HAL
open science

Prédire et influencer l'apparition des événements dans une séquence complexe

Lina Fahed

▶ **To cite this version:**

Lina Fahed. Prédire et influencer l'apparition des événements dans une séquence complexe. Algorithme et structure de données [cs.DS]. Université de Lorraine, 2016. Français. NNT : 2016LORR0125 . tel-01470524

HAL Id: tel-01470524

https://theses.hal.science/tel-01470524

Submitted on 17 Feb 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ DE LORRAINE

AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact : ddoc-theses-contact@univ-lorraine.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

École doctorale IAEM Lorraine

Prédire et influencer l'apparition des événements dans une séquence complexe

THÈSE

présentée et soutenue publiquement le 21/10/2016

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Lina Fahed

Composition du jury

Président :	Vincent Chevrier	Professeur, Université de Lorraine
Rapporteurs :	Julien Velcin	Maître de conférences, Université Lumière Lyon 2
	Cyril De Runz	Maître de conférences, Université de Reims
Examinateurs :	Patrick Gallinari	Professeur, Université Paris 6
	Yolaine Bourda	Professeur, CentraleSupélec Gif-sur-Yvette
Invité :	Jean-Philippe Blanchard	Crédit Agricole S.A.
Directrice :	Anne Boyer	Professeur, Université de Lorraine
Co-encadrante :	Armelle Brun	Maître de conférences, Université de Lorraine

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

Je tiens tout d'abord à adresser mes remerciements à mes encadrantes de thèse : Anne Boyer et Armelle Brun, pour leur soutien, leur disponibilité et les échanges intéressants qu'on a eu pendant la thèse et pour tous les conseils qu'elles m'ont prodigué qui m'ont permis d'avancer et de donner le meilleur de moi-même.

Je remercie ensuite le groupe Crédit Agricole S.A. pour avoir soutenu financièrement cette thèse et en particulier Jean-Philippe Blanchard et Isabelle Serot pour leur collaboration et pour leurs conseils avisés qui m'ont permis de mener ce travail de thèse dans un cadre industriel.

Je tiens aussi à remercier les membres du jury : Julien Velcin et Cyril De Runz en tant que rapporteurs, Vincent Chevrier, Patrick Gallinari et Yolaine Bourda en tant qu'examinateurs ainsi que Jean-Philippe Blanchard en tant qu'invité pour avoir accepté d'évaluer ces travaux.

Mes remerciements vont aussi à tous les membres de l'équipe KIWI que j'ai côtoyés au quotidien. J'ai beaucoup apprécié l'ambiance de travail et les moments agréables passés avec eux. Une pensée particulière est adressée à mes amis : Laura Infante-Blanco, Hayat Nasser, Marharyta Aleksandrova et Charif Haydar. La liste n'étant pas exhaustive, mes remerciements les plus sincères sont adressés à toute personne qui a contribué de près ou de loin à la réalisation de cette thèse.

Enfin, je tiens à exprimer ma gratitude envers mes parents, mes sœurs et mon frère qui m'ont poussé jusqu'au bout pour effectuer cette thèse et sans lesquels les remerciements précédents n'auraient jamais eu l'occasion d'être lus.

Je dédie cette thèse

Table des matières

Chapitre 1
Introduction **3**

1.1 Contexte général 3
1.2 Problématique 5
1.3 Contributions 6
1.3.1 Un algorithme de fouille de règles d'épisode : DEER 7
1.3.2 Un algorithme de détection de l'émergence de règles d'épisode : EER . . 7
1.3.3 Un algorithme de détection d'événements influenceurs : IE 7
1.4 Application et corpus d'évaluation 8
1.5 Présentation du plan de la thèse 9

