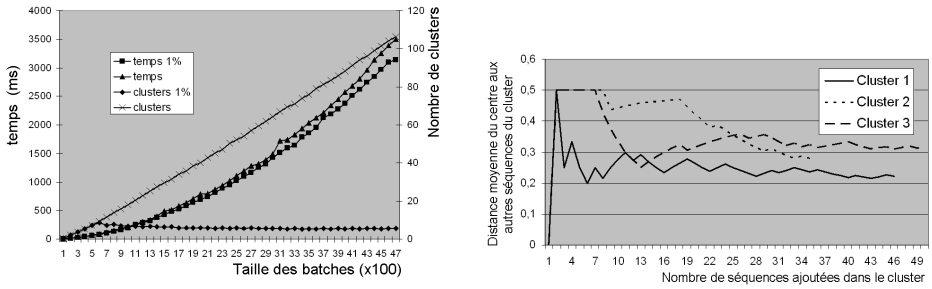


FIG. 13.2: Taille des batches et le pire cluster

Puisque la complexité de SMDS dépend de la taille de batches, nous avons mené une étude concernant l'impact de cette taille sur les temps de réponse. Nous reportons dans la figure 13.2 (partie gauche) les temps de réponses quand S (le nombre de séquences du batch) varie de 100 à 5000 séquences. La courbe *temps* représente les temps d'exécution, *cluster* représente les nombres de clusters extraits, *cluster* 1% représente le nombre de clusters tels que : $|c| > 0.01 \times S$. En effet, nous pensons qu'il ne faut considérer que les clusters dont la taille est supérieure à une certaine proportion du nombre total de séquences (on ne garde que les clusters les plus grands). Avec 1% et un batch de 1000 séquences, par exemple, un cluster c tel que $|c| < 10$ ne sera pas considéré. *time* 1% représente le temps d'exécution quand on ne garde que les clusters de taille supérieure à 1% de S. On peut observer que le nombre de clusters augmente de façon linéaire. Le temps de réponse est de toute évidence lié à la taille des batches, mais il est raisonnable de dire que l'utilisateur final peut choisir la taille de ses batches en fonction du degré de rapidité souhaité.

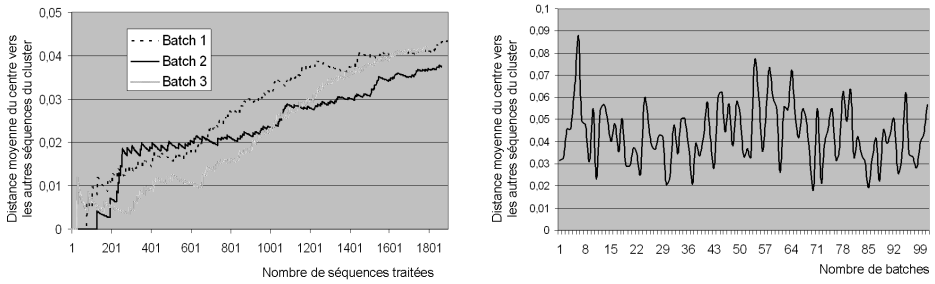


FIG. 13.3: Distance globale étape par étape et batch par batch

Analyse de la qualité des clusters

Afin de mesurer la qualité des classes produites par SMDS, notre principal outil sera la distance entre deux séquences. Soit s_1 et s_2, deux séquences, la distance $dist(s_1, s_2)$ entre s_1 et s_2 est basée sur $sim(s_1, s_2)$, la mesure de similitude donnée par la définition 13.1 et telle que $dist(s_1, s_2) = 1 - sim(s_1, s_2)$. On a donc $dist(s_1, s_2) \in [0..1]$ et $dist(s_1, s_2)$ proche de 0 signifie que les séquences sont proches (similaire si cette valeur est nulle) alors que $dist(s_1, s_2)$ proche de 1 signifie que les séquences sont éloignées (ne partagent aucun item si cette valeur est 1). Nous avons utilisé deux mesures pour la qualité des classes. La première est le diamètre d'une classe C. Il s'agit de la plus grande distance entre deux séquences de C. Un diamètre de 0% montre que la classe est constituée uniquement de séquences égales alors qu'un diamètre de 100% montre que la classe contient au moins deux séquences qui ne partagent aucun item. Lors de nos expérimentations, le diamètre moyen a varié entre 2% et 3%. La seconde mesure est la "double moyenne". Elle est basée sur le centre de la classe. Soit C une classe, le centre de C est une séquence c telle que : $\forall s \in C, \sum_{x \in C} dist(s, x) \geq \sum_{y \in C} dist(c, y)$. Nous sommes donc en mesure de donner, pour C, la distance moyenne (DM) entre c et toutes les autres séquences de C : $DM = \frac{\sum_{x \in C} dist(x,c)}{|C|}$. Nous reportons dans la figure 13.2 (partie droite) quelques distances moyennes parmi les pires obtenues durant nos expérimentations. Pour chaque séquence ajoutée dans une classe, nous donnons la valeur DM de cette classe. Par exemple, après l'ajout de la dernière séquence dans le cluster 1, la valeur de DM pour ce cluster est 22%. On peut observer que DM varie de 0 (quand $|C| = 1$ l'unique séquence est le centre) à 50%. DM décroit alors rapidement jusqu'à des valeurs comprises entre 20% et 35%, ce qui est un bon

résultat compte tenu du fait que la figure 13.2 (partie droite) ne comprend que les résultats des clusters les moins homogènes. Les autres clusters sont homogènes et offrent à l'algorithme d'alignement un cadre adéquat. Nous reportons dans la figure 13.3 (partie gauche) la double moyenne DBM après avoir traité chaque séquence d'un batch. DBM est calculée de la manière suivante : soit C l'ensemble des classes, $DBM = \frac{\sum_{i \in C} \sum_{x \in C_i} dist(x,c_i)}{|C|}$ avec c_i le centre de C_i (la i^{eme} classe). On peut observer dans la figure 13.3 (partie gauche) que pour le second batch, DBM augmente rapidement jusqu'à 2% (séquence 220), puis augmente lentement jusqu'à 3,7%. La valeur finale de DBM à la fin du batch est donnée par la figure 13.3 (partie droite). On peut y observer que DBM est toujours comprise entre 2% et 9%. A la fin du processus, la valeur moyenne de DBM est de 4,3% (une qualité moyenne des classes de 95,7%).

Ce premier algorithme de clustering, ainsi que les éléments présentés dans les chapitres 9 et 10, a donné lieu aux publications suivantes :

- MARASCU, A. & MASSEGLIA, F. (2005). Mining Sequential Patterns from Temporal Streaming Data. In *ECML/PKDD first Workshop on Mining Spatio-Temporal Data (MSTD'05). Held in conjunction with PKDD'05,* Porto, Portugal.
- MASSEGLIA, F. & PONCELET, P. & TEISSEIRE, M. & MARASCU, A. (2005). Web Usage Mining : Extracting Unexpected Periods from Web Logs. In *IEEE 2nd Workshop on Temporal Data Mining (TDM'05). Held in conjunction with ICDM'05,* Houston, USA.
- MARASCU, A. & MASSEGLIA, F. (2005). Mining Data Streams for Frequent Sequences Extraction. In *IEEE first Workshop on Mining Complex Data (MCD'05). Held in conjunction with ICDM'05,* Houston, USA.
- MARASCU, A. & MASSEGLIA, F. (2006). Extraction de motifs séquentiels dans les flots de données d'usage du Web, *Extraction et Gestion des Connaissances* , Lille, France.
- MASSEGLIA, F. & PONCELET, P. & TEISSEIRE, M. & MARASCU, A. (2006). Usage Mining : extraction de périodes denses à partir des logs Web. *Extraction et Gestion des Connaissances,* Lille, France.
- MASSEGLIA, F. & PONCELET, P. & TEISSEIRE, M. & MARASCU, A. (2008). Web Usage Mining : Extracting Unexpected Periods from Web Logs. In *Data*

Mining and Knowledge Discovery (DMKD) Journal. Springer Netherlands (Ed). 16(1) : 39-65.

13.2.2 Clustering II

Notre deuxième méthode de clustering est une approche centroïde qui permet de maintenir un représentant de chaque classe.
Notre approche fonctionne de la manière suivante : initialement l'algorithme ne contient aucun cluster. Ensuite, pour chaque navigation n dans le batch, s_n est comparée avec chaque cluster c. Soit c le cluster dont le centroïde est le plus proche de s_n, alors s_n est insérée dans c. Si s_n n'est insérée dans aucun cluster, alors un nouveau cluster est crée avec s_n.
Trois éléments sont essentiels dans notre processus de clustering. Le premier est le calcul du centroïde ς_c d'un cluster c. Ce calcul est détaillé dans la section 13.2.2.1. Ensuite pour comparer la séquence de navigation s_n avec le cluster c, nous proposons de définir la similitude entre s_n et le centroïde de c. Cette étape est expliquée dans la section 13.2.2.3. Cet algorithme porte le nom de **SCDS (Sequence Clustering in Data Streams)**.

13.2.2.1 Le centroïde d'un cluster

Le centroïde d'un cluster est une séquence qui est représentative de toutes les séquences contenues dans le cluster, autrement dit c'est un résumé de toutes les séquences contenues dans le cluster.
La séquence pondérée présentée dans la section 10.3 et réprise dans la figure 13.4 résume un ensemble de séquences et les avantages présentés dans la section 10.4 en font un excellent candidat pour le centroïde d'un cluster qui évolue dans un environnement dynamique. Au tout début de la formation d'un nouveau cluster, le centre du cluster est représenté par la première séquence insérée et au fur et à mesure que d'autres séquences sont ajoutées au même cluster, la séquence-centroïde est mise à jour. A l'ajout d'une nouvelle séquence à un cluster, on calcule un alignement entre l'ancien centroïde et cette nouvelle séquence ajoutée ; le résultat (une séquence pondérée) devient le nouveau centroïde. La section 13.2.2.2 décrit d'autres opérations faites lors de l'ajout d'une séquence. Les opérations de mises à jour du centroïde, c'est-à-dire les alignements, se font de manière incrémentale à chaque ajout d'une séquence dans le cluster.

Etape 1 :				
S_1 :	<(a,c)	(e)	()	(m,n)>
S_2 :	<(a,d)	(e)	(h)	(m,n)>
SA_{12} :	(a :2, c :1, d :1) :2	(e :2) :2	(h :1) :1	(m :2, n :2) :2
Etape 2 :				
SA_{12} :	(a :2, c :1, d :1) :2	(e :2) :2	(h :1) :1	(m :2, n :2) :2
S_3 :	<(a,b)	(e)	(i,j)	(m)>
SA_{13} :	(a :3, b :1, c :1, d :1) :3	(e :3) :3	(h :1, i :1, j :1) :2	(m :3, n :2) :3
Etape 3 :				
SA_{13} :	(a :3, b :1, c :1, d :1) :3	(e :3) :3	(h :1, i :1, j :1) :2	(m :3, n :2) :3
S_4 :	<(b)	(e)	(h,i)	(m)>
SA_{14} :	(a :3, b :2, c :1, d :1) :4	(e :4) :4	(h :2, i :2, j :1) :3	(m :4, n :2) :4

FIG. 13.4: Etapes de l'alignement de séquences

13.2.2.2 Contribution : structure permettant de détecter rapidement la dégradation du centroide

Comme mentionné dans la section 10.4, l'ordre des séquences influence la qualité du résultat. Afin de résoudre ce problème j'ai mis en place une structure qui détecte la dégradation de la qualité du centroïde et donne le signal de rafraîchissement des alignements.

La structure est composée d'une **matrice de comptage** des items dans chaque séquence et d'un **tableau des distances** entre chaque séquence et les autres. Ces éléments sont illustrés par la figure 13.5. La matrice (à gauche) stocke pour chaque séquence le nombre d'apparitions de chaque item dans cette séquence. Par exemple la séquence s_1 contient deux fois l'item a. Le tableau des distances stocke la somme des similitudes ($similMatrice$) entre chaque séquence et les autres séquences du cluster. Soit s_{1_i} le nombre d'apparitions de l'item i dans la séquence s_1 et m le nombre total d'items. $similMatrice$ est calculé grâce à la matrice de la manière suivante :

$$similMatrice(s_1, s_2) = \sum_{i=1}^{m} min(s_{1_i}, s_{2_i}).$$

Par exemple, avec les séquences s_1 et s_2 de la matrice donnée à la figure 13.5 cette somme vaut $s_{1_a} + s_{2_b} + s_{2_c} = 1 + 0 + 1 = 2$.

Cet alignement n'est cependant pas toujours calculé de manière incrémentale. Considérons l'ajout d'une séquence s_n. Tout d'abord s_n est ajoutée à la matrice et sa distance aux autres séquences est calculée ($\sum_{i=1}^{n} similMatrice(s_n, s_i)$). s_n est alors insérée dans le tableau de distances, en gardant l'ordre décroissant des valeurs de distances. Par exemple, dans la figure 13.5, s_n est insérée après s_2. Soit r le rang auquel s_n est insérée (dans notre exemple, $r = 2$) dans c. Il y a alors deux possibilités après l'insertion de s_n [1] :

1. $r > 0.5 \times |c|$. Dans ce cas, l'alignement est calculé de manière incrémentale et $\varsigma_c = alignement(\varsigma_c, s_n)$.
2. $r \leq 0.5 \times |c|$. Dans ce cas il faut rafraîchir le centroïde du cluster. Pour cela, les séquences sont d'abord triées par densité. Ensuite, l'alignement est recalculé pour toutes les séquences du cluster.

Séq	a	b	c
s_1	2	0	1
s_2	1	0	1
⋮			

Séq	$\sum_{i=1}^{n} similMatrice(s, s_i)$
s_1	16
s_2	14
s_n	13
s_3	11
⋮	
s_{n-1}	1

FIG. 13.5: Distances entre les séquences

13.2.2.3 Comparaison d'une séquence avec un centroïde

Soit s la séquence courante à affecter dans un cluster et C l'ensemble de clusters existants à un moment donné. Afin de trouver le cluster le plus similaire à cette séquence, une série de comparaisons sont effectuées. Comme pour chaque cluster on a trouvé un représentant des séquences contenues, le centroïde, il suffit de comparer le centroïde avec la nouvelle séquence à affecter. Donc, il faut parcourir l'ensemble des clusters de

[1] Le paramètre 0.5 est utilisé afin de séparer les données en moitié.

C et pour chaque cluster $c \in C$, il faut effectuer une comparaison entre s et ς_c (le centroïde de c, qui est donc un alignement). Cette comparaison est basée sur *la plus longue sous-séquence commune (PLSC)* entre s et ς_c. La similitude entre deux séquences ($sim(s_1, s_2)$) est donnée dans la définition 13.1. Ensuite, la longueur de la séquence est également prise en compte car elle doit être comprise entre 80% et 120% de la longueur de la première séquence du cluster.

Soit t la longueur de la première séquence insérée dans c. Les conditions pour que s soit affectée à c sont donc les suivantes :

- $\forall d \in C / d \neq c, dist(s, \varsigma_c) \leq dist(s, \varsigma_d)$: Cette condition assure que s est affectée dans le cluster dont le centroïde est le plus similaire à s ;
- $0.8 \times t \leq |s| \leq 1.2 \times t$: Cette condition assure que les clusters contiendront des séquences de taille homogène et que la taille moyenne des séquences d'un cluster variera peu ;
- $dist(s, \varsigma_c) < 0.3$: Si aucun centroïde ayant un degré de similitude supérieur à 70% avec s n'est trouvé, alors s n'est affecté à aucun cluster ; dans ce cas, un nouveau cluster est crée et s y est affectée.

Le pseudo-code de l'algorithme est le suivant :

Algorithme 1 (SCDS)
Input : $B = \cup_{i=0}^{\infty} B_i$: un ensemble infini de batches de transactions ; $minSize$: la taille minimum des classes à synthétiser ; $minSim$: la similitude minimum entre deux séquences pour qu'un cluster soit mis à jour ; k : le filtre dans la méthode d'alignement[1].
Output : L'arbre préfixé mis à jour avec les séquences pondérées.
While (B) **Do**
$b \leftarrow$ NextBatch() ;
$C \leftarrow$ Clustering($b, minSize, minSim$) ;// 1) Obtenir des clusters de taille $> minSize$
Foreach ($c \in C$) **Do** // 2) Résumer chaque cluster avec un filtre k ;
$SA_c \leftarrow$ centroïd(c, k) ;
If (SA_c) **Then** PrefixTree $\leftarrow$ PrefixTree+SA_c ; //3) Stocker les centroïdes
Done
// Séquences obsolètes

[1] Il aide à filtrer les séquences pondérées et de retenir seulement les items dont le nombre d'apparitions est plus élevé que la valeur du filtre.

TailPruning(PrefixTree) ; // 4) Mettre à jour les Tableaux de Granularités Temporelles Adaptatives (ATGT). TailPruning concerne l'élagage des motifs qui n'apparaissent plus pour un certain nombre de batches (dans notre cas, 3 batches)
Done (fin Algorithme SCDS) ;

13.2.2.4 Expérimentations

SCDS a été implémenté en Java sur un Pentium (2,1 Ghz) exploité par un système Linux Fedora. Nous avons évalué notre proposition sur des réelles issues des usages du Web de l'Inria Sophia Antipolis.

Temps de réponse et robustesse de SCDS

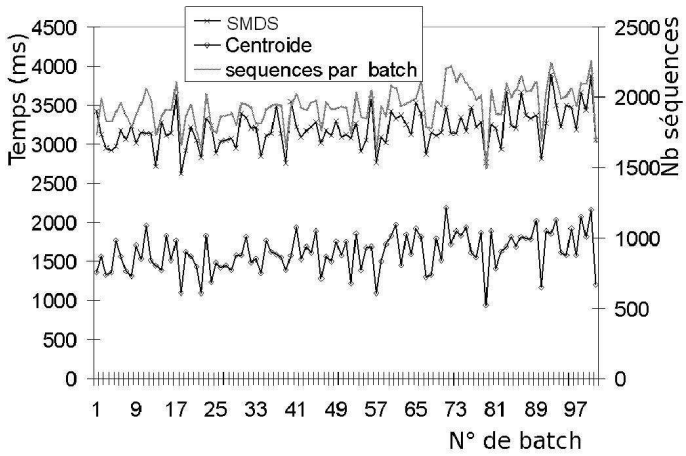


FIG. 13.6: Temps d'exécution de SCDS et comparaison de temps d'exécution avec SMDS.

Dans le but de montrer l'efficacité de SCDS, nous reportons à la figure 13.6 le temps nécessaire pour classer les séquences sur chaque batch correspondant à des données d'usage sur le site Web de l'Inria. Les données ont été collectées sur une période de 14 mois et représentent 14 Go. Le nombre total de navigations est de 3,5 millions pour 300000 navigations. Nous avons découpé le fichier log en batches de 4500 transactions (soit environ 1500 séquences en moyenne). Nous avons comparé ce temps de réponse à celui de SMDS qui n'optimise pas la phase de clustering. En effet dans SMDS, la

séquence à classer s est comparée à toutes les séquences de tous les clusters, jusqu'à ce que l'un des clusters présente une séquence compatible avec s. Nous pouvons observer que le temps de réponse de SCDS varie de 1000 ms à 2000 ms alors que le temps d'exécution de SMDS varie de 2500 ms à 4000 ms. Nous avons ajouté à la figure 13.6 le nombre de séquences de chaque batch pour expliquer les différences de temps d'exécution d'un batch à un autre. On peut observer, par exemple, que le batch 1 contient 1750 séquences et que SCDS demande 1400 ms pour en extraire les motifs séquentiels.

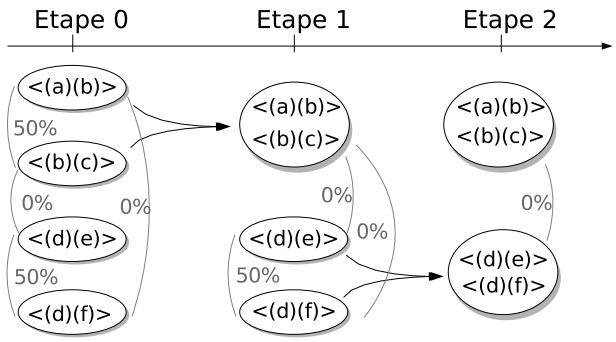


FIG. 13.7: Clustering hiérarchique des séquences d'un batch.

Nous avons également implémenté un clustering hiérarchique sur les séquences de chaque batch. Le principe de ce clustering est décrit par la figure 13.7. Chaque séquence du batch est d'abord considérée comme un cluster (voir "étape 0" de la figure 13.7). A chaque étape la matrice des similitudes entre chaque cluster est calculée. Par exemple entre $<(a)(b)>$ et $<(b)(c)>$ la similitude est de 50%, les deux séquences partagent en effet la moitié de leurs informations. Entre $<(a)(b)>$ et $<(d)(e)>$ en revanche, la similitude est de 0%. Les deux séquences n'ont rien en commun. Les deux clusters les plus proches (ici il s'agit d'un ex-aequo entre $\{<(a)(b)>, <(b)(c)>\}$ d'un côté et $\{<(d)(e)>, <(d)(f)>\}$ de l'autre) sont regroupés en un seul cluster. L'étape "1" de la figure 13.7 nous montre en effet trois clusters : $\{<(a)(b)>, <(b)(c)>\}$, $\{<(d)(e)>\}$ et $\{<(d)(f)>\}$. Ce processus est alors réitéré jusqu'à ce que plus aucun cluster n'affiche une similitude supérieure à zéro avec au moins un des clusters restants. L'étape "2" de la figure 13.7 nous montre donc le résultat de cette classification : $\{<(a)(b)>, <(b)(c)>\}$ et $\{<(d)(e)>, <(d)(f)>\}$.

La comparaison avec les temps d'exécution du clustering hiérarchique est reportée à la figure 13.8 pour les 10 premiers batches. On peut y observer que SCDS obtient les résultats en un temps compris entre 1000 ms et 2000 ms. Le clustering hiérarchique nécessite entre 11000 ms et 17000 ms. Plus précisément, le temps d'exécution moyen de SCDS est de 1485 ms, contre 12451 ms pour le clustering hiérarchique.

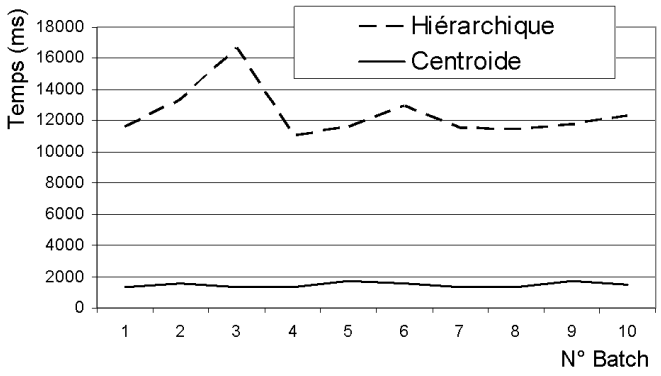


FIG. 13.8: Temps de réponse du clustering hiérarchique et de SCDS pour 10 batches

Analyse de la qualité des clusters

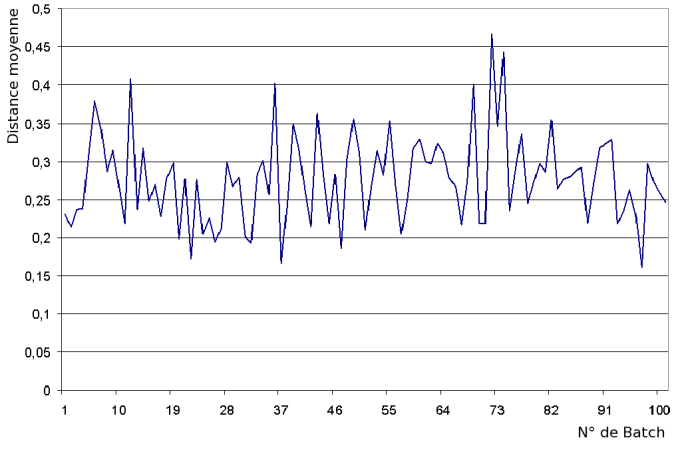


FIG. 13.9: Distance globale, batch par batch

Afin de mesurer la qualité des classes produites par SCDS, notre principal outil sera

la distance entre deux séquences. Soit s_1 et s_2, deux séquences, la distance $dist(s_1, s_2)$ entre s_1 et s_2 est donnée par la définition 13.1. On a donc $dist(s_1, s_2) \in [0..1]$ et $dist(s_1, s_2)$ proche de 0 signifie que les séquences sont proches (similaires si cette valeur est nulle) alors que $dist(s_1, s_2)$ proche de 1 signifie que les séquences sont éloignées (ne partagent aucun item si cette valeur est 1). Nous reportons dans la figure 13.9 la double moyenne DBM après avoir traité chaque batch. Soit C l'ensemble des classes, DBM est calculée de la manière suivante :

$DBM = \frac{1}{|C|} \sum_{i \in C} \frac{\sum_{x \in C_i} dist(x, c_i)}{|C_i|}$

La valeur finale de DBM à la fin du batch est donnée par la figure 13.3. On peut y observer que DBM est comprise entre 15% et 45%. A la fin du processus, la valeur moyenne de DBM est de 28% (une qualité moyenne des classes de 72%).

Afin de compléter notre étude de la qualité des clusters obtenus avec SCDS, nous avons utilisé les mesures d'entropie et pureté, par comparaison avec les clusters obtenus par le clustering hiérarchique. L'entropie d'un cluster C de taille n_r est calculée selon la formule suivante : $E(C) = -\frac{1}{logq} \sum_{i=1}^{q} \frac{n_r^i}{n_r} log \frac{n_r^i}{n_r}$, où q est le nombre total de clusters et n_r^i est le nombre de séquences du i^{eme} cluster qui font partie du cluster C. L'entropie du clustering est ensuite donnée par la formule :

$Entropie = \sum_{r=1}^{k} \frac{n_r}{n} E(C_r)$, avec n le nombre total de séquences. On considère qu'une petite valeur d'entropie traduit un bon clustering (par rapport au clustering de référence).

La pureté d'un cluster est donnée définie par : $P(C_r) = \frac{1}{n_r} max(n_r^i)$ et la pureté du clustering est donnée par la formule suivante : $Purete = \sum_{r=1}^{k} \frac{n_r}{n} P(C_r)$. Une grande valeur de pureté traduit un bon clustering par rapport au clustering de référence.

Les valeurs que nous avons obtenues pour l'entropie et la pureté sur les 10 premiers batches sont données dans le tableau suivant et dans la figure 13.10 :

Batch	Entropie	Pureté
1	0,012523916	0,95107037
2	0,013794152	0,94326377
3	0,017435603	0,93279576
4	0,015584618	0,93603873
5	0,016993482	0,93243974
6	0,018524481	0,92739403
7	0,0202273	0,9281229
8	0,015651526	0,9380881
9	0,017332898	0,93359625
10	0,01842611	0,93085754

On peut observer que la valeur de l'entropie se situe entre 0.012 et 0.02 et que la valeur de la pureté se situe entre 0.92 et 0.95. Lors de ces expérimentations, la valeur moyenne de l'entropie a été de 0.0166 et la valeur moyenne de la pureté a été de 0.9353. Ces valeurs attestent une très bonne qualité du clustering.

Ce deuxième algorithme de clustering, ainsi que les éléments présentés dans les chapitres 9 et 10, a donné lieu aux publications suivantes :

- MARASCU, A. & MASSEGLIA, F. (2006). Classification de flots de séquences basée sur une approche centroïde, *INFORSID Informatique des organisations et systèmes d'information et de décision*, Hammamet, Tunisie.
- MARASCU, A. & MASSEGLIA, F. (2006). Mining Sequential Patterns from Data Streams : a Centroid Approach. In *Journal for Intelligent Information Systems (JIIS)*. Issue 27, Number 3, pp 291-307.
- MARASCU, A. & MASSEGLIA, F. (2006). Classification de flots de séquences basée sur une approche centroïde, In *Fouille de données complexes dans un processus d'extraction des connaissances (FDC'06)*, Lille, France, pp 131-139.

13.2.3 Clustering III

A l'heure actuelle il existe très peu de méthodes de classification non supervisée des séquences dans un flux de données. Cette version apporte un côté incrémental et semi-supervisée à la méthode précédente de clustering et a reçu le nom de ICDS (Incremental Clustering in Data Streams). ICDS reprend les étapes principales de SCDS, mais apporte une modification au clustering des séquences contenues dans un batch. Cette modification est présentée en détail dans la section suivante.

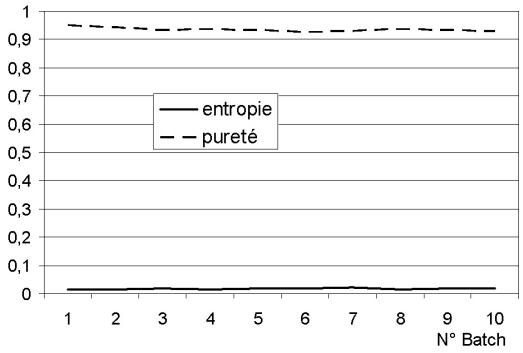


FIG. 13.10: Evaluation de l'entropie et de la pureté

13.2.3.1 Aspect incrémental

Dans ICDS, nous avons implémenté une méthode de clustering incrémentale des séquences (ce qui la rend semi-supervisée) comme illustré à la figure 13.11. Le clustering commence au batch B_0 pour lequel aucune information n'est disponible et nous obtenons un découpage du batch en plusieurs clusters. Ce découpage est conservé lors du passage au batch B_1. Chaque séquence de B_1 est alors comparée aux centroïdes des clusters obtenus sur le batch B_0. A la fin du traitement de B_n (le n^{eme} batch), le clustering peut avoir évolué par rapport à celle de B_{n-1} selon trois cas possibles :

1. Modification du centroïde d'un cluster. Dans ce cas, le contenu du cluster a changé et son centroïde s'est déplacé (l'alignement est différent avec des éléments nouveaux, des poids différents, une modification de leur ordonnancement, etc.).
2. Disparition de clusters. Dans le cas où un cluster deviendrait trop petit pour être pris en compte, ou bien si il n'accueille aucune séquence, il disparaît.
3. Apparition de clusters. Un nouveau cluster apparaît, signe d'une évolution importante des comportements.

Et ce traitement continue, en faisant évoluer le clustering au fil du flux. Le pseudo-code d'ICDS est donné ci-dessous :

Algorithm (ICDS)
Input : $B = \cup_{i=0}^{\infty} B_i$: un ensemble infini de batches de transactions ; $minSize$: la

taille minimum des classes à synthétiser ; $minSim$: la similitude minimum entre deux séquences pour qu'un cluster soit mis à jour ; k : le filtre dans la méthode d'alignement[1].
Output : L'arbre préfixé mis à jour avec les séquences pondérées.
$c \leftarrow \oslash$;
While (B) **Do**
$b \leftarrow$ NextBatch() ;
$C \leftarrow$ Clustering(c, b, $minSize$, $minSim$) ; // 1) Obtenir des cluster de taille //> $minSize$ et marquer les clusters non utilisés
Foreach ($c \in C$) **Do** // 2) Résumer chaque cluster avec un filtre k ;
If (notUsed(c)) **Then** Delete(c) ; // Disparition de clusters
Else DeleteOldAlignment(c) ; // Apparition ou évolution de clusters
$SA_c \leftarrow$ Alignment(c, k) ;
PrefixTree $\leftarrow$ PrefixTree+SA_c ; //3) Stocker les séquences pondérées
Done
Done (fin Algorithm ICDS) ;

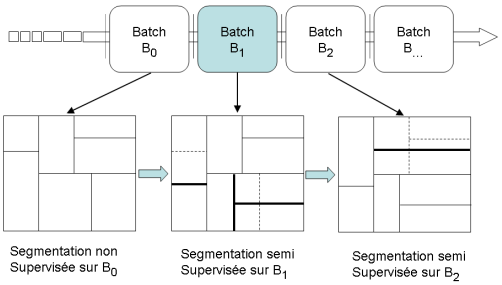


FIG. 13.11: Principe du clustering incrémental dans un flux de données

Au début de l'algorithme, C, le résultat du clustering contient un ensemble vide de clusters. Cet ensemble est alors mis à jour par la méthode "Clustering(c,b, $minSize$, $minSim$)" qui prend comme variables d'entrée c, le résultat du clustering au batch précédent (au début de l'algorithme, il n'y a pas eu de batch précédent, donc le paramètre c est ignoré) ; b, les séquences du batch ; $minSize$,la taille minimum d'un cluster

[1]Il aide à filtrer les séquences pondérées et de retenir seulement les items dont le nombre d'apparitions est plus élevé que la valeur du filtre.

à conserver en fin de traitement ; et enfin $minSim$, la similitude minimum pour affecter une séquences dans un cluster.
Pour les batches suivants, le traitement est similaire et pour chaque appel à la méthode "Clustering" le résultat du clustering sur le batch précédent est pris en compte.