Chapitre 2
État de l'art **11**

2.1 Principes de la fouille de données 12
2.1.1 Introduction 12
2.2 Les données simples 13
2.2.1 Les transactions et les séquences 13
2.2.2 Autres types de données : les graphes, les textes, etc. 15
2.3 La fouille de données simples 16
2.3.1 L'extraction de motifs et de règles d'association 16
2.3.2 La classification 20
2.3.3 Le clustering 21
2.4 Vers des données complexes 22
2.4.1 Quelques exemples de données complexes 23
2.4.1.1 Les séquences d'événements 23
2.4.1.2 Les graphes dynamiques 23
2.4.1.3 Les textes bruts 24
2.4.1.4 Les flux de données 25

2.4.2 Les défis de la fouille de données complexes 25
2.4.3 L'impact de données complexes sur la fouille de données : les nouvelles connaissances fouillées 26
2.5 La fouille de séquences d'événements 28
2.5.1 Importance et exemples d'applications 29
2.5.2 Définitions 29
2.5.3 Algorithmes d'extraction d'épisodes dans une séquence d'événements . . 32
2.5.3.1 L'algorithme WINEPI 32
2.5.3.2 L'algorithme MINEPI 34
2.5.3.3 Les mesures de fréquence pour l'extraction d'épisodes 36
2.5.4 Extraction des épisodes avec contraintes 37
2.5.5 Extraction des règles d'épisode 38
2.6 Prédiction dans une séquence d'événements 39
2.6.1 Les approches existantes 40
2.6.2 Discussion 41
2.7 Détection de l'émergence dans un flux d'événements 42
2.7.1 Les approches existantes 42
2.7.1.1 La fouille de flux d'événements 42
2.7.1.2 La détection d'émergence 44
2.7.2 Discussion 45
2.8 Influencer les événements futurs 45
2.8.1 Les approches existantes 46
2.8.2 Discussion 48
2.9 Plateformes de traitement de séquences et de flux 49

Chapitre 3
Un algorithme de fouille de règles d'épisode : DEER **51**

3.1 Introduction 51
3.1.1 Les défis de la prédiction d'événements distants 52
3.1.2 Principe de l'algorithme $DEER$ 54
3.2 Nouveaux concepts et redéfinitions 56
3.3 Les étapes de l'algorithme $DEER$ 58
3.3.1 Identification du préfixe 59
3.3.2 Identification de la conséquence 59
3.3.3 Complétion de l'antécédent 60
3.3.4 La confiance temporelle 61
3.4 Discussion comparative : efficacité de $DEER$ en terme de temps d'exécution . . . 64

3.4.1 Choix d'algorithme de l'état de l'art 64
3.4.2 Comparaison sur des exemples 65
3.4.3 Synthèse : *DEER* vs. un algorithme de la littérature 66
3.5 Expérimentations 66
3.5.1 Caractéristiques des règles extraites 67
3.5.1.1 Phase d'initialisation 67
3.5.1.2 Impact de seuil de support et de confiance sur les règles extraites 67
3.5.1.3 Impact de gap inséré entre l'antécédent et la conséquence de règles 70
3.5.2 Évaluation de la confiance temporelle 70
3.5.3 Performance en prédiction 72
3.5.4 Comparaison des règles formées par *DEER* à celles de l'état de l'art . . . 73
3.5.5 Temps d'exécution 74
3.5.6 Discussion 75
3.6 Synthèse 76

Chapitre 4
EER : Un algorithme de détection de l'émergence de règles d'épisode 79

4.1 Introduction 79
4.2 Principe de la plate-forme de traitement distribué *STORM* 81
4.3 Principe de l'algorithme *EER* 81
4.3.1 Phase d'initialisation 84
4.3.2 Parallélisation : Topologie proposée pour *STORM* 84
4.3.2.1 Le traitement effectué dans les unités de calcul *Spouts* 84
4.3.2.2 Le traitement effectué dans les unités de calcul *Bolts* : premier niveau 84
4.3.2.3 Le traitement effectué dans les unités de calcul *Bolts* : deuxième niveau 85
4.3.3 Phase de détection de l'émergence 86
4.3.3.1 La détection de l'émergence préliminaire 86
4.3.3.2 La détection de l'émergence finale 86
4.4 Expérimentations 88
4.4.1 Corpus : flux d'événements 88
4.4.2 Phase d'initialisation 90
4.4.3 Caractéristiques de règles extraites 91
4.4.4 Évaluation de la performance 93
4.4.5 Impact du choix de se baser sur uniquement les règles similaires aux règles de référence 96

4.4.6 Temps d'exécution 97
4.4.7 Discussion 98
4.5 Synthèse 98

Chapitre 5
IE : un algorithme de détection d'événements influenceurs 101

5.1 Introduction 101
5.2 Principe de l'algorithme IE 104
5.3 Les trois types d'événements influenceurs 106
5.3.1 Événements influenceurs du support : événements de disparition 107
5.3.2 Événements influenceurs de la confiance 107
5.3.3 Événements influenceurs de la distance 108
5.4 Intégration de l'algorithme IE dans un algorithme de fouille de règles d'épisode 109
5.4.1 Se baser sur un algorithme traditionnel 110
5.4.2 Se baser sur un algorithme dédié 111
5.4.3 Synthèse 112
5.5 Expérimentations 112
5.5.1 Temps d'exécution 113
5.5.2 Caractéristiques des événements influenceurs 114
5.5.2.1 Événements de disparition 115
5.5.2.2 Événements influenceurs de confiance 115
5.5.2.3 Événements influenceurs de distance 119
5.5.2.4 Discussion 122
5.6 Synthèse 122

Chapitre 6
Conclusion et perspectives 125

6.1 Résumé de contributions 126
6.1.1 La prédiction au plus tôt des événements futurs dans une séquence d'événements 126
6.1.2 La détection au plus tôt de l'émergence de nouvelles règles d'épisode dans un flux d'événements 126
6.1.3 La détection des événements "influenceurs" des événements futurs dans une séquence d'événements 127
6.2 Perspectives 128

Bibliographie 131

Table des figures

2.1 Exemple d'une séquence d'événements S. 30
2.2 Les différents types d'épisodes. 30
2.3 Schéma des outils de fouille de données (De Francisci Morales, 2013). 49

3.1 Un exemple d'une règle essentielle avec une conséquence distante : R : *[problème de carte bancaire, s'intéresser à une offre d'une banque concurrente]* $\rightarrow$ *[demande de fermeture de compte]* ; la conséquence est 20 jours loin de l'antécédent, la règle est essentielle car il n'y a pas besoin d'ajouter l'événement *relation tendue avec le conseiller* (en couleur gris) pour réaliser une prédiction fiable. 55
3.2 Présentation des sous-fenêtres contenant un exemple d'une occurrence de la règle d'épisode $p_1, p_2 \rightarrow q_1$. 56
3.3 Deux exemples d'occurrences de la règle $R : p_1, p_2 \rightarrow q_1$: une occurrence en couleur rouge et une autre en bleu. L'occurrence en bleu est une occurrence minimale. . . 58
3.4 Exemple d'une séquence d'événements S. 58
3.5 Exemple d'une occurrence de la règle $R : p_1, p_2 \rightarrow q_1$. 61
3.6 Exemple de la complétion de l'antécédent de la règle $R : p_1, p_2 \rightarrow q_1$ avec l'épisode p_3 et du glissement de $Win_{invisible}$. 61
3.7 Nombre de règles en fonction de seuil de support ($minsupp$). 69
3.8 Nombre de règles en fonction de seuil de confiance ($minconf$). 69
3.9 Nombre de règles en fonction de taille de *gap* entre l'antécédent et la conséquence ($w_{invisible}$). 71
3.10 Nombre de règles en fonction de taille de *gap* entre l'antécédent et la conséquence ($w_{invisible}$) et de seuil de confiance temporelle ($minconf_t$). 71
3.11 Précision, rappel en fonction de taille de *gap* entre l'antécédent et la conséquence ($w_{invisible}$). 73
3.12 Temps d'exécution relatif en fonction de taille de *gap* entre l'antécédent et la conséquence ($w_{invisible}$) ($temps\ relatif = 1$ représente le temps d'exécution de $MINEPI$). 75