13.2.3.2 Expérimentations

ICDS a été implémenté en Java sur un Pentium (2,1 Ghz) exploité par un système Linux Fedora. Nous avons évalué notre proposition sur des données réelles issues des usages du Web de l'Inria Sophia Antipolis.

Temps de réponse comparés de ICDS et SCDS

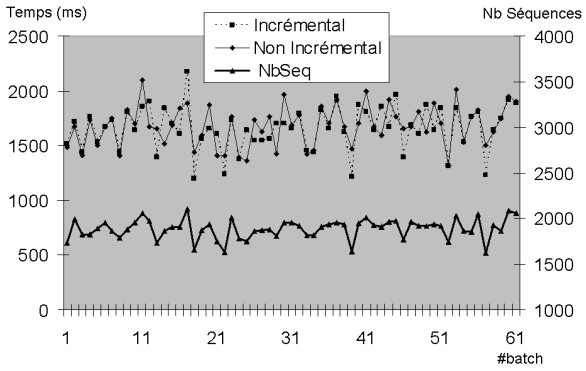


FIG. 13.12: Temps d'exécution de SCDS.

Dans le but de comparer les temps de réponse d'ICDS de SCDS, nous reportons à la figure 13.12 le temps nécessaire pour classer les séquences sur chaque batch correspondant à des données d'usage sur le site Web de l'Inria. Les données ont été collectées sur une période de 14 mois et représentent 14 Go. Le nombre total de navigations est de 3,5 millions pour 300000 navigations. Nous avons découpé le fichier log en batches de 4500 transactions (soit environ 1500 séquences en moyenne). Nous pouvons observer que les temps de réponse sont assez semblables (ils varient de 1200 ms à 2200 ms) avec un avantage tangible pour SCDS. En effet, le temps moyen d'exécution d'ICDS est supérieur de 5% à celui de SCDS. Cela peut s'expliquer de deux manières :

1. Au début du traitement d'un batch B_n, soit dès le traitement de s_1 (la première séquence de B_n) il faut parcourir un nombre déjà grand de clusters existants (les

clusters hérités du batch B_{n-1}) afin de trouver celui qui pourrait correspondre à s_1. Alors que dans SCDS, l'ensemble des clusters est vide à l'initialisation d'un batch, donc s_1 n'est comparée à aucun autre cluster et un premier cluster est créé.

2. A la fin du traitement du batch il faut recalculer les alignements pour ne plus tenir compte des anciennes séquences de chaque cluster.

Nous avons ajouté à la figure 13.12 le nombre de séquences de chaque batch pour expliquer les différences de temps d'exécution d'un batch à un autre. Il s'agit d'un premier élément montrant qu'il n'est pas justifié de passer d'une méthode amnésique à une méthode incrémentale dans notre cas.

Analyse de la qualité des clusters

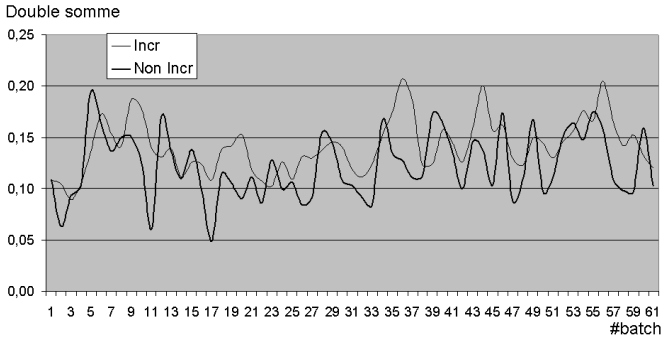


FIG. 13.13: Distance globale, batch par batch

Afin de mesurer la qualité des classes produites par ICDS, notre principal outil sera la distance entre deux séquences. Soit s_1 et s_2, deux séquences, la distance $dist(s_1, s_2)$ entre s_1 et s_2 est basée sur $sim(s_1, s_2)$, la mesure de similitude donnée par la définition 13.1 et telle que $dist(s_1, s_2) = 1 - sim(s_1, s_2)$. On a donc $dist(s_1, s_2) \in [0..1]$ et $dist(s_1, s_2)$ proche de 0 signifie que les séquences sont proches (similaires si cette valeur est nulle) alors que $dist(s_1, s_2)$ proche de 1 signifie que les séquences sont éloignées (ne partagent aucun item si cette valeur est 1). Nous reportons dans la figure 13.13 la double moyenne DBM après avoir traité chaque batch. Soit C l'ensemble des classes, DBM est calculée de la manière suivante :

$$DBM = \frac{\sum_{i \in C} \frac{\sum_{x \in C_i} dist(x, c_i)}{|C_i|}}{|C|}$$

Avec c_i le centre de C_i (la i^{eme} classe). Soit C_i la i^{eme} classe, le centre de C_i est une séquence c_i telle que :

$\forall s \in C_i, \sum_{x \in C_i} dist(s,x) \geq \sum_{y \in C_i} dist(c_i,y)$.

La valeur finale de DBM à la fin du batch est donnée par la figure 13.3. On peut y observer que DBM est comprise entre 5% et 18%. A la fin du processus, la valeur moyenne de DBM est de 14% (une qualité moyenne des classes de 86%). Pour SCDS cette moyenne est de 12% (soit une qualité de 88%). Une fois de plus, il s'agit d'un élément en faveur de SCDS et ne justifiant pas le passage à l'incrémental.

Nous avons également implémenté un clustering hiérarchique sur les séquences de chaque batch. Afin de compléter notre étude de la qualité des clusters obtenus avec ICDS, nous avons utilisé les mesures d'entropie et pureté, par comparaison avec les clusters obtenus par le clustering hiérarchique. L'entropie d'un cluster C de taille n_r est calculée selon la formule suivante : $E(C) = -\frac{1}{logq} \sum_{i=1}^{q} \frac{n_r^i}{n_r} log \frac{n_r^i}{n_r}$, où q est le nombre total de clusters et n_r^i est le nombre de séquences du i^{eme} cluster qui font partie du cluster C. L'entropie du clustering est ensuite donnée par la formule :

$Entropie = \sum_{r=1}^{k} \frac{n_r}{n} E(C_r)$, avec n le nombre total de séquences. On considère qu'une petite valeur d'entropie traduit un bon clustering (par rapport au clustering de référence).

La pureté d'un cluster est donnée par : $P(C_r) = \frac{1}{n_r} max(n_r^i)$ et la pureté du clustering est donnée par la formule suivante : $Purete = \sum_{r=1}^{k} \frac{n_r}{n} P(C_r)$. Une grande valeur de pureté traduit un bon clustering par rapport à un clustering de référence.

Les valeurs que nous avons obtenues pour l'entropie et la pureté sur les 10 premiers batches sont données dans le tableau suivant :

Batch	Entropie SCDS	Entropie ICDS	Pureté SCDS	Pureté ICDS
1	0,012523916	0.012523916	0,95107037	0.95107037
2	0,013794152	0.015740575	0,94326377	0.9367343
3	0,017435603	0.017890325	0,93279576	0.9295173
4	0,015584618	0.014704098	0,93603873	0.9371327
5	0,016993482	0.016103497	0,93243974	0.9366611
6	0,018524481	0.018878395	0,92739403	0.9258607
7	0,0202273	0.021773964	0,9281229	0.92114794
8	0,015651526	0.017257141	0,9380881	0.93306845
9	0,017332898	0.015581713	0,93359625	0.9383797
10	0,01842611	0.016277248	0,93085754	0.93546456

Lors de ces expérimentations, pour SCDS la valeur moyenne de l'entropie a été de 0.0166 et la valeur moyenne de la pureté a été de 0.9353. Pour ICDS la valeur moyenne de l'entropie a été de 0,0166 et la valeur moyenne de la pureté a été de 0,9345. On constate donc un changement (non significatif) sur la pureté. Cette troisième évaluation de nos résultats confirme les limites d'une approche incrémentale pour le clustering des données d'un flux.
Un premier avantage de la méthode ICDS est qu'elle masque le découpage rigide du flux en batches ; en gardant les centres des clusters d'un batch à l'autre on permet la détection des clusters dont les séquences se retrouvent séparées dans des batches différent lors du coupage en batches.
Un deuxième avantage est au niveau de la vitesse de classification : en ayant déjà les centres des batches précédents et comme beaucoup des comportements fréquents (clusters) se répètent dans le temps, les séquences sont classifiées plus rapidement. Ce gain en vitesse de classification est quand même équilibré par un plus grand nombre de comparaisons et recalculs d'alignements.
Le choix entre ICDS et SCDS n'est pas facile, parce que d'un côté garder de l'information d'un batch à l'autre permet d'avancer plus vite dans la classification des séquences, mais d'un autre côté, cela réduit la sensibilité de l'algorithme face aux nouveaux changements et cela peut être pénalisant.

Ce troisième algorithme de clustering, ainsi que les éléments présentés dans les chapitres 9 et 10, a donné lieu à la publication suivante :

- MARASCU, A. & MASSEGLIA, F. (2007). Limites d'une approche incrémentale pour la segmentation de séquences dans les flux. In *Atelier Flux de Données*, Namur, Belgique, p. 49-60.

13.3 Discussion

Dans ce chapitre nous avons proposé trois méthodes de clustering :

- Un clustering qui compare la nouvelle séquence à classer avec toutes les séquences de tous les clusters ;
- Un clustering qui compare la nouvelle séquence à classer avec tous les centres des clusters existants a l'instant t et met a jour le centre du cluster ;

– Un clustering qui conserve les centres d'un batch à l'autre, afin de comparer les séquences du nouveau batch aux centres précédents.

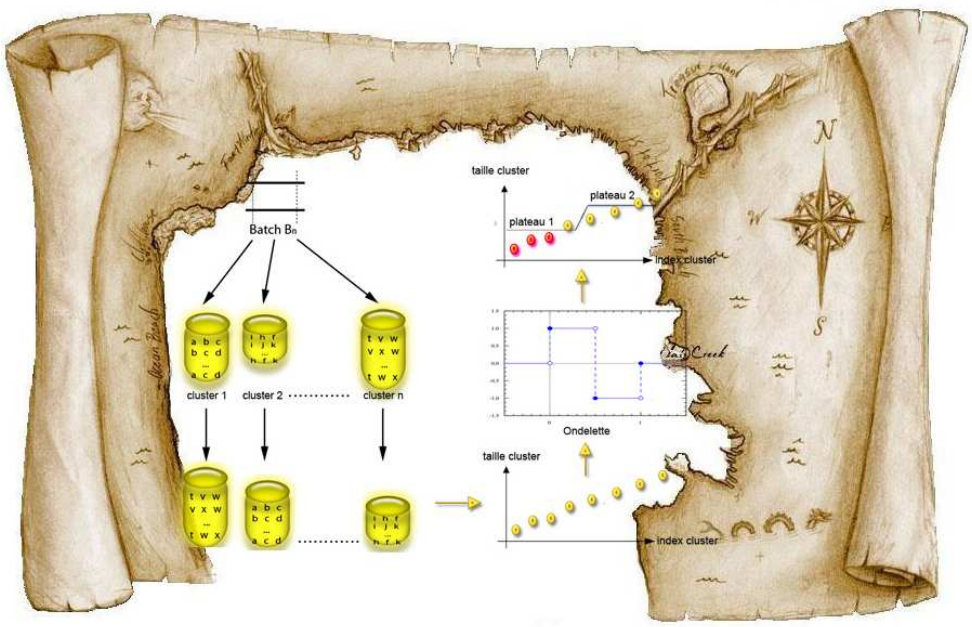
Les méthodes de clustering proposées représentent la première étape de notre méthode d'extraction de motifs séquentiels approximatifs dans les flux de données, mais ces méthodes peuvent être utilisées indépendamment afin de classifier les séquences d'un flux. Plusieurs types de méthodes ont été proposées afin d'extraire de l'information des flux de données (voir la section 9.2), mais une méthode commençant avec un clustering présente le bénéfice de pouvoir adapter facilement les critères qui séparent les données selon le domaine de l'application. La justesse des résultats des méthodes de clustering réside dans la précision des processus de comparaison et de mise à jour des centres des clusters. Pour calculer les centres des clusters nous avons utilisé une méthode basée sur un alignement de séquences Kum (2004). Lors de l'exécution, selon l'ordre des séquences à classer, la qualité des centres se détériorent. Afin de résoudre ce problème, j'ai proposé une structure nouvelle composée par un tableau des distances et une matrice de comptage.

Notre méthode est capable de détecter des comportements partagés par un nombre relativement faible d'utilisateurs (e.g. 13 utilisateurs ou encore 0.5%) ce qui est proche du problème de l'extraction de motifs séquentiels avec un support très faible. Nous avons prouvé que ces méthodes respectent les contraintes de vitesse des flux et qu'elles peuvent être intégrées dans un contexte temps réel. En effet, les temps de réponse de SCDS montrent que ce dernier est bien plus performant que SMDS et relativement meilleur que ICDS. Nous avons également montré que les classes proposées ont une très bonne qualité, ce qui permet d'extraire les motifs les plus pertinents et de manière exhaustive. Une comparaison des résultats de SCDS avec un clustering hiérarchique a été conduite du point de vue des temps d'exécution, mais également sur le plan de la qualité des résultats. Des calculs de l'entropie et de la pureté des clusters ont été fait afin de confirmer la qualité des résultats. De plus, nos résultats ont été obtenus pour des temps d'exécution largement inférieurs. Concernant la méthode incrémentale ICDS, les expérimentations ont montré qu'une méthode incrémentale est un peu complexe à l'exécution comparée à une méthode non incrémentale. De plus, cette complexité n'est pas forcement justifiée par de meilleurs résultats. Toutefois, la méthode ICDS a l'avantage de bien masquer le découpage inflexible en batches et de permettre la

détection des clusters répandus dans plusieurs batches voisins.

On a vu notre proposition en matière d'extraction de motifs séquentiels approximatifs dans les flux de données. Dans le chapitre suivant, on passe au deuxième sujet traité lors de cette thèse : la détection d'anomalies dans les flux de données.

CARTE GENERALE



Dans le prochain chapitre, on propose une méthode de détection d'anomalies. Ce sujet est connexe à notre sujet d'intérêt principal. Sur les clusters obtenus dans l'étape de clustering, on applique une méthode originale basée sur les ondelettes ; cette méthode distingue les anomalies des données normales.

CHAPITRE 14

DETECTION D'ANOMALIES

14.1 Motivation

SOMME ON A VU dans la section 5.2, les résultats d'une méthode de détection d'anomalies se présentent sous la forme d'un ensemble de données estampillées (soit par une étiquette, soit par un score). Si pour les données étiquetées, il suffit de lire l'étiquette pour connaître les anomalies (mais avec le désavantage qui est présenté dans la section 5.2), par contre pour les données attachées à un score on a une étape supplémentaire à franchir : établir le bon seuil qui dissocie les anomalies des éléments normaux.

À notre connaissance, le calcul de la valeur du seuil repose toujours sur le choix d'un paramètre [Jin *et al.* (2001); Joshua Oldmeadow *et al.* (2004); Portnoy *et al.* (2001); Zhong *et al.* (2007)], tel qu'un pourcentage, un top-n d'anomalies etc.

Mais dans le contexte d'un ***environnement de flux*** les données sont générées à une grande vitesse qui interdisent toute opération bloquante. En conséquence, demander un paramètre tel que k, pour les top-k anomalies, ou x comme pourcentage des éléments en queue de distribution qui seront considérés comme des anomalies, doit être évité. Premièrement, parce que l'utilisateur n'a pas assez de temps pour tester

plusieurs paramètres. Deuxièmement, parce qu'une valeur choisie à un instant t dans le flux sera probablement inadaptée au temps $t + n$. En effet sur le flux, d'une fenêtre d'observations à l'autre, les résultats évoluent et leur distribution change, ainsi que le nombre ou pourcentage d'anomalies. Pour ces raisons, la détection d'anomalies ne devrait pas dépendre d'un paramètre mais devrait être adaptative, dans le but de garder la meilleure précision tout au long de l'exécution du flux.
Dans ces conditions, une méthode de détection des événements atypiques ***ne doit pas uniquement dépendre d'un paramètre***. D'un autre côté, une méthode sans paramètre doit garantir de bons résultats en séparant de manière pertinente les anomalies du reste des données. Elle doit également ***être auto-adaptative*** et s'ajuster en fonction des changements de distribution (exponentielle, logarithmique, linéaire, etc.). Enfin, elle doit s'adapter à n'importe quel type de données.