4.1 La composition d'une topologie dans la plate-forme $STORM$ (Smart Grid Big Data Management Team, 2014). 81
4.2 Un exemple de répartition des instants pour calculer le taux de croissance, $k = 5$. 87
4.3 Schéma de l'algorithme EER. 90
4.4 Nombre de règles en fonction de leur support. 92
4.5 Pourcentage de règles similaires, identiques et non similaires aux règles de référence en fonction de seuil de mesure de similarité ($max_{similarité}$). 93

4.6 Nombre moyenne de règles similaires aux règles de référence en fonction du seuil de mesure de similarité ($max_{similarité}$). 94
4.7 Précision et rappel : (A) en fonction de seuil d'émergence préliminaire ($minsupp_{émergPré}$), (B) en fonction de seuil de taux de croissance ($min_{croissance}$). 96
4.8 Précision et rappel en fonction de seuil de taux de croissance ($min_{croissance}$) : (i) avec la prise en compte de similarité avec les règles de référence, (ii) sans la prise en compte de similarité. 97

5.1 Influence sur le support (la disparition) : (A) les règles influencées, (B) les événements influenceurs et (C) les conséquences influencées, pour $\theta_{Infl_{dist}} = -0,8$. . . 116
5.2 Les cas d'influence sur la confiance ($\#R'$) en fonction de la mesure d'influence sur la confiance. 117
5.3 Influence sur la confiance : (A) les règles influencées, (B) les événements influenceurs et (C) les conséquences influencées. 118
5.4 Les cas d'influence sur la distance ($\#R'$) en fonction de la mesure d'influence sur la distance. 120
5.5 Influence sur la distance : (A) les règles influencées, (B) les événements influenceurs et (C) les conséquences influencées. 121

Chapitre 1

Introduction

1.1 Contexte général

Des données numériques de plus en plus volumineuses, de plus en plus variées et provenant de plus en plus de sources différentes apparaissent et sont désormais disponibles. Par exemple, 2,5 trillions d'octets de données sont générés chaque jour provenant de nombreuses sources comme le Web, les réseaux de télécommunications, etc. Cela signifie que 90% des données dans le monde ont été créées au cours des deux dernières années seulement (IBM, 2015). Ces données représentent une mine d'information et c'est la raison pour laquelle il devient primordial de les exploiter.

Ces données sont souvent qualifiées de données complexes. Nous les définissons comme suit : Les **données complexes** [1] sont les données : (i) très volumineuses : il y a une immense quantité de données générées, (ii) véloces : la fréquence à laquelle les données sont générées, capturées et partagées est très élevée, (iii) variées : les données sont bruitées, brutes et non structurées (comme des mots, images, vidéos, séquence de signaux de capteurs, etc.) et parviennent potentiellement de plusieurs sources. Ces trois caractéristiques de données complexes sont traditionnellement représentées par la règle des "3V" (Laney, 2001). Cette règle est encore largement utilisée aujourd'hui pour décrire le phénomène récent de données complexes.

La fouille de données est le domaine qui consiste à traiter, gérer et créer des modèles de données à partir d'une grande quantité de données. Le processus traditionnel de fouille de données suit plusieurs étapes : collecter les données, les traiter, les transformer et appliquer des algorithmes dits de fouille dans le but de les analyser et d'identifier des associations (des liens) cachées dans ces données (Han et al., 2011). Le modèle ainsi formé reflète les connaissances présentes dans les données et est utilisé pour résoudre des problèmes applicatifs tels que l'étude du panier de la ménagère (Kaur and Kang, 2016), la gestion de la relation client (Linoff and Berry, 2011), la recommandation d'achats (Zhao et al., 2014), la détermination du profil client (van Dam and van de Velden, 2015), etc. Dans la fouille de données, on distingue deux types de modélisation : (i) la modélisation descriptive qui vise à décrire le mécanisme de génération des données. Nous pouvons par exemple citer l'estimation de distributions de probabilité sur les données, la recherche de *clusters* de données, etc. (ii) la modélisation prédictive qui vise à prédire des valeurs dans les données à l'aide du modèle généré. Nous pouvons par exemple citer la classification, l'extraction de règles d'association, etc. Dans cette thèse, nous nous intéressons particulièrement à la modélisation prédictive de données.

Les données complexes sont très riches en informations cachées, ce qui augmente leur pouvoir prédictif (Lohr, 2012). Ce pouvoir prédictif est très important dans de nombreux domaines,

1. Cette définition de données complexes correspond à la définition usuelle de *Big Data*.

comme la santé publique, la prévision économique (Lohr, 2012), la sécurité des réseaux (Zhang and Lee, 2000) etc. Plusieurs organisations ont récemment pris conscience de cette importance. Par exemple l'Organisation des Nations Unies (ONU) a lancé en 2009 le projet "Global Pulse" (Global Pulse, 2009) qui a pour objectif d'exploiter, rapidement et efficacement, les flux de données provenant de toutes traces numériques. Cette exploitation a pour but de prédire, par exemple la perte d'emploi ou des pics de maladies dans une région. Leur objectif est d'utiliser les signaux faibles [2] numériques pour lancer des alertes précoces dans l'intérêt public. Dans le même ordre d'idées, selon Radinsky (Radinsky, 2014), un modèle prédictif s'applique de la façon suivante : en observant les faits et les phénomènes statistiques, si un événement (ou une combinaison d'événements) apparaît ou se reproduit, renseignant ainsi le risque d'occurrence d'un événement futur, il est possible de lancer une alerte et de réagir en injectant un (ou des) événement(s) dans le but de diminuer le risque de cet événement futur. Dans ce cas, nous pouvons dire qu'un modèle prédictif permet d'impacter et d'influencer le futur.

Depuis de nombreuses années, les modèles prédictifs de fouille de données ont été dédiés et validés sur des données dites simples (provenant d'une source unique, avec une structure nette, etc.). Cependant, la généralisation et la prédominance de données complexes amènent à de nouveaux défis principalement liés à la prise en compte des nouvelles caractéristiques de ces données (volume, vélocité et variété) dans la modélisation prédictive. Par conséquent, proposer un modèle prédictif adapté aux données complexes est devenu incontournable pour mieux prédire voire influencer le futur. Parce que prédire au plus tôt l'occurrence d'un événement permet de réagir tôt, d'influencer l'apparition des événements futurs et donc de diminuer au maximum le risque associé, si nécessaire. C'est pourquoi nous nous intéressons à ce sujet.