Ce chapitre propose une méthode de détection d'anomalies qui répond à toutes les demandes énoncées au-dessus et parcourt successivement les points suivants :

- Description générale du principe utilisé par notre méthode
- Présentation de la théorie des ondelettes
- Avantages de l'utilisation des ondelettes
- Expérimentations réalisées

14.2 Contribution : Méthode automatique de détection d'anomalies

14.2.1 Idée générale

Comme décrit dans le chapitre 7, on a choisi un traitement du flux de données par paquets et le traitement du paquet courant commence avec une étape de clustering (chapitre 13). A la fin de cette étape on a les données groupées dans des clusters selon un critère de similitude (la plus longue sous-séquence commune). Cette distribution des données est propice à la recherche des anomalies.
Nous proposons la méthode DOO (Détection d'Outliers par les Ondelettes) ; cette méthode fait partie des méthodes de détection d'anomalies basées sur un algorithme de clustering et peut s'appliquer à tout type de données tant qu'il y a un attribut qui établit un ordre croissant sur les éléments observés. C'est une méthode sans paramètre destinée

à l'extraction automatique d'anomalies dans des résultats ordonnés par un critère. La différence avec les travaux existants réside dans la technique de séparation des valeurs en deux ensembles, l'un correspondant aux segments (clusters) et l'autre aux anomalies. Nous considérons que nos clusters sont aussi denses que possible, grâce à notre algorithme de clustering, et nous voulons extraire les anomalies en utilisant le critère de la taille (en faisant l'hypothèse que les événements atypiques sont moins nombreux que les événements normaux). Le problème consiste donc à séparer les clusters, en fonction de leur taille, dans deux catégories : les normaux et les anomalies. Après les avoir triés par taille croissante, notre solution se base sur une analyse de la distribution de ces clusters. Une distribution classique est illustrée par la figure 14.1 (capture d'écran réalisée sur nos données). Afin de trouver la meilleure séparation, notre idée est d'utiliser la transformée en ondelettes de cette distribution. Grâce à une connaissance *a priori* du nombre de plateaux (on veut deux plateaux, l'un pour les petits segments, ou anomalies, et l'autre pour les segments normaux) nous pouvons couper cette distribution d'une manière efficace. Dans la figure 14.1, la taille des clusters est reportée en y et leur indice dans la distribution est reporté en x. Les deux plateaux correspondent à la séparation entre les anomalies et les clusters normaux. En effet, chaque cluster dont la taille est inférieure ou égale à la valeur du premier plateau sera considéré comme une anomalie.

14.2.2 Ondelettes

La transformée en ondelettes est un outil qui permet de décomposer des données, ou des fonctions, ou des opérateurs, en différentes composantes de fréquence et ensuite d'étudier chaque composante avec une résolution adaptée à son échelle [Daubechies (1992)]. En d'autres termes, la théorie des ondelettes permet de représenter une série de valeurs et la décomposer en plusieurs parties reliées ; quand ces parties sont mises à l'échelle et translatées, cette décomposition est appelée transformation en ondelettes. La reconstruction en ondelettes, ou transformée en ondelettes inverse, implique de remettre les différentes parties ensemble et de retrouver l'objet d'origine [Young (1995)]. Du point de vue mathématique, la transformée en ondelettes continue est définie par :

$$T^{wav} f(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} f(x)\psi^*(\frac{x-b}{a})dx$$

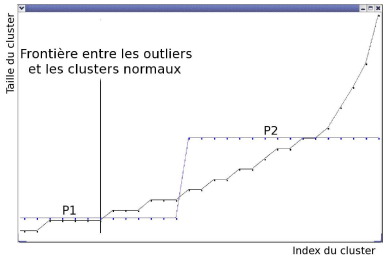


FIG. 14.1: Détection d'anomalies par les ondelettes de Haar. L'ordonnée donne la taille des clusters et l'abscisse l'indice dans la distribution. Les deux plateaux correspondent à la séparation entre les anomalies et les clusters normaux.

où z^* dénote le nombre complexe conjugué de z, $\psi^*(x)$ est l'ondelette, a (> 0) est le facteur de mise à l'échelle et b est le paramètre de translation. Cette transformation est linéaire et covariante en translation et dilatation. Cette expression peut être également interprétée comme une projection du signal sur une famille de fonctions analysant $\psi_{a,b}$, construite à partir d'une ondelette mère et selon l'équation suivante : $\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a})$. Les ondelettes sont une famille de fonctions localisées en temps et en fréquence et sont obtenues par translations et dilatations à partir d'une fonction $\psi(t)$, dite ondelette mère. Pour des choix particuliers de a, b et ψ, $\psi_{a,b}$ est une base orthonormale dans $L^2(\mathbb{R})$. Tout signal peut être décomposé en le projetant sur sa fonction d'ondelette. Pour comprendre le mécanisme de la transformée en ondelettes, il faut comprendre l'analyse multi-résolution (AMR). Une analyse multi-résolution de l'espace $L^2(\mathbb{R})$ est une séquence de sous-espaces imbriqués tels que :

$... \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} ... \subset V_{j+1} \subset V_j ...$

$$\overline{\bigcup_{j \in Z} V_j} = L^2(\mathbb{R})$$
$$\bigcap_{j \in Z} V_j = \{0\}$$

$\forall j \in \mathbb{Z}$ if $f(x) \in V_j \iff f(2^{-1}x) \in V_{j+1}$ (or $f(2^j x) \in V_0$)

$\forall k \in \mathbb{Z}$ if $f(x) \in V_0 \iff f(x-k) \in V_0$

Il existe une fonction $\varphi(x) \in L^2(\mathbb{R})$, appellée fonction de mise l'échelle qui, par dilatation et translation, génère une base orthonormale de V_j. Les fonctions sont construites selon la relation suivante : $\varphi_{j,n}(x) = 2^{-\frac{j}{2}}\varphi(2^{-j}x - n), n \in \mathbb{Z}$, et la base est orthonormée si $\int_{-\infty}^{+\infty} \varphi(x)\varphi^*(x+n)dx = \delta(n), n \in \mathbb{Z}$. Pour chaque V_j, son complément orthogonal

W_j dans V_{j-1} peut être défini comme suit : $V_{j-1} = V_j \oplus W_j$ et $L^2(\mathbb{R}) = \bigoplus_{j\in\mathbb{Z}} W_j$. Comme W_j est orthogonal à V_{j-1}, alors W_{j-1} est orthogonal à W_j, donc $\forall j, k \neq j$ on a $W_j \perp W_k$.

Il existe une fonction $\psi(x) \in \mathbb{R}$, appelée ondelette qui, par dilatations et translations, génère une base orthonormale de W_j, et donc de $L^2(\mathbb{R})$. Les fonctions sont construites comme suit : $\psi_{j,n}(x) = 2^{-\frac{j}{2}}\psi(2^{-j}x - n), n \in \mathbb{Z}$. Donc, $L^2(\mathbb{R})$ est décomposé en une séquence infinie d'espaces d'ondelettes, *i.e.* $L^2(\mathbb{R}) = \bigoplus_{j\in Z} W_j$. Pour résumer la décomposition en ondelettes : soit une fonction f_n dans V_n, f_n est décomposée en deux parties, une partie dans V_{n-1} et l'autre dans W_{n-1}. À l'étape suivante, la partie V_{n-1} est encore décomposée en deux parties, une partie dans V_{n-2}, l'autre dans W_{n-2} et ainsi de suite...

Une application directe de l'AMR est la transformée rapide discrète en ondelettes (Discrete Wavelet Transform). L'idée est d'aplanir les données de manière itérative et de garder les détails séparément. Des preuves plus formelles sur les ondelettes peuvent être trouvées dans [Daubechies (1992)]. Les ondelettes sont généralement utilisées pour résumer des données ou trouver une tendance dans des fonctions numériques. En pratique, la majorité des coefficients d'ondelettes sont petits ou insignifiants. Ainsi, pour représenter une tendance, seul un petit nombre de coefficients significatifs est nécessaire. Nous utilisons les ondelettes de Haar dans notre méthode de détection d'anomalies. Considérons la série de valeurs suivante : $(1, 1, 2, 5, 9, 10, 13, 15)$. Pour calculer les coefficients de l'ondelette de Haar, on groupe les éléments deux par deux et pour chaque groupe on calcule la valeur moyenne et la différence entre le premier élément et cette moyenne. Sa transformée en ondelettes de Haar est illustrée dans la table suivante :

Niveau	Approximations	Coefficients
8	1, 1, 2, 5, 9, 10, 13, 15	
4	1, 3.5, 9.5, 14	0, -1.5, -0.5, -1
2	2.25, 11.75	-1.25, -2.25
1	7	-4.75

On garde alors les deux coefficients les plus significatifs (voir le théorème 14.1) et les autres sont mis à zéro. Dans notre série de coefficients $(7, -4,75, -1.25, -2.25, 0, -1.5, -0.5, -1)$ les deux plus significatifs sont 7 et -4.75, afin que la série devient égale à [7, -4.75, 0, 0, 0, 0, 0, 0]. Dans les étapes suivantes, la transformée inverse est calculée et nous obtenons une approximation des données d'origine (2.25, 2.25, 2.25, 2.25, 11.75,

11.75, 11.75, 11.75). Cette approximation donne deux plateaux, correspondants aux valeurs $\{1, 1, 2, 5\}$ et $\{9, 10, 13, 15\}$. Dans notre cas, l'ensemble des anomalies contient les clusters dont la taille est inférieure au premier plateau (*i.e.* 2.25). Dans notre exemple, $o = \{1, 1, 2\}$ donne la taille des anomalies (*i.e.* les clusters dont la taille est inférieure ou égale à 2).

14.2.3 Avantages de l'utilisation des ondelettes

Plus généralement, les avantages de cette méthode pour notre problème sont illustrés à la figure 14.2. Selon la distribution, la transformée en ondelettes donne différents indices de coupure. Par exemple, dans nos données d'usage du site de l'Inria Sophia-Antipolis il y a une variation de la distribution entre le jour et la nuit sur certains scripts PHP. Cette variation aboutit à deux formes principales. La figure 14.2 donne une illustration de deux distributions différentes, similaires à celle retrouvées dans nos expérimentations. Considérons qu'un filtre de 10% soit appliqué à cette distribution pour en extraire les anomalies. Si ce filtre obtient des résultats corrects sur la première distribution (partie gauche correspondant aux requêtes vers 01h00) il est en revanche inadapté à la distribution de droite (usages vers 17h00) et renvoie des clusters qui ne devraient pas être considérés comme anomalies. Notre principe de détection d'anomalies à base d'ondelettes, par contre, est auto-adaptatif et s'ajustera pour prendre en compte la nouvelle forme de la distribution en augmentant très peu le niveau de sélection des anomalies. L'application de la transformée en ondelettes

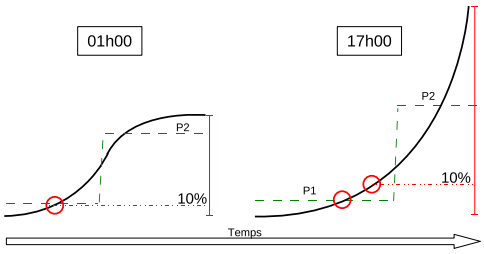


FIG. 14.2: Une distribution des tailles des clusters qui varie avec le temps

sur des séries de valeurs permet d'obtenir une bonne compression et, en même temps, selon différentes tendances, une bonne séparation. Sachant que les anomalies sont des objets rares, elles seront toujours groupés dans les petits clusters (ou resteront isolés).

Le principe de DOO, pour séparer les anomalies des clusters normaux, est basé sur le théorème 14.1.

Proposition 14.1 *Soit F, l'ensemble des caractéristiques utilisées pour construire la distribution D sur les résultats de l'algorithme de clustering. Soit P1 et P2 les deux plateaux obtenus après avoir sélectionné les deux coefficients les plus significatifs de la transformée en ondelettes de D. La séparation optimale en deux groupes, selon F et respectant la minimisation de la somme des erreurs carrées, est données par P1 et P2.*

Preuve Dans une base orthonormale, il est montré que les k coefficients d'ondelette les plus grands donnent la meilleure k-approximation de Haar du signal d'origine, selon la minimisation de la somme des erreurs carrés pour un k fixé [Stollnitz *et al.* (1995)]. Pour cela, considérons le signal d'origine $f(x)$ et l'ensemble de fonctions u_1, $...u_m$. Le signal peut alors être représenté selon la fonction suivante : $f(x) = \sum_{i=1}^{m} c_i u_i(x)$

Le but est de trouver un ensemble réduit de fonctions u_i. Soit σ une permutation de $1, ..., m$ et f' la fonction d'approximation utilisant les premiers m' éléments de σ, avec $m' < m$. $f'(x) = \sum_{i=1}^{m'} c_{\sigma(i)} u_{\sigma(i)}(x)$

La somme des erreurs carrés de L^2 de cette approximation est :

$$\begin{aligned}
\left|\left|f(x) - f'(x)\right|\right|_2^2 &=< f(x) - f'(x) | f(x) - f'(x) > \\
&= \left\langle \sum_{i=m'+1}^{m} c_{\sigma(i)} u_{\sigma(i)} \Big| \sum_{j=m'+1}^{m} c_{\sigma(j)} u_{\sigma(j)} \right\rangle \\
&= \sum_{i=m'+1}^{m} \sum_{j=m'+1}^{m} c_{\sigma(i)} c_{\sigma(j)} < u_{\sigma(i)} | u_{\sigma(j)} > \\
&= \sum_{i=m'+1}^{m} (c_{\sigma(i)})^2
\end{aligned}$$

Grâce à l'orthonormalité, $< u_i, u_j >= \delta_j^i$. Pour tout $m' < m$, alors minimiser cette erreur revient à choisir la permutation croissante σ (c'est la permutation où les éléments sont ordonnés par ordre croissant). Donc, pour $m' = 2$ on obtient la meilleure 2-approximation de Haar du signal d'origine.□

Sur la base du théorème 14.1, on retient les 2 coefficients les plus significatifs et on sélectionne les clusters dont la taille est inférieure ou égale à la valeur du premier plateau.

Ces clusters sont alors considérés comme des anomalies sans qu'aucun paramètre ne soit demandé à l'utilisateur.

14.2.4 Expérimentation

Nous avons expérimenté notre méthode sur un flux de données d'un site Web. Nous retenons uniquement les données liées aux scripts PHP dans la mesure où les comportements malveillants peuvent exploiter les failles de ces scripts. La détection d'attaques revient à la détection de comportements atypiques sur ces scripts. Pour chaque paquet du flux notre prétraitement consiste premièrement à ne retenir que les requêtes à des scripts PHP avec paramètre(s). Ensuite, pour chaque paramètre, nous construisons un objet correspondant au mot clé donné par l'utilisateur. Par exemple, avec la requête `auteur.php?Nom=Jean&Prnm=Dupont`, les objets seront o_1 =`Jean` and o_2 =`Dupont`. Ainsi, à la fin de l'étape de prétraitement, notre ensemble d'objets est constitué de tous les paramètres donnés aux scripts PHP du site. Un clustering est appliqué ensuite selon 13.2.2.

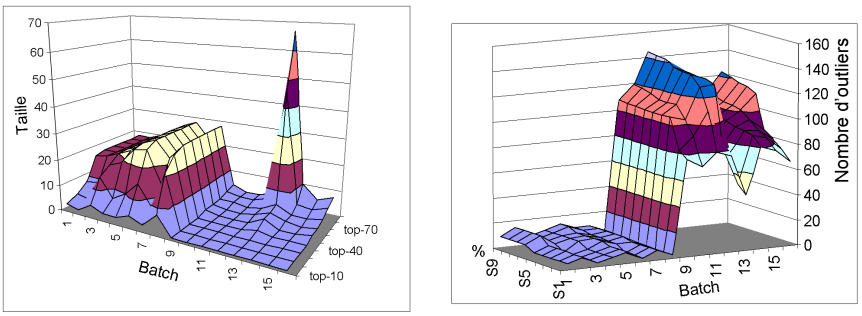


FIG. 14.3: Taille des anomalies avec un filtre top-k et nombre d'anomalies avec un filtre p%

Nos expérimentations ont été réalisées sur des données réelles, venant de log d'usage Web sur le site de l'Inria Sophia-Antipolis de janvier 2006 à avril 2007. Les données originales représentent 18 Go et correspondent à un total de 11 millions de navigations, regroupées par paquets de 8500 requêtes (en moyenne). Dans cette section nous montrons uniquement les résultats sur 16 paquets bien représentatifs des résultats globaux

Paquet	1	2	3	4	5	6	7	8
Objets	1425	1365	1805	1600	1810	2115	1855	2930
Clusters	44	42	35	37	34	34	41	37
Paquet	9	10	11	12	13	14	15	16
Objets	1980	2225	2175	2470	3645	2730	4040	4105
Clusters	109	124	148	133	84	99	120	134

TAB. 14.1: Paquets, objets et clusters

et qui illustrent bien les changements de distribution. Les 8 premiers paquets sont sélectionnés sur les requêtes PHP exécutées entre 01h00 et 02h00. Les 8 paquets suivants, sur les requêtes PHP exécutées entre 15h et 16h.

La figure 14.3 montre les résultats obtenus par un top-k et un p% sur ces 16 paquets. Pour chaque paquet, les nombres d'objets et de clusters sont donnés par la table 14.1. La première surface de la figure 14.3 (gauche) donne la taille des clusters sélectionnés par un filtre top-k. Le principe est de sélectionner les k derniers clusters après les avoir triés par taille décroissante. Un désavantage évident est de sélectionner soit trop, soit pas assez, de clusters. Considérons, par exemple, le paquet 13 de la figure 14.3 (gauche). Avec $k = 50$, la taille maximum d'une anomalie est de 4, alors qu'avec $k = 90$, cette taille est de 67 (ce qui est la taille maximum d'un cluster dans ce paquet qui n'en contient que 84).

Nous avons également implémenté un filtre basé sur la valeur p du pourcentage de clusters considérés comme anomalies dans la distribution. Le nombre d'anomalies sélectionnées par ce filtre, selon différentes valeurs de p (*i.e.* de 0.01 à 0.09), est donné dans la figure de droite 14.3. Le principe est de considérer $p \in [0..1]$, un pourcentage donné par l'utilisateur, $d = maxVal - minVal$ l'intervalle des tailles de clusters et $y = (p \times d) + minVal$. Ensuite, le filtre sélectionne les clusters de taille t tels que $t \leq y$. Par exemple, avec $s = \{1, 3, 10, 11, 15, 20, 55, 100\}$, une distribution de tailles et $p = 0.1$, on obtient $d = 100 - 1 = 99$, $y = 1 + (0.1 \times 99) = 10$ et l'ensemble des anomalies sera $o = \{1, 3, 10\}$. Dans nos expérimentations, ce filtre est généralement plus résistant aux variations de distribution que le filtre top-k. On peut noter des résultats plus homogènes dans la figure 14.3 (absence de "pics"). Par exemple, avec le paquet 13, on note que ce filtre extrait des anomalies de taille 44 (1%) à 78 (9%), ce qui correspond aux résultats

des filtres top-40 à top-70.

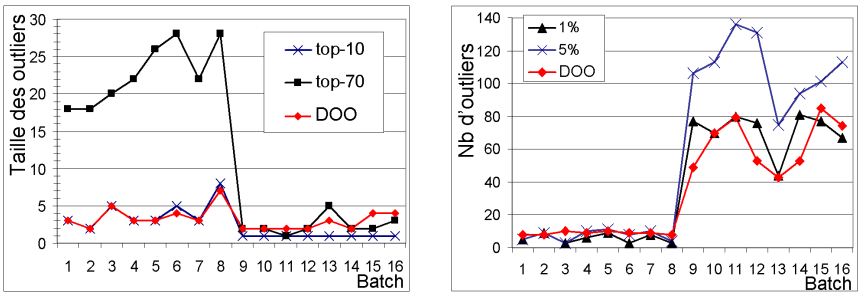


FIG. 14.4: Comparaison entre DOO et les filtres top-k et p%

La figure 14.4 donne une comparaison de DOO (sur les même paquets) avec les filtres top-k et p%. La partie gauche montre les résultats de DOO, d'un top-10 et d'un top-70. Sur les 8 premiers paquets, DOO et le top-10 donnent les meilleurs résultats. Cependant, pour les paquets 9 à 16, le top-10 donne des résultats trop petits (la taille maximale est de 1). Sur les paquets 9 à 16, les meilleurs résultats sont donnés par DOO et le top-70. En revanche, le top-70 donne des résultats bien trop élevé sur les 8 premiers paquets. En conclusion, aucune valeur de k ne peut servir de référence tout le long du flux et l'utilisateur devra modifier k en fonction des changements de distribution d'un paquet à l'autre. Ces résultats font du top-k une méthode difficile à utiliser dans le contexte des flux. D'un autre côté, les résultats de DOO montrent sa capacité à s'adapter d'une distribution à l'autre et à sélectionner automatiquement la bonne taille des événements atypiques.

La partie droite de la figure 14.4, compare les résultats de DOO avec ceux des filtres 1% et 5%. Nous observons que DOO et le filtre 1% donnent des résultats similaires. Par exemple sur le paquet 7, DOO obtient 9 anomalies et le filtre 1% en obtient 8. Cependant, la plupart des valeurs du filtre 1% sont basses sur les 8 premiers paquets. Sur ces paquets, le filtre 5% obtient de meilleurs résultats, mais il n'est plus du tout adapté aux paquets 9 à 16 (jusqu'à 138 anomalies contre au plus 84 pour DOO). Cela est dû à la variation de distribution, comme illustré par la figure 14.2. L'avantage de DOO sur le filtre p% est donc double :

1. DOO n'a pas besoin de paramètre. Cette méthode s'ajuste automatiquement à toute forme de distribution et tout nombre d'objets et de clusters. En revanche,

l'utilisateur doit essayer plusieurs valeurs de pourcentages avant de trouver le bon intervalle (par exemple 5% pour les premiers paquets). Les anomalies données par DOO resteront valides après un changement de distribution.

2. Selon le théorème 14.1, DOO donne une séparation optimale entre les clusters et les anomalies. Considérons l'exemple précédent avec la distribution $s = \{1, 3, 10, 11, 15, 20, 55, 100\}$. Avec cette distribution, le filtre 10% donnera l'ensemble d'anomalies $o = \{1, 3, 10\}$. Pourquoi ne pas inclure 11 dans o? En effet, 10 et 11 sont des valeurs très proches. D'un autre côté, DOO donnera l'ensemble $o = \{1, 3\}$, ce qui est un résultat réaliste et naturel.

14.3 Discussion

Tant qu'on collectera des informations intéressantes, on devra les protéger des personnes désirant les posséder. Tant qu'on sera des êtres humains et non des robots, on devra se prévenir des erreurs de saisie, de mesure, de calcul etc. Il s'agit là des deux grands défis liés à la présence des anomalies dans les données. On a vu dans la section 5.1 le grand besoin de les séparer du reste des données. Mais tracer une ligne séparatrice entre les anomalies et les données normales n'est pas toujours une tâche facile, surtout si la décision doit être rapide malgré les calculs complexes nécessaires sur ces grandes masses de données qui évoluent dans le temps, comme c'est le cas des flux de données. Ces contraintes temporelles et quantitatives nécessitent des prises de décisions rapides et, en conséquence, mènent à l'élimination de tout élément ralentissant le processus de décision. Un exemple d'éléments ralentissants sont les interventions répétées de l'utilisateur afin de spécifier différents paramètres. Nous avons proposé *une technique de séparation qui élimine tout paramétrage à demander à l'utilisateur*. Ainsi, le processus assure l'*automatisation des calculs* et le rend intégrable dans un environnement dynamique où l'utilisateur n'a pas le temps de spécifier des paramètres au fil de l'eau. En plus, la complexité du calcul du point de coupure pour une série de n valeurs n'est pas élevée, elle est de $O(n^2)$. Notre méthode *s'auto-adapte* aux données et trouve le point parfait de coupure *sans intervention de l'utilisateur*. Nous avons illustré notre méthode sur l'algorithme de clustering présenté dans la section 13.2.2, mais nous aurions pu utiliser toute autre méthode de clustering, comme, par exemple, les k-means ou les méthodes neuronales. De plus, elle peut être appliquée

sur les résultats de tout type de clustering et toutes les caractéristiques peuvent être utilisées (distances entre objets [Knorr & Ng (1998c)], densité [Breunig *et al.* (2000a); Papadimitriou *et al.* (2003)] ou taille des clusters[Jaing *et al.* (2001); Sequeira & Zaki (2002))]. Notre objectif n'est pas de proposer une méthode de détection d'intrusion, dans la mesure où les anomalies ne sont pas forcément des intrusions. Toutefois, nous voulons proposer des résultats potentiellement utiles dans une telle démarche. Ces travaux ne sont pas une solution universelle, mais constituent une brique solide pour répondre au problème d'identification des intrusions.

Les travaux concernant cette méthode de détection d'anomalies dans les flux de données a donné lieu à plusieurs publications :

- SINGH, G. & MASSEGLIA, F. & FIOT, C. & MARASCU, A. & PONCELET, P. (2009). Collaborative Outlier Mining for Intrusion Detection. In *Extraction et gestion des connaissances*, Strasbourg, France, pp 313-324. (Nominé pour le prix du meilleur papier applicatif).
- MARASCU, A. & MASSEGLIA, F. (2009). Détection d'enregistrements atypiques dans un flot de données : une approche multi-résolution. In *Extraction et gestion des connaissances*, Strasbourg, France, pp 455-456.
- MARASCU, A. & MASSEGLIA, F. (2009). Parameterless Outlier Detection in Data Streams. In *24th Annual ACM Symposium on Applied Computing (ACM SAC'09)*. Honolulu, Hawaii, USA.
- MARASCU, A. & MASSEGLIA, F. (2009). Atypicity Detection in Data Streams : a Self-Adjusting Approach. In *Intelligent Data Analysis Journal (IDA)*. A paraître.

Avec ce chapitre, on finit la présentation de la deuxième partie de ce mémoire. Dans cette partie nous avons présenté les solutions apportées concernant deux grands sujets : l'extraction de motifs séquentiels approximatifs dans les flux de données et la détection d'anomalies dans les flux de données. La partie suivante présente les conclusions, une exploitation dans le monde réel et les nombreuses perspectives ouvertes par ce travail.

Troisième partie

SORTIE DU LABYRINTHE

Dans cette partie nous présentons nos conclusions aux travaux développés dans ce mémoire suivie par une description d'une exploitation dans le monde réel et, enfin, les nombreuses perspectives ouvertes.

CHAPITRE 15

CONCLUSIONS

CONCLUONS ce mémoire avec un résumé de notre travail de recherche et de nos principales contributions.

Dans ce mémoire nous avons proposé *une méthode d'extraction de motifs séquentiels approximatifs dans les flux de données et une méthode non-paramétrique de détection d'anomalies*. Nous avons prouvé l'efficacité de nos méthodes par des évaluations (tant théoriques que pratiques) tout au long du mémoire.

Notre méthode d'extraction de motifs séquentiels approximatifs traite le flux par batches et commence avec un clustering appliqué aux séquences du batch courant. Notre première contribution consiste en **trois méthodes de clustering** validées par des expérimentations. Deux de nos méthodes de clustering utilisent un représentant par cluster (un centroïde). Afin d'assurer la mise à jour des centroïdes tout au long de l'exécution, nous avons proposé et mis en place **une nouvelle structure** capable de gérer ce problème et qui propose automatiquement le recalcul des centroïdes en cas de détérioration.

Les centroïdes des clusters représentent les motifs séquentiels approximatifs et sont gardés pendant l'exécution dans **une structure de données** fiable et efficace. Cette structure permet un stockage optimal des motifs séquentiels fréquents tout en ayant les avantages suivants : *s'autolimite à la quantité de mémoire disponible, permet d'être interrogée en temps réel, permet des opérations de mises à jour et des interrogations dans un temps très faible et garde le comportement général des motifs séquentiels fréquents dans le temps*. Les motifs sont retenus dans une structure d'arbre préfixé, alors que leur historique est géré par des **composants ATGT**. Les composants ATGT que nous avons mis en place ont de multiples avantages :

- *détection rapide de la meilleure compression à faire (en termes d'erreur globale et locale)* ;
- *utilisation d'une nouvelle technique rapide et efficace de compression de segments (AMi)* ;
- *minimisation de l'erreur de réponse à des interrogations sur tout l'historique* (et pas seulement pour les événements récents, comme la majorité des méthodes le proposent) ;
- *limitation à une quantité de mémoire fixée au départ* (caractéristique très importante, car ainsi il suffit de spécifier nos capacités en termes de mémoire et l'algorithme trouvera tout seul la meilleure qualité des résultats possible) ;
- *garantie des meilleurs résultats approximatifs par rapport aux vrais comportements (caractéristique assurée par REGLO)* ;
- *précision des résultats* (spécification de l'erreur avec laquelle un résultat est donné) ;
- *vitesse de calcul* (ce qui le rend très efficace et adaptable au contexte des flux).

Afin de détecter et compresser deux segments contenus par une courbe représentant le comportement d'un motif séquentiel dans le temps, nous avons proposé **deux méthodes : une basée sur des formules simplifiées de la régression linaire et l'autre basée sur une technique innovante (AMi)**. Nous avons également proposé des formules de calcul et de mise à jour de l'erreur, ce qui permet à l'utilisateur de savoir l'erreur avec laquelle un résultat lui est présenté.

Une partie des avantages des composants ATGT est assurée grâce à une nouvelle stratégie de gestion de l'historique des motifs séquentiels fréquents (REGLO). En résumé, afin de décider le motif et l'endroit à compresser, cette stratégie prend en compte

à la fois l'erreur globale et les erreurs locales correspondantes aux composants ATGT existants en mémoire. Á notre connaissance, **REGLO est la première méthode qui optimise en parallèle l'erreur globale et locale et qui s'auto-adapte à un espace de mémoire fixe.**

Notre deuxième méthode proposée lors de cette thèse concerne la détection d'anomalies. Même si les anomalies sont un sujet de grand intérêt dans le domaine des flux, nous avons constaté que, à notre connaissance, toutes les méthodes de détection d'anomalies existantes avaient besoin d'un ou plusieurs paramètres, ce qui posait des problèmes d'intégration dans un flux. Pour répondre à ce problème, nous avons proposé une méthode qui, à notre connaissance, **est la première méthode de détection d'anomalies sans paramètre et auto-adaptative.**

Pour situer CLARA par rapport aux travaux du domaine, je propose le tableau illustré par la figure 15. Ce tableau reprend les principales caractéristiques de notre approche de manière comparative :

- *Extraction de motifs séquentiels à partir de séquence d'itemset.* Dans la littérature, nous avons trouvé peu de travaux permettant cette extraction dans les flux de données. Citons en particulier MDSDS Raïssi & Plantevit (2008), RESERVOIR Raïssi & Poncelet (2007) et MILE Chen *et al.* (2005). Notons que le passage des séquences d'items aux séquences d'itemsets représente une différence très importante en raison de l'explosion combinatoire liée à ces données.
- *Gestion de l'historique.* Notre approche permet de gérer l'historique des motifs avec un point de vue nouveau et dont l'efficacité sur le plan de l'approximation est démontrée dans nos travaux. L'ATGT retient les événements marquants avec plus de finesse (accorder cette finesse aux événements récents n'est pas toujours souhaitable).
- *Détection d'anomalies.* Dans notre approche, la détection d'anomalie se base sur une étape de clustering. Ainsi, une fois les séquences classifiées, il est possible d'obtenir les atypiques. Notons que Raïssi & Plantevit (2008) se base sur les travaux de Plantevit (2008) qui permettent la détection d'anomalies dans les données multidimensionnelles statiques. Une extension de Raïssi & Plantevit (2008) pour la détection d'anomalies dans les flux est donc certainement envisageable. En revanche, les anomalies sont généralement considérées comme des événements rares.

Dans ce sens, on peut penser (pour des raisons statistiques) que de tels événements risquent d'être ignorés par une méthode basée sur l'échantillonnage. La technique proposée par Raïssi & Poncelet (2007) serait certainement moins adaptée pour une telle détection.

– *Clustering.* A notre connaissance, CLARA est la seule approche permettant la classification non supervisée de séquences dans les flux de données.

Comme nous l'avons montré dans ce mémoire, les flux de données présentent des contraintes fortes. En raison de ces contraintes, les traitements classiques conçus pour les données statiques ne peuvent être appliqués aux flux et doivent être ré-inventés. Notre objectif avec ce travail était de présenter une approche la plus complète possible pour l'extraction de connaissance dans ces flux. Tout d'abord, l'extraction doit se faire avec la plus grande rapidité sur des "instantanés" qui correspondent à la fenêtre d'observation que l'on peut se permettre sur le flux. Notre technique d'approximation basée sur des alignements de motifs séquentiels permet cela. Mais se contenter d'extraire des connaissances sur une fenêtre au temps t n'est pas suffisant pour résumer le flux. Il faut également gérer l'historique de ces connaissances pour garder une trace de l'évolution du contenu du flux. Notre proposition d'une représentation de l'historique qui privilégie les événements importants répond à ce problème avec un point de vue plus efficace que les techniques basées sur le facteur de vieillissement. Enfin, la détection d'anomalie dans un flux impose des temps de réaction très rapides et interdit de solliciter l'utilisateur. Notre technique d'ondelette permet cette détection de manière automatique et efficace. Les principales difficultés liées au traitement et à l'extraction de connaissances dans un flux sont donc abordées dans cette thèse et, pour chacune, une solution est proposée et s'inscrit dans notre approche globale.