La plupart du temps, la modélisation prédictive est réalisée sur des données transactionnelles (Weiss and Indurkhya, 1998; Faith et al., 2013; Choi et al., 2015). Cependant, dans plusieurs applications de fouille de données (Mannila et al., 1997), les données se présentent sous la forme d'une séquence unique d'événements, où un événement représente un ou plusieurs items (un item représente l'unité de données la plus petite). Nous pouvons par exemple citer la séquence d'alarmes dans un réseau de télécommunication, le comportement d'un utilisateur dans une interface utilisateur, la séquence de messages dans un blog ou dans un réseau social, etc. Dans ce type de séquence, chaque événement est généralement associé à son instant d'apparition. Une collection d'événements relativement proches, appelée *épisode* est souvent extraite à partir de ces séquences (Mannila et al., 1997). La modélisation d'une **séquence unique d'événements** est un problème fondamental et complexe de la fouille de données. En effet, la séquence est généralement très longue et peut être sans limite (infinie). C'est la raison pour laquelle la zone d'impact d'un événement dans la séquence n'est pas limitée à une transaction (ce qui est le cas dans les données transactionnelles) ou à une durée dans le temps. Les associations entre événements peuvent donc être trouvées tout au long de la séquence, ce qui complexifie la tâche de fouille. Une fois ces associations trouvées (c'est-à-dire que le modèle est généré), elles peuvent être utilisées comme des règles (appelées *règles d'épisode*) pour prédire des événements futurs dans la séquence (Mannila et al., 1997). De plus, dans une séquence complexe, tout événement, même trivial (c'est-à-dire qui est peu fréquent) *a priori*, doit être étudié pendant le processus de modélisation, car il peut avoir un intérêt en association avec d'autres événements. Par conséquent, la complexité des algorithmes de modélisation d'une séquence complexe est très élevée (Pietsch, 2013). Nous appelons "séquence complexe" une séquence d'événements dont les données sont complexes, c'est-à-dire qu'elles ont la particularité d'être volumineuses, bruitées, variées, brutes,

2. Les signaux faibles sont les informations disponibles avec une fréquence d'apparition faible, qui représentent des indicateurs précoces d'un événement futur.

non structurées et véloces (voir définition de données complexes). La prise en compte de ces particularités rend la modélisation prédictive dans une séquence d'événements ardue et encore plus complexe. La modélisation prédictive dans une séquence complexe représente l'objectif de cette thèse.

1.2 Problématique

Lors de l'utilisation d'un modèle prédictif, des événements dits déclencheurs représentent les événements primordiaux de la prédiction. Nous les définissons somme suit :

Définition 1.2.1. *Les* **événements déclencheurs** *sont une combinaison d'événements, qui lorsqu'elle se produit, peut déclencher la prédiction d'un autre événement.*

Comme nous l'avons mentionné précédemment, nous partons de l'hypothèse que la prédiction au plus tôt est une solution pour mieux influencer et impacter les événements futurs car elle accorde un temps suffisant pour réagir suite à cette prédiction, c'est-à-dire suite à l'apparition des événements déclencheurs. Cependant, la modélisation dédiée à la prédiction au plus tôt et à l'influence sur des événements futurs dans une séquence comporte plusieurs défis, auxquels s'ajoute la difficulté d'une séquence complexe :

1. Dans une séquence complexe, les modèles prédictifs de l'état de l'art tendent à extraire les associations cachées (de type *règles d'épisodes*) composées d'événements rapprochés dans le temps (notamment pour des raisons de complexité des algorithmes). Lorsque ces associations sont exploitées, elles prédisent donc l'occurrence d'un événement proche, c'est-à-dire que son horizon d'apparition est proche et donc il apparaît peu de temps après qu'il ait été prédit (Luo and Bridges, 2000) (en particulier en raison de la vélocité de la séquence). Par conséquent, l'événement prédit apparaît avant que l'on puisse avoir le temps de réagir. Pour cette raison, la prédiction au plus tôt d'événements est de grand intérêt. De notre point de vue, prédire au plus tôt doit, dans la mesure de possible, permettre d'avoir un temps suffisant pour réagir avant l'apparition de l'événement prédit, si besoin. Par conséquent, il est intéressant de concevoir un modèle prédictif d'événements non seulement proches mais également distants, c'est-à-dire que l'horizon d'apparition de l'événement à prédire est éloigné temporellement. **La modélisation des événements à toute distance : événements proches et distants**, est une tâche très complexe. À notre connaissance, aucun modèle prédictif de l'état de l'art ne s'est intéressé à ce type de modélisation, car aucun modèle n'est capable de prédire des événements distants dans une séquence complexe, ce qui est, de notre point de vue, primordial pour une prédiction au plus tôt.
2. En raison de la variété et de la vélocité des événements dans une séquence complexe, de nouvelles associations peuvent apparaître au fil du temps. Par conséquent, un modèle conçu à partir d'une séquence (c'est-à-dire appris sur cette séquence à un moment donné) et utilisé pour prédire dans le futur, est limité car il ne recensera pas les nouvelles associations présentes dans la suite de cette séquence. Concevoir un modèle prédictif qui se met à jour régulièrement, en intégrant les nouvelles associations qui apparaissent, est une solution proposée dans l'état de l'art (Gaber et al., 2005). Dans ces modèles, une association devient candidate lorsqu'elle est apparue relativement fréquemment. Cependant, n'intégrer une nouvelle association qu'à partir du moment où elle est fréquente, limite sa possibilité d'utilisation en prédiction. En effet, en attendant qu'une nouvelle association soit fréquente, on se prive de la possibilité de l'utiliser en prédiction. Par conséquent, nous

considérons que **la détection des associations émergentes**, à savoir estimer qu'une nouvelle association sera fréquente avant qu'elle ne le soit effectivement, est une solution pour intégrer au plus tôt cette association dans le modèle et donc pour pouvoir rapidement bénéficier de son pouvoir prédictif. Prédire au plus tôt la fréquence élevée de l'association dans le futur est une tâche très complexe. À notre connaissance, cette tâche n'a jamais été traitée dans la littérature.

3. Comme nous l'avons mentionné précédemment, une fois qu'un événement est prédit, il est possible de réagir à cette prédiction en injectant un ou plusieurs événements dans la séquence de façon à impacter et à influencer l'occurrence de l'événement prédit. La plupart du temps, un expert du domaine connaît l'impact de certains événements sur d'autres événements (comme l'événement prédit), et peut ainsi déterminer le ou les événements adéquats à injecter. Pour des raisons de disponibilité d'experts et de rapidité, nous souhaitons automatiser cette tâche. À notre connaissance, aucun modèle de la littérature en fouille de données ne s'est intéressé à étudier l'impact d'événements sur d'autres événements. Certains modèles extraient les événements dits "importants" ou "risqués" qui portent des intérêts statistiques ou des caractéristiques prédéfinies, sans étudier leur impact sur d'autres événements. Par ailleurs, lorsque la prédiction d'un événement est réalisée dans une séquence complexe, une difficulté se présente en raison de la variété et de la vélocité des événements. En effet, un événement injecté, qui doit impacter et influencer un événement prédit, peut impacter de manière inattendue d'autres événements. Il peut également ne plus pouvoir impacter l'événement prédit car ce dernier peut devenir très proche entre temps voire apparaître. Par conséquent, **influencer automatiquement les événements futurs** est un grand défi dans une séquence complexe.

Dans cette thèse, nous nous intéressons donc à la problématique suivante : "**comment prédire au plus tôt et influencer l'apparition des événements futurs dans une séquence complexe ?**". Nous nous intéressons à cette question en nous fixant pour objectif de concevoir des modèles qui serviront à prédire et à influencer les événements futurs à partir d'une séquence complexe d'événements qui a les caractéristiques suivantes :

1. Variée : les données sont bruitées, brutes et non-structurées.
2. Volumineuse : la quantité de données générées est importante.
3. Véloce : les données arrivent rapidement et la séquence peut être sans limite.

1.3 Contributions

Les contributions apportées par nos travaux ont l'originalité de maîtriser l'horizon d'apparition des événements futurs (proches et distants) et sont les suivantes :

1. DEER (Distant and Essential Episode Rules) : un algorithme de fouille d'associations de type règles d'épisode pour la prédiction au plus tôt des événements futurs dans une séquence complexe.
2. EER (Emergent Episode Rules) : un algorithme pour la détection au plus tôt de l'émergence de nouvelles règles d'épisode dans un flux complexe d'événements.
3. IE (Influencer Events) : un algorithme pour la détection des événements "influenceurs" qui servent à influencer les événements futurs dans une séquence complexe.