Parallèlement aux validations de notre méthode présentées tout au long du mémoire vient une validation avec Orange R&D et qui sera décrite en détail dans le chapitre 16.

Approche	Clustering	Technique d'extraction	Détection d'anomalies	Gestion de l'historique	Traitement temporel	Disponibilité des connaissances
SPEED		✓ PLSC		✓ Tilted Time W	Batches	✓ A chaque batch
RESERVOIR		✓ PrefixSpan			Sliding W	*Nécessite un appel à PrefixSpan*
MDSDS		✓ PrefixSpan	*Envisageable*	✓ Tilted Time W	Batches	✓ A chaque batch
MILE		✓ PrefixSpan			Sliding W	*En fin de flux*
CLARA	✓ 3 approches	✓ Alignement	✓ Ondelettes	✓ ATGT	Batches	✓ A chaque batch

FIG. 15.1: Etat de l'art - tableau comparatif

CHAPITRE 16

EXPLOITATION DANS LE MONDE RÉEL

"Le premier académicien que je vis était une maigre et piètre figure usée ; il avait un visage et des mains couleur de suie, la barbe et les cheveux longs, un habit et une chemise de même que sa peau. Il avait pâli huit ans sur un projet consistant à extraire des concombres des rayons du soleil"

Les Voyages de Gulliver, Jonathan Swift

Nous voulons proposer des travaux exploitables pour des applications réelles et nous espérons qu'ils seront plus convaincants que des travaux sur l'extraction des rayons du soleil à partir des concombres !

CE CHAPITRE contient la présentation de l'application de notre méthode sur des données d'Orange R&D.

Les travaux présentés dans ce mémoire font actuellement l'objet d'une adaptation dans le projet ANR MIDAS. Le projet MIDAS (Mining Data Streams) réunit des partenaires académiques (**Télécom ParisTech, LIRMM, INRIA, CEREGMIA**) et industriels (**EDF R&D, Orange R&D et le CHU de Fort de France**). L'objectif de ce projet est de proposer des techniques de résumé pour les flux de données et de développer des structures de données et des traitements génériques.

Cette section résume le projet d'exploitation de la méthode de clustering présentée en 13.2.2 sur des données proposées par Orange R&D. Ces données proviennent de l'utilisation du portail mobile Orange. Elles contiennent, pour chaque requête, l'identifiant de l'utilisateur, l'estampille de la requête, la page demandée (ainsi que les thèmes associés), la page précédente et des informations de session. La méthode proposée consiste à :

1. Accumuler les navigations sur une fenêtre de temps fixée par l'utilisateur ;
2. Pour chaque fenêtre, établir une segmentation de ces navigations en un nombre de clusters correspondant à une distance minimum fixée par l'utilisateur. Cette distance détermine le seuil à partir duquel deux navigations peuvent être regroupées.

16.1 Données en entrée

Les données proviennent des logs d'usage des utilisateurs du portail Orange mobile. Ces logs sont produits par les plates-formes de service sous la forme suivante :

- une table de logs au format TimeStamp ; Client.Id ; Session.Id ; Page.Id ;
- une table de correspondance Page.Id ; Theme.Id rattachant chaque page à un thème.

16.2 Résultats attendus

L'application produira une segmentation (ou clustering) qui consiste à regrouper les navigations données en entrée selon un degré de similitude fourni par l'utilisateur. Nous proposons de travailler sur des navigations qui seront filtrées sur les pages demandées. Par exemple la navigation PAGEACCUEIL – CARREFOURINFO – sport, exprime le fait qu'un utilisateur (son identifiant ne sera pas utilisé lors de la segmentation) a consulté les pages « PAGEACCUEIL », « CARREFOURINFO » et « sport » (dans cet ordre).

16.3 Segmentation des navigations

L'application fournira un ensemble de classes (clusters) qui regroupent les navigations du fichier d'entrée en fonction de leur similitude. Le degré de similitude sera basé sur la plus longue sous-séquence commune entre deux navigations.

Exemple 8 *Considérons les séquences suivantes en entrée : Séquence 1 = PAGEACCUEIL - CARREFOURINFO - sport - chat Séquence 2 = PAGEACCUEIL - internet - sport*

Ces deux séquences présentent une similitude de 57%. Elles partagent une sous-séquence (non-contigüe) maximale composée de 2 pages (« PAGEACCUEIL » suivi de « sport ») sur un total de 7 pages (taille cumulée des deux navigations).

En utilisant cette mesure, les navigations seront alors regroupées par similitude grâce à un algorithme agglomératif. Le principe de cet algorithme est le suivant :

1. *La première navigation est insérée dans le premier cluster ;*
2. *Le représentant du premier cluster est son unique navigation ;*
3. *Soit s, la séquence suivante. s est comparée à tous les clusters ;*
4. *Si la similitude avec le représentant le plus proche de s est supérieure au seuil minimum alors s est insérée dans le cluster correspondant ;*

5. *Sinon, un nouveau cluster est crée pour s ;*

6. *Si s n'est pas la dernière séquence à classer, alors retour à l'étape 3).*

Exemple 9 *Considérons les séquences suivantes en entrée : Séquence 1 = PAGEACCUEIL - CARREFOURINFO - sport Séquence 2 = PAGEACCUEIL - internet - chat Séquence 3 = PAGEACCUEIL - CARREFOURINFO - sport Séquence 4 = internet - actu - chat Séquence 5 = PAGEACCUEIL - CARREFOURINFO - CARREFOURINFO*

Ces séquences, selon le degré de similitude choisi, devraient être regroupées en deux clusters :

Cluster 1 : Séquence 1 = PAGEACCUEIL - CARREFOURINFO - sport Séquence 3 = PAGEACCUEIL - CARREFOURINFO - sport Séquence 5 = PAGEACCUEIL - CARREFOURINFO - CARREFOURINFO

Cluster 2 : Séquence 2 = PAGEACCUEIL - internet - chat Séquence 4 = internet - actu - chat

Pour chaque cluster, l'application fournira un représentant, permettant i) de le résumer et ii) de le comparer à la prochaine séquence à classer. Ce représentant sera le résultat d'une technique d'alignement appliquée au contenu de chaque cluster (i.e. aux séquences de navigation de ce cluster). Ainsi, l'alignement de plusieurs séquences d'un cluster donne une seule séquence qui synthétise l'ensemble du cluster.

Exemple. Considérons le cluster suivant, issu de l'étape 2 : Séquence 1 = PAGEACCUEIL - CARREFOURINFO - sport Séquence 3 = PAGEACCUEIL - CARREFOURINFO - sport Séquence 5 = PAGEACCUEIL - CARREFOURINFO - CARREFOURINFO

L'alignement sur ce cluster produit la séquence suivante : (PAGEACCUEIL : 3) - (CARREFOURINFO : 3) - (sport : 2 , CARREFOURINFO : 1)

Ce résultat s'interprète comme : « dans ce cluster, les utilisateurs commencent tous (' : 3') par consulter l'élément 'PAGEACCUEIL', puis continuent tous sur la page

'CARREFOURINFO' et enfin consultent soit l'élément 'sport' (dans 2 cas sur 3), soit l'élément CARREFOURINFO (dans 1 cas sur 3) ». L'ensemble des n clusters ainsi calculés permet d'obtenir un ensemble de n résumés (alignements des contenus des clusters). Ces résumés peuvent remplacer les données de détail à des fins de stockage, mais également d'analyses ultérieures (comptage, fouille).

16.4 Etude et expérimentations

La mise en œuvre de cette application permettra à Orange et à l'Inria d'étudier l'impact de la taille de la fenêtre d'accumulation sur la qualité des résultats en termes de clustering. En effet, une fenêtre d'accumulation d'une journée ne permet pas de prétendre à une étude en temps réel (et donc de répondre à des besoins de type flux de données). D'un autre côté une fenêtre de seulement quelques requêtes rendrait inutilisable les résultats du clustering. Il s'agit donc d'étudier le rapport taille-de-fenêtre/qualité-du-clustering de manière plus approfondie pour satisfaire les besoins de cette application sur le plan du temps réel.

16.5 Autres applications possibles

Les motifs séquentiels peuvent se retrouver dans beaucoup d'applications. Dans la suite je présente quelques exemples de telles applications.

Notre méthode pourrait être utilisée par une compagnie devant cartographier et fixer les routes des transports en commun. Les motifs fréquents extraits des comportements d'usagers inciteraient à ne pas mettre un arrêt devant ma maison, mais plutôt devant une maison mémorielle d'une grande personnalité où le flux de visiteurs est important. Elle permettrait également de fixer facilement les bons horaires des transports en commun en fonction des événements importants (Noël, jours de concerts, jours de grèves etc). Une étude des motifs séquentiels optimiserait ces horaires (selon les événements on pourrait anticiper les routes embouteillées dans les prochaines heures). Ainsi, les transports en commun représentent un domaine d'application tout à fait

envisageable pour notre approche.

Il y a quelques mois, j'ai entendu la radio France Info parler de la mise en place d'un programme national ou international qui permettrait d'étudier les comportements alimentaires des personnes et où les internautes étaient invités de remplir en ligne des formulaires contenants des questions sur leurs habitudes alimentaires. Voici encore une application possible de notre méthode. En allant encore plus loin, essayons de nous imaginer ces formulaires corrélés avec des parcours médicaux des internautes : une vraie mine d'or pour la médecine contemporaine.

Les données d'usage, de manière générale, sont désormais présentes sous forme de flux dans des domaines de plus en plus nombreux. Les usages du portail mobile d'Orange en sont un exemple mais on peut y ajouter la consommation électrique des foyers, l'utilisation de la TV numérique par un fournisseur d'accès à Internet, les usages du Web, les véhicules de location (qui commencent à être équipés de GPS permettant de suivre les parcours empruntés), etc. Toutes ces données d'usage sont vouées à être temporaires en raison de leur volume et de leur rapidité de production. D'un autre côté, toutes ces données peuvent révéler des informations précieuses à leurs propriétaires pour améliorer un service, sécuriser un système, augmenter une rentabilité, etc. Ces données seront certainement de plus en plus fournies sous forme séquentielle (évolutive) et se prêteront à des analyses telles que les nôtres.

CHAPITRE 17

PERSPECTIVES

BEAUCOUP de chemins partent en étoile de notre travail et chacun éveille notre intérêt ; quelques uns de ces chemins seront présentés dans la suite.

Nos méthodes de clustering ont été validées par des expérimentations et ont prouvé leur capacité à être intégrées dans le contexte des flux. Pourtant, beaucoup de choses restent à faire dans ce domaine. Premièrement, faire un résumé des méthodes de clustering représente un travail important. Les clusters dépendent de beaucoup de facteurs : la nature des données, le domaine d'application, les formes des clusters, la dimensionnalité des données, la distance utilisée etc. Par conséquent, il existe beaucoup d'améliorations possibles de notre méthode de clustering. Un premier travail en perspective consisterait dans la prise en compte de la forme des clusters dans le calcul de la séquence centroïde. Notre méthode favorise les clusters sphériques, alors que le centroïde d'un cluster elliptique n'est pas identique. J'envisage une méthode qui retienne une frontière du cluster et qui adapte le calcul du centroïde en fonction de celle-ci. D'autres caractéristiques du cluster (telles que la densité par régions et totale, plusieurs sous-centres par cluster et une relation d'appartenance issue de la logique floue) seraient alors considérées. Pour les batches contenant beaucoup de données, une

méthode de projection dans un espace de moindre dimension pourrait être envisagée (la forme des clusters seraient projetée dans un espace en 2D et les autres caractéristiques seraient résumées).

Notre méthode d'extraction de motifs séquentiels approximatifs a été appliquée pour un seul flux de données. Toutefois, il lui est possible de traiter plusieurs flux à la fois. Une optimisation de l'arbre préfixé pourrait être étudiée. Deux possibilités existent : garder un seul arbre pour tous les flux et ajouter un attribut d'identification des flux aux noeuds ou bien, garder un arbre par flux. Le composant ATGT s'adapterait à chacun de ces cas, vu sa capacité de traiter plusieurs séquences indépendamment, quel que soit leur flux d'origine.

La méthode AMi donne de très bons résultats. Toutefois, une étape préliminaire de détection d'éléments très éloignés du reste des éléments serait un plus pour la qualité des résultats donnés par AMi. Notre méthode de détection automatique d'anomalies pourrait être envisagée dans ce but.

L'ajout d'une caractéristique de détection de tendances au composant ATGT serait une piste très intéressante à suivre. Celle-ci lui permettrait de prévoir les séquences compressibles en avance et de les mettre les premières dans l'ordre de traitement. On pourrait également aller plus loin et en cas de période très abondante en données (donc fort besoin de traiter rapidement les entrées) de les compresser directement sans calculer les autres possibilités.

La technique d'alignement des séquences n'assure pas un résultat unique. Une méthode qui assure le meilleur alignement possible ou des améliorations des calculs serait un chemin à suivre.

Une application de ce travail dans la détection d'intrusions serait intéressante à étudier. La détection de comportements atypiques pourrait remplacer la détection des comportements les plus fréquents (les motifs séquentiels). La méthode de détection automatique d'anomalies que nous proposons représenterait le premier pas sur ce chemin.

Références

Agarwal, R.C., Aggarwal, C.C. & Prasad, V.V.V. (2000a). Depth first generation of long patterns. In *KDD '00 : Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 108–118, ACM, New York, NY, USA. 60

Agarwal, R.C., Aggarwal, C.C. & Prasad, V.V.V. (2000b). A tree projection algorithm for generation of frequent itemsets. *Journal of Parallel and Distributed Computing*, **61**, 350–371. 60

Agrawal, R. & Srikant, R. (1994a). Fast algorithms for mining association rules. In J.B. Bocca, M. Jarke & C. Zaniolo, eds., *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, 487–499, Morgan Kaufmann. 59, 64

Agrawal, R. & Srikant, R. (1994b). Fast algorithms for mining association rules. In J.B. Bocca, M. Jarke & C. Zaniolo, eds., *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, 487–499, Morgan Kaufmann. 62

Agrawal, R. & Srikant, R. (1994c). Fast algorithms for mining association rules in large databases. In *VLDB '94 : Proceedings of the 20th International Conference on Very Large Data Bases*, 487–499, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 25

Agrawal, R. & Srikant, R. (1995). Mining sequential patterns. *Data Engineering, International Conference on*, **0**, 3. 20, 21, 64

Agrawal, R., Imielinski, T. & Swami, A.N. (1993). Mining association rules between sets of items in large databases. In P. Buneman & S. Jajodia, eds., *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, 207–216, Washington, D.C. 19, 23, 24, 59

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H. & Verkamo, A.I. (1996). Fast discovery of association rules. 307–328. 59, 61, 62

Airoldi, E. & Faloutsos, C. (2004). Recovering latent time-series from their observed sums : network tomography with particle filters. In *KDD '04 : Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 30–39, ACM, New York, NY, USA. 99

Alon, N., Matias, Y. & Szegedy, M. (1996). The space complexity of approximating the frequency moments. In *STOC '96 : Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 20–29, ACM, New York, NY, USA. 79

Alon, N., Gibbons, P.B., Matias, Y. & Szegedy, M. (1999). Tracking join and self-join sizes in limited storage. In *PODS '99 : Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 10–20, ACM, New York, NY, USA. 79

Amir, A., Feldman, R. & Kashi, R. (1997). A new and versatile method for association generation. *Inf. Syst.*, **22**, 333–347. 61

Andreas Arning, P.R., Rakesh Agrawal (1996). A linear method for deviation detection in large databases. 37

Angiulli, F. & Pizzuti, C. (2005). Outlier mining in large high-dimensional data sets. *IEEE Trans. on Knowl. and Data Eng.*, **17**, 203–215. 39

Ayres, J., Flannick, J., Gehrke, J. & Yiu, T. (2002). Sequential pattern mining using a bitmap representation. In *KDD '02 : Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 429–435, ACM, New York, NY, USA. 64

Barbar'a, D., Dumouchel, W., Faloutsos, C., Haas, P.J., Hellerstein, J.M., Ioannidis, Y., Jagadish, H.V., Johnson, T., Ng, R., Poosala, V., Ross, K.A. & Sevcik, K.C. (1997). The new jersey data reduction report. *IEEE Data Engineering Bulletin*, **20**, 3–45. 77

Barnett, V. & T. Lewis, T., eds. (1994). *Outliers in statistical data*. John Wiley & Sons. 37

Bayardo, R.J., Jr. (1998). Efficiently mining long patterns from databases. In *SIGMOD '98 : Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, 85–93, ACM, New York, NY, USA. 61, 62

Ben-Gal, I. (2005). Outlier detection. 37, 41

Berkhin, P. (2002). Survey of clustering data mining techniques. Tech. rep., Accrue Software, San Jose, CA. 80

Bettini, C., Wang, X.S. & Jajodia, S. (1998). Mining temporal relationships with multiple granularities in time sequences. *Data Engineering Bulletin*, **21**, 32–38. 67

Birgit, H., Geert, W. & Koen, V. (2003). Web usage mining by means of multidimensional sequence alignment methods. In *WEBKDD 2002 - MiningWeb Data for Discovering Usage Patterns and Profiles*, 50–65, Springer Berlin / Heidelberg. 85

Borgelt, C. & Kruse, R. (2002). Induction of association rules : Apriori implementation. In *15th Conference on Computational Statistics (COMPSTAT 2002)*, Physica Verlag, Heidelberg, Germany. 61

Borgne, Y.A.L., Santini, S. & Bontempi, G. (2007). Adaptive model selection for time series prediction in wireless sensor networks. *Signal Process.*, **87**, 3010–3020. 99

Breunig, M.M., Kriegel, H.P., Ng, R.T. & Sander, J. (2000a). Lof : identifying density-based local outliers. *SIGMOD Records*, **29**, 93–104. 42, 170

Breunig, M.M., Kriegel, H.P., Ng, R.T. & Sander, J. (2000b). Lof : identifying density-based local outliers. *SIGMOD Rec.*, **29**, 93–104. 42

Brin, S., Motwani, R., Ullman, J.D. & Tsur, S. (1997). Dynamic itemset counting and implication rules for market basket data. In *SIGMOD '97 : Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, 255–264, ACM, New York, NY, USA. 60, 62

Ch, V., Banerjee, A., Kumar, V. & Chandola, V. (2007). Outlier detection : A survey. 41, 42

Chan, K.P. & Fu, A.W.C. (1999). Efficient time series matching by wavelets. In *ICDE '99 : Proceedings of the 15th International Conference on Data Engineering*, 126, IEEE Computer Society, Washington, DC, USA. 99

Chang, J.H. & Lee, W.S. (2003a). estwin : adaptively monitoring the recent change of frequent itemsets over online data streams. In *CIKM '03 : Proceedings of the twelfth international conference on Information and knowledge management*, 536–539, ACM, New York, NY, USA. 3, 61, 62, 76

Chang, J.H. & Lee, W.S. (2003b). Finding recent frequent itemsets adaptively over online data streams. In *KDD '03 : Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 487–492, ACM, New York, NY, USA. 3, 61, 62, 63, 75

Chang, J.H. & Lee, W.S. (2003c). Finding recent frequent itemsets adaptively over online data streams. In *KDD'03 : Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 487–492. 63, 95

Chang, J.H. & Lee, W.S. (2004). A sliding window method for finding recently frequent itemsets over online data streams. *JOURNAL OF INFORMATION SCIENCE AND ENGINEERING*, **20**, 753–762. 76

Charikar, M., Chen, K. & Farach-Colton, M. (2002). Finding frequent items in data streams. In *ICALP '02 : Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, 693–703, Springer-Verlag, London, UK. 63

Chawla, S. & Sun, P. (2006). Outlier detection, principles, techniques and applications. 42

Chen, G., Wu, X. & Zhu, X. (2005). Sequential pattern mining in multiple streams. *Data Mining, IEEE International Conference on*, **0**, 585–588. 4, 71, 177

Chen, Y., Dong, G., Han, J., Wah, B. & Wang, J. (2002). Multidimensional regression analysis of time-series data streams. 95, 96, 100, 105, 114

Cheng, J., Ke, Y. & Ng, W. (2008). A survey on algorithms for mining frequent itemsets over data streams. *Knowledge and Information Systems*, **16**, 1–27. 75, 77

Chi, Y., Wang, H., Yu, P.S. & Muntz, R.R. (2006). Catch the moment : maintaining closed frequent itemsets over a data stream sliding window. *Knowl. Inf. Syst.*, **10**, 265–294. 3, 62, 76

Cios, K.J., Pedrycz, W. & Swiniarski, R.W. (1998). Data mining methods for knowledge discovery. *IEEE Transactions on Neural Networks*, **9**, 1533–1534. 13

Cohen, E. & Strauss, M. (2003). Maintaining time-decaying stream aggregates. In *PODS '03 : Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 223–233. 95

Cormen, T.H., Leiserson, C.E., Rivest, R.L. & Stein, C. (2001). *Introduction to Algorithms, Second Edition*. McGraw-Hill Science/Engineering/Math. 86

Cormode, G. & Muthukrishnan, S. (2004). An improved data stream summary : The count-min sketch and its applications. 29–38. 79

Dasgupta, D. & Majumdar, N.S. (2002). Anomaly detection in multidimensional data using negative selection algorithm. In *CEC '02 : Proceedings of the Evolutionary Computation on 2002. CEC '02. Proceedings of the 2002 Congress*, 1039–1044, IEEE Computer Society, Washington, DC, USA. 40

Datar, M., Gionis, A., Indyk, P. & Motwani, R. (2002). Maintaining stream statistics over sliding windows. *SIAM J. Comput.*, **31**, 1794–1813. 61

Daubechies, I. (1992). *Ten lectures on wavelets*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA. 114, 161, 163

Dobra, A., Garofalakis, M., Gehrke, J. & Rastogi, R. (2002). Processing complex aggregate queries over data streams. In *SIGMOD '02 : Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, 61–72, ACM, New York, NY, USA. 79

Edgeworth, F. (1887). On discordant observations. In *Philosofical Magazine*, 364–375. 39

Elfeky, M.G. (2000). Incremental mining of partial periodic patterns in time-series databases. 67

Eskin, E., Arnold, A., Prerau, M., Portnoy, L. & Stolfo, S. (2002). *A geometric framework for unsupervised anomaly detection : Detecting intrusions in unlabeled data*. Kluwer. 42

Faloutsos, C., Ranganathan, M. & Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. In *SIGMOD '94 : Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, 419–429, ACM, New York, NY, USA. 99

Fayyad, U., Piatetsky-shapiro, G. & Smyth, P. (1996a). From data mining to knowledge discovery in databases. *AI Magazine*, **17**, 37–54. 2, 13, 14

Fayyad, U.M., Piatetsky-Shapiro, G. & Smyth, P. (1996b). From data mining to knowledge discovery : An overview. In *Advances in Knowledge Discovery and Data Mining*, 1–34. xv, 15

Friedman, J.H. (1997). Data mining and statistics : What's the connection ? 13, 14

Garofalakis, M., Gehrke, J. & Rastogi, R. (2002). Querying and mining data streams : you only get one look a tutorial. In *SIGMOD '02 : Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, 635–635, ACM, New York, NY, USA. 77

Garofalakis, M.N., Rastogi, R. & Shim, K. (1999). Spirit : Sequential pattern mining with regular expression constraints. In *VLDB '99 : Proceedings of the 25th International Conference on Very Large Data Bases*, 223–234, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 65

GIANNELLA, C., HAN, J., PEI, J., YAN, X. & YU, P. (2003). *Mining Frequent Patterns in Data Streams at Multiple Time Granularities*. In H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha (eds.), Next Generation Data Mining, AAAI/MIT. 3, 61, 62, 63, 71, 76, 96, 99, 100, 114

GIBBONS, P.B. (2001). Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *VLDB '01 : Proceedings of the 27th International Conference on Very Large Data Bases*, 541–550, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 78

GIBBONS, P.B. & MATIAS, Y. (1998). New sampling-based summary statistics for improving approximate query answers. In *SIGMOD '98 : Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, 331–342, ACM, New York, NY, USA. 78

GILBERT, A.C., KOTIDIS, Y., MUTHUKRISHNAN, S. & STRAUSS, M. (2001). Surfing wavelets on streams : One-pass summaries for approximate aggregate queries. In *VLDB '01 : Proceedings of the 27th International Conference on Very Large Data Bases*, 79–88, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 79

GOETHALS, B. (2003). Survey on frequent pattern mining. 61, 62

GRUBBS, F.E. (1969). Procedures for detecting outlying observations in samples. In *Technometrics, Vol 11, No 1*, 1–21. 39

GUSFIELD, D. (1997). *Algorithms on Strings, Trees, and Sequences : Computer Science and Computational Biology*. Cambridge University Press. 86

HAN, J. & KAMBER, M. (2001). *Data Mining, concepts and techniques*. Morgan Kaufmann. 67, 77, 80

HAN, J., DONG, G. & YIN, Y. (1999). Efficient mining of partial periodic patterns in time series database. In *ICDE '99 : Proceedings of the 15th International Conference on Data Engineering*, 106, IEEE Computer Society, Washington, DC, USA. 67

Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U. & Hsu, M.C. (2000a). Freespan : frequent pattern-projected sequential pattern mining. In *KDD '00 : Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 355–359, ACM, New York, NY, USA. 65

Han, J., Pei, J. & Yin, Y. (2000b). Mining frequent patterns without candidate generation. In *SIGMOD '00 : Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 1–12, ACM, New York, NY, USA. 60, 61

Han, J., Pei, J., Yin, Y. & Mao, R. (2004). Mining frequent patterns without candidate generation : A frequent-pattern tree approach. *Data Min. Knowl. Discov.*, **8**, 53–87. 60

Hodge, V. & Austin, J. (2004). A survey of outlier detection methodologies. *Artif. Intell. Rev.*, **22**, 85–126. 41

Hu, T. & Sung, S.Y. (2003). Detecting pattern-based outliers. *Pattern Recogn. Lett.*, **24**, 3059–3068. 40

Jain, A.K., Murty, M.N. & Flynn, P.J. (1999). Data clustering : a review. *ACM Comput. Surv.*, **31**, 264–323. 80

Jaing, M.F., Tseng, S.S. & Su, C.M. (2001). Two-phase clustering process for outliers detection. *Pattern Recogn. Lett.*, **22**, 691–700. 170

Jin, C., Qian, W., Sha, C., Yu, J.X. & Zhou, A. (2003a). Dynamically maintaining frequent items over a data stream. In *CIKM '03 : Proceedings of the twelfth international conference on Information and knowledge management*, 287–294, ACM, New York, NY, USA. 61

Jin, C., Qian, W., Sha, C., Yu, J.X. & Zhou, A. (2003b). Dynamically maintaining frequent items over a data stream. In *In Proc. Of CIKM*, 287–294, ACM Press. 63

Jin, W., Tung, A.K.H. & Han, J. (2001). Mining top-n local outliers in large databases. In *7th ACM SIGKDD international conference on Knowledge discovery and data mining*, 293–298. 159

John, G.H. (1997). *Enhancements to the data mining process*. Ph.D. thesis, Stanford, CA, USA. 14

Joshua Oldmeadow, J., Ravinutala, S. & Leckie, C. (2004). Adaptive clustering for network intrusion detection. In *8th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, vol. 3056 of *Lecture Notes in Computer Science*, 255–259. 159

Kaufman, L. & Rousseeuw, P.J. (1990). *Finding Groups in Data : An Introduction to Cluster Analysis*. Wiley-Interscience. 42

Keogh, E.J. & Pazzani, M.J. (1998). An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *KDD*, 239–243. 99

Knorr, E.M. & Ng, R.T. (1997). A unified notion of outliers : Properties and computation. In *In Proc. of the International Conference on Knowledge Discovery and Data Mining*, 219–222, AAAI Press. 42

Knorr, E.M. & Ng, R.T. (1998a). Algorithms for mining distance-based outliers in large datasets. In *VLDB '98 : Proceedings of the 24rd International Conference on Very Large Data Bases*, 392–403, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 39

Knorr, E.M. & Ng, R.T. (1998b). Algorithms for mining distance-based outliers in large datasets. In *Proc. 24th Int. Conf. Very Large Data Bases, VLDB*, 392–403. 42

Knorr, E.M. & Ng, R.T. (1998c). Algorithms for mining distance-based outliers in large datasets. In *24rd International Conference on Very Large Data Bases*, 392–403. 170

Knorr, E.M. & Ng, R.T. (1999). Finding intensional knowledge of distance-based outliers. In *VLDB '99 : Proceedings of the 25th International Conference on Very Large Data Bases*, 211–222, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 42

Knorr, E.M., Ng, R.T. & Tucakov, V. (2000). Distance-based outliers : Algorithms and applications. *VLDB Journal : Very Large Data Bases*, **8**, 237–253. 42

Kooi, R.P. (1980). *The optimization of queries in relational databases*. Ph.D. thesis, Cleveland, OH, USA. 78

Koudas, N. & Srivastava, D. (2005). Data stream query processing. In *ICDE '05 : Proceedings of the 21st International Conference on Data Engineering*, 1145, IEEE Computer Society, Washington, DC, USA. 33

Kum, H.C. (2004). Approximate mining of consensus sequential patterns. 4, 23, 85, 88, 154

Lazarevic, A. & Kumar, V. (2005). Feature bagging for outlier detection. In *KDD '05 : Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 157–166, ACM, New York, NY, USA. 38, 41

Leichter, J. & Whiteside, R.A. (1989). Implementing linda for distributed and parallel processing. In *ICS*, 41–49. 14

Li, H.F., Shan, M.K. & Lee, S.Y. (2008). Dsm-fi : an efficient algorithm for mining frequent itemsets in data streams. *Knowl. Inf. Syst.*, **17**, 79–97. 62, 76

Lin, J., Keogh, E.J., Lonardi, S. & chi Chiu, B.Y. (2003). A symbolic representation of time series, with implications for streaming algorithms. In *DMKD*, 2–11. 99

Lin, M.Y. & Lee, S.Y. (2002). Fast discovery of sequential patterns by memory indexing. In *DaWaK 2000 : Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery*, 150–160, Springer-Verlag, London, UK. 65

Lu, H., Han, J. & Feng, L. (1998). Stock movement prediction and n-dimensional inter-transaction association rules. In *In Proc. ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, 12–1. 21

Luo, C., Zhao, Y., Cao, L., Ou, Y. & Zhang, C. (2008). Exception mining on multiple time series in stock market. In *WI-IAT '08 : Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 690–693, IEEE Computer Society, Washington, DC, USA. 37

Manku, G.S. & Motwani, R. (2002). Approximate frequency counts over data streams. In *VLDB '02 : Proceedings of the 28th international conference on Very Large Data Bases*, 346–357, VLDB Endowment. 61, 62, 63, 76, 78

Mannila, H., Toivonen, H. & Verkamo, I. (1994). Efficient algorithms for discovering association rules. In *Knowledge Discovery in Databases (KDD'94)*, 181–192, U.M. Fayyad and R. Uthurusamy (eds.), AAAI Press 1994. 59

Mannila, H., Mannila, H., Toivonen, H., Toivonen, H., Verkamo, A.I. & Verkamo, A.I. (1997). Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, **1**, 259–289. 21

Markou, M. & Singh, S. (2003a). Novelty detection : A review - part 1 : Statistical approaches. *Signal Processing*, **83**, 2003. 39

Markou, M. & Singh, S. (2003b). Novelty detection : A review - part 2 : Neural network based approaches. *Signal Processing*, **83**, 2499–2521. 39

Masseglia, F., Cathala, F. & Poncelet, P. (1998). The psp approach for mining sequential patterns. In *PKDD '98 : Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery*, 176–184, Springer-Verlag, London, UK. 25, 64, 68, 124, 134

Masseglia, F., Poncelet, P. & Teisseire, M. (2003). Incremental mining of sequential patterns in large databases. *Data Knowl. Eng.*, **46**, 97–121. 66

Matias, Y., Vitter, J.S. & Wang, M. (1998). Wavelet-based histograms for selectivity estimation. *SIGMOD Rec.*, **27**, 448–459. 79

Mendenhall & Reinmuth (1993). *Statistics for Management and Economics*. Wadsworth. 39

Michalski, R.S. & Stepp, R. (1983). Automated construction of classifications conceptual clustering versus numerical taxonomy. 80

Motwani, R. (2002). Models and Issues in Data Stream Systems. 30

Oehler, K.L. & Gray, R.M. (1995). Combining image compression and classification using vector quantization. *IEEE Trans. Pattern Anal. Mach. Intell.*, **17**, 461–473. 80