Nous présentons maintenant brièvement ces contributions.

1.3.1 Un algorithme de fouille de règles d'épisode : DEER

Notre première contribution est DEER (Distant and Essential Episode Rules) (Fahed et al., 2015a; Fahed et al., 2014a; Fahed et al., 2014b; Fahed et al., 2014c) : un algorithme de fouille d'association de type règles d'épisode dans une séquence complexe d'événements. DEER modélise la séquence complète afin d'extraire des règles d'épisode avec deux caractéristiques. (i) les règles comportent une distance temporelle entre l'antécédent (le déclencheur) et la conséquence (l'événement futur), ce qui permet de préciser l'horizon d'apparition de cet dernier événement ; (ii) les règles sont essentielles : leur antécédent (le déclencheur) est minimal, c'est-à-dire le plus petit possible en terme de nombre d'événements et en durée temporelle. Par conséquent, les règles extraites permettront de se contenter d'une quantité minimale d'informations pour effectuer une prédiction au plus tôt d'un événement distant.

Par ailleurs, DEER diffère des algorithmes traditionnels d'extraction de règles d'épisode par le fait que le processus de construction de règles proposé a l'originalité de fixer la conséquence de la règle très tôt dans le processus, ce qui n'a jamais été fait dans la littérature. Les expérimentations menées montrent la pertinence de notre algorithme. Notamment, en comparant la performance de DEER avec un algorithme de la littérature de fouille de règle d'épisode (Mannila et al., 1997), nous montrons que DEER est efficace pour la prédiction au plus tôt et que notre processus de construction de règles favorise la diminution de la complexité en temps de DEER.

1.3.2 Un algorithme de détection de l'émergence de règles d'épisode : EER

Nous proposons l'algorithme EER (Emergent Episode Rules) : un algorithme pour la détection au plus tôt de l'émergence de nouvelles associations de type règles d'épisode dans un flux d'événements. Nous considérons un flux d'événements comme une séquence complexe d'événements dont le traitement est réalisé en temps réel.

Contrairement à l'état de l'art, nous partons de l'hypothèse que les nouvelles règles ne naissent pas du vide et ne sont pas totalement nouvelles, elles sont forcément liées ou influencées par d'autres règles déjà connues. Pour cela, nous considérons que les nouvelles règles qui tendent à apparaître et qui sont similaires à des règles connues, vont émerger dans le futur. Notre algorithme a l'originalité de traiter le flux d'événements en temps réel afin de capturer les premières apparitions de nouvelles règles d'épisode et de détecter au plus tôt l'émergence de ces nouvelles règles similaires aux règles connues (apparues fréquemment auparavant). EER diffère des algorithmes de la détection de l'émergence existants par le fait qu'il détecte l'émergence des règles (et non pas des événements uniques) à la fois inconnues mais également non encore fréquentes. EER s'exécute sur une plate-forme existante de traitement distribué de flux de données "STORM" (Apache Storm, 2012). Les expérimentations menées montrent la pertinence de notre algorithme pour la détection au plus tôt de l'émergence de nouvelles règles.

1.3.3 Un algorithme de détection d'événements influenceurs : IE

Concernant les événements futurs prédits, nous nous intéressons à les impacter et à les influencer pour pouvoir changer ce qui a été prédit. Nous proposons IE (Influencer Events) (Fahed et al., 2015b) : un algorithme pour la détection automatique d'événements que nous appelons "influenceurs" dans une séquence complexe d'événements. Notre algorithme modélise les associations entre les événements au travers de l'impact porté par certains événements sur d'autres événements. Cet algorithme a l'originalité de détecter les événements qui une fois injectés dans un contexte spécifique permettront d'influencer des événements futurs. Influencer ces événements est

effectué en impactant certaines de leurs caractéristiques : leur probabilité d'apparition (augmentation, diminution, mise