Ogras, Y. & Ferhatosmanoglu, H. (2006). Online summarization of dynamic time series data. *The VLDB Journal*, **15**, 84–98. 99

Ord, K. (1996). Outliers in statistical data : V. barnett and t. lewis, 1994, 3rd edition, (john wiley & sons, chichester), 584 pp., [uk pound]55.00, isbn 0-471-93094-6. *International Journal of Forecasting*, **12**, 175–176. 43

Orlando, S., Palmerini, P. & Perego, R. (2001). Enhancing the apriori algorithm for frequent set counting. In *DaWaK '01 : Proceedings of the Third International Conference on Data Warehousing and Knowledge Discovery*, 71–82, Springer-Verlag, London, UK. 62

Orlando, S., Palmerini, P., Perego, R. & Silvestri, F. (2002). Adaptive and resource-aware mining of frequent sets. In *ICDM '02 : Proceedings of the 2002 IEEE International Conference on Data Mining*, 338, IEEE Computer Society, Washington, DC, USA. 62

Otey, M., Parthasarathy, S., Ghoting, A., Li, G., Narravula, S. & Panda, D. (2003). Towards nic-based intrusion detection. In *KDD '03 : Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 723–728, ACM, New York, NY, USA. 42

Palpanas, T., Vlachos, M., Keogh, E.J. & Gunopulos, D. (2008). Streaming time series summarization using user-defined amnesic functions. *IEEE Trans. Knowl. Data Eng.*, **20**, 992–1006. 100, 105, 108

Papadimitriou, S., Kitagawa, H., Gibbons, P. & Faloutsos, C. (2003). LOCI : fast outlier detection using the local correlation integral. In *19th International Conference on Data Engineering*. 42, 170

Papadimitriou, S., Sun, J. & Faloutsos, C. (2005). Streaming pattern discovery in multiple time-series. In *In VLDB*, 697–708. 95, 99

Park, J.S., Chen, M.S. & Yu, P.S. (1995a). An effective hash-based algorithm for mining association rules. In *SIGMOD '95 : Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, 175–186, ACM, New York, NY, USA. 62

Park, J.S., syan Chen, M. & Yu, P.S. (1995b). An effective hash-based algorithm for mining association rules. 175–186. 60

Parra, L., Deco, G. & Miesbach, S. (1996). Statistical independence and novelty detection with information preserving nonlinear maps. *Neural Comput.*, **8**, 260–269. 43

Parthasarathy, S., Zaki, M.J., Ogihara, M. & Dwarkadas, S. (1999). Incremental and interactive sequence mining. In *CIKM '99 : Proceedings of the eighth international conference on Information and knowledge management*, 251–258, ACM, New York, NY, USA. 65

Pei, J., Han, J., Asl, M.B., Pinto, H., Chen, Q., Dayal, U. & Hsu, M.C. (2001a). Prefixspan mining sequential patterns efficiently by prefix projected pattern growth. In *Proc.17th Int'l Conf. on Data Eng.*, 215–226. 25, 65, 134

Pei, J., Han, J., Lu, H., Nishio, S., Tang, S. & Yang, D. (2001b). H-mine : hyper-structure mining of frequent patterns in large databases. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, 441–448. 61

Pei, J., Han, J., Mortazavi-asl, B., Pinto, H., Chen, Q., Dayal, U. & chun Hsu, M. (2001c). Prefixspan : Mining sequential patterns efficiently by prefix-projected pattern growth. 215–224. 70, 71

Plantevit, M. (2008). *Extraction De Motifs Séquentiels Dans Des Données Multidimensionnelles*. Ph.D. thesis, Université de Montpellier II. 177

Poosala, V. (1997). *Histogram-based estimation techniques in database systems*. Ph.D. thesis, Madison, WI, USA. 78

Popivanov, I. (2002). Similarity search over time-series data using wavelets. In *ICDE '02 : Proceedings of the 18th International Conference on Data Engineering*, 212, IEEE Computer Society, Washington, DC, USA. 99

Portnoy, L., Eskin, E. & Stolfo, S. (2001). Intrusion detection with unlabeled data using clustering. In *ACM CSS Workshop on Data Mining Applied to Security*. 159

Pyle, D. (1999). *Data preparation for data mining*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 38

Rafiei, D. & Mendelzon, A. (1997). Similarity-based queries for time series data. In *SIGMOD '97 : Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, 13–25, ACM, New York, NY, USA. 99

Raïssi, C. & Plantevit, M. (2008). Mining multidimensional sequential patterns over data streams. In *DaWaK*, 263–272. 70, 177

Raïssi, C. & Poncelet, P. (2007). Sampling for sequential pattern mining : From static databases to data streams. In *Data Mining, IEEE International Conference on*, 631–636. 70, 177, 178

Raïssi, C., Poncelet, P. & Teisseire, M. (2005). Need for speed : Mining sequential patterns in data streams. In *BDA*. 4, 69

Raissi, C., Poncelet, P. & Teisseire, M. (2006). SPEED : Mining Sequential Pattens in Data Streams. In *IEEE IS*. 69

Rao, S. & Pisharam, P. (1990). A noise-reduction neural network as a preprocessing stage in the svd based method of harmonic retrieval. 491–494 vol.1. 39

Raphisak, P., Schuckers, S. & de Jongh Curry, A. (2004). An algorithm for emg noise detection in large ecg data. 369–372. 37

Ratle, F., Kanevski, M., Terrettaz-Zufferey, A.L., Esseiva, P. & Ribaux, O. (2007). *A Comparison of One-Class Classifiers for Novelty Detection in Forensic Case Data*, vol. 4881/2007. Springer Berlin. 37

Rousseeuw, P. & Leroy, A.M., eds. (1996). *Robust Regression and Outlier Detection*. Wiley-IEEE. 43

royal.pingdom.com (2008). Internet 2008 in numbers. 1

Saha, B.N., Ray, N. & Zhang, H. (2009). Snake validation : A pca-based outlier detection method. 549–552. 43

Savasere, A., Omiecinski, E. & Navathe, S. (1995). An efficient algorithm for mining association rules in large databases. 60, 62

Sequeira, K. & Zaki, M. (2002). Admit : anomaly-based data mining for intrusions. In *KDD '02 : Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 386–395, ACM, New York, NY, USA. 170

Shatkay, H. & Zdonik, S.B. (1996). Approximate queries and representations for large data sequences. In *ICDE '96 : Proceedings of the Twelfth International Conference on Data Engineering*, 536–545, IEEE Computer Society, Washington, DC, USA. 99

Singh, S. & Markou, M. (2004). An approach to novelty detection applied to the classification of image regions. 39

Sneath, P. & Sokal, R.R. (1973). *Numerical Taxonomy*. San Francisco : W.H. Freeman. 80

Srikant, R. (1996). *Fast algorithms for mining association rules and sequential patterns*. Ph.D. thesis, supervisor-Naughton,, Jeffrey F. 62

Srikant, R. & Agrawal, R. (1995). Mining generalized association rules. In *VLDB '95 : Proceedings of the 21th International Conference on Very Large Data Bases*, 407–419, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 59

Srikant, R. & Agrawal, R. (1996). Mining sequential patterns : Generalizations and performance improvements. In P.M.G. Apers, M. Bouzeghoub & G. Gardarin, eds., *Proc. 5th Int. Conf. Extending Database Technology, EDBT*, vol. 1057, 3–17, Springer-Verlag. 64

Stollnitz, E.J., DeRose, T.D. & Salesin, D.H. (1995). Wavelets for computer graphics : A primer, part 1. *IEEE Computer Graphics and Applications*, **15**, 76–84. 165

Tatbul, N., Çetintemel, U., Zdonik, S., Cherniack, M. & Stonebraker, M. (2003). Load shedding in a data stream manager. In *VLDB '2003 : Proceedings of the 29th international conference on Very large data bases*, 309–320, VLDB Endowment. 3

Teng, W.G., Chen, M.S. & Yu, P.S. (2003). A Regression-Based Temporal Pattern Mining Scheme for Data Streams. In *VLDB*, 93–104. 63, 95, 99, 100

Toivonen, H. (1996). Sampling large databases for association rules. In *VLDB '96 : Proceedings of the 22th International Conference on Very Large Data Bases*, 134–145, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 60, 62

Vitter, J.S. (1985). Random sampling with a reservoir. *ACM Trans. Math. Softw.*, **11**, 37–57. 78

Walfish, S. (2006). A review of statistical outlier methods. 43

Wang, J. & Han, J. (2004). Bide : Efficient mining of frequent closed sequences. In *ICDE '04 : Proceedings of the 20th International Conference on Data Engineering*, IEEE Computer Society, Washington, DC, USA. 65

Wang, R.Y., Reddy, M.P. & Kon, H.B. (1995). Toward quality data : an attribute-based approach. *Decis. Support Syst.*, **13**, 349–372. 38

Williams, G., Baxter, R., He, H., Hawkins, S. & Gu, L. (2002). A comparative study of rnn for outlier detection in data mining. In *ICDM '02 : Proceedings of the 2002 IEEE International Conference on Data Mining*, 709, IEEE Computer Society, Washington, DC, USA. 38, 41

Xifeng Yan, R.A., Jiawei Han (2003). Clospan : Mining closed sequential patterns in large datasets. 65

Xu, R. & Ii (2005). Survey of clustering algorithms. **16**, 645–678. 80

Yang, J. (2008). 77

Yang, J., Wang, W. & Yu, P.S. (2000). Mining asynchronous periodic patterns in time series data. In *KDD '00 : Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 275–279, ACM, New York, NY, USA. 67

Ye, M., Li, X. & Orlowska, M.E. (2009). Projected outlier detection in high-dimensional mixed-attributes data set. *Expert Systems with Applications*, **36**, 7104 – 7113. 43

Yi, B.K. & Faloutsos, C. (2000). Fast time sequence indexing for arbitrary lp norms. In *VLDB*, 385–394. 99

Young, R.K. (1995). *Wavelet Theory and Its Applications*. Kluwer Academic Publishers Group. 161

Zaki, M.J. (2000). Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, **12**, 372–390. 60

Zaki, M.J. (2001). Spade : An efficient algorithm for mining frequent sequences. *Machine Learning*, **42**, 31–60. 64

Zaki, M.J., Parthasarathy, S., Ogihara, M. & Li, W. (1997). New algorithms for fast discovery of association rules. In *In 3rd Intl. Conf. on Knowledge Discovery and Data Mining*, 283–286, AAAI Press. 60

Zaïane, O.R. (1999). cours cmpt690 principles of knowledge discovery in databases. 1–11. 15, 16

Zhang, C. & Selinus, O. (1998). Statistics and gis in environmental geochemistry – some problems and solutions. *Journal of Geochemical Exploration*, **64**, 339 – 354. 41

Zhang, M., Kao, B., Cheung, D. & Yip, C.L. (2002). Efficient algorithms for incremental update of frequent sequences. **2336**, 186–? ? 65

Zheng, Q., Xu, K., Ma, S. & Lv, W. (2002). The Algorithms of Updating Sequential Patterns. *ArXiv Computer Science e-prints*. 66

Zhong, S., Khoshgoftaar, T.M. & Seliya, N. (2007). Clustering-based network intrusion detection. *International Journal of Reliability, Quality and Safety Engineering*, **14**. 159

Zhu, Y. & Shasha, D. (2002). Statstream : statistical monitoring of thousands of data streams in real time. In *VLDB '02 : Proceedings of the 28th international conference on Very Large Data Bases*, 358–369, VLDB Endowment. 95, 96, 99, 100

Bibliographie personnelle

Articles dans des revues internationales avec comité de lecture

- MARASCU, A. & MASSEGLIA, F. (2009). Atypicity Detection in Data Streams : a Self-Adjusting Approach. In *Intelligent Data Analysis Journal (IDA).* A paraître.
- MASSEGLIA, F. & PONCELET, P. & TEISSEIRE, M. & MARASCU, A. (2008). Web Usage Mining : Extracting Unexpected Periods from Web Logs. In *Data Mining and Knowledge Discovery (DMKD) Journal.* Springer Netherlands (Ed). 16(1) : 39-65.
- MARASCU, A. & MASSEGLIA, F. (2006). Mining Sequential Patterns from Data Streams : a Centroid Approach ». In *Journal for Intelligent Information Systems (JIIS).* Issue 27, Number 3, pp 291-307.

Articles dans des conférences internationales avec comité de sélection

- MARASCU, A. & MASSEGLIA, F. & LECHEVALLIER, Y. (2010). A Fast Approximation Strategy for Summarizing a Set of Streaming Time Series. In *25th Annual ACM Symposium on Applied Computing.* Sierre, Suisse. A paraître.
- MARASCU, A. & MASSEGLIA, F. (2009). A Multi-Resolution Approach for Atypical Behaviour Mining. In *13th Pacific-Asia Conference on Knowledge Discovery*

and Data Mining (PAKDD'09). Bangkok, Thailand.

– SINGH, G. & MASSEGLIA, F. & FIOT, C. & MARASCU, A. & PONCELET, P. (2009). Data Mining for Intrusion Detection : from Outliers to True Intrusions. In *13th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'09)*. Bangkok, Thailand.

– MARASCU, A. & MASSEGLIA, F. (2009). Parameterless Outlier Detection in Data Streams. In *24th Annual ACM Symposium on Applied Computing (ACM SAC'09)*. Honolulu, Hawaii, USA.

Articles dans des conférences nationales avec comité de sélection

– MARASCU, A. & MASSEGLIA, F. & LECHEVALLIER, Y. (2010). REGLO : une nouvelle stratégie pour résumer un flux de séries temporelles. In *Extraction et gestion de connaissances,* Hammamet, Tunisie. A paraître.

– SINGH, G. & MASSEGLIA, F. & FIOT, C. & MARASCU, A. & PONCELET, P. (2009). Collaborative Outlier Mining for Intrusion Detection. In *Extraction et gestion des connaissances*, Strasbourg, France, pp 313-324. (Nominé pour le prix du meilleur papier applicatif).

– MARASCU, A. & MASSEGLIA, F. (2009). Détection d'enregistrements atypiques dans un flot de données : une approche multi-résolution. In *Extraction et gestion des connaissances*, Strasbourg, France, pp 455-456.

– MARASCU, A. & MASSEGLIA, F. (2009). Classification de flots de séquences basée sur une approche centroïde, *INFORSID Informatique des organisations et systèmes d'information et de décision*, Hammamet, Tunisie.

– MARASCU, A. & MASSEGLIA, F. (2006). Extraction de motifs séquentiels dans les flots de données d'usage du Web », *Extraction et Gestion des Connaissances*, Lille, France.

– MASSEGLIA, F. & PONCELET, P. & TEISSEIRE, M. & MARASCU, A. (2006). Usage Mining : extraction de périodes denses à partir des logs Web ». *Extraction et Gestion des Connaissances*, Lille, France.

Articles dans des ateliers internationaux avec comité de sélection

- MASSEGLIA, F. & PONCELET, P. & TEISSEIRE, M. & MARASCU, A. (2005). Web Usage Mining : Extracting Unexpected Periods from Web Logs. In *IEEE 2nd Workshop on Temporal Data Mining. Held in conjunction with ICDM'05*, Houston, USA.
- MARASCU, A. & MASSEGLIA, F. (2005). Mining Data Streams for Frequent Sequences Extraction. In *IEEE first Workshop on Mining Complex Data. Held in conjunction with ICDM'05*, Houston, USA.
- MARASCU, A. & MASSEGLIA, F. (2005). Mining Sequential Patterns from Temporal Streaming Data. In *ECML/PKDD first Workshop on Mining Spatio-Temporal Data. Held in conjunction with PKDD'05*, Porto, Portugal. .

Article dans des ateliers nationaux

- MARASCU, A. & MASSEGLIA, F. (2007). Limites d'une approche incrémentale pour la segmentation de séquences dans les flux. In *Atelier Flux de Données*, Namur, Belgique, p. 49-60.
- MARASCU, A. & MASSEGLIA, F. (2006). Classification de flots de séquences basée sur une approche centroïde, In *Fouille de données complexes dans un processus d'extraction des connaissances (FDC'06)*, Lille, France, p. 131-139.

Livres

- VIRTOPEANU, I. & MARASCU, A. & VIERU, G. & GOLEA, G. & MIU,S. & ENACHE, C. (2003). Analyse mathématique pour la première et terminale serie S (Analiza matematica pentru elevii din clasele a XI-a si a XII-a), *Editura Reprograph*, Craiova, Roumanie.
- VIRTOPEANU, I. & ROSESCU, R. & VIERU, G. & PASARE, I. & TATOMIRESCU, P. (2003). Guide des mathématiques (Ghid pentru matematica, *Editura Reprograph*, Craiova, Roumanie (collaborateur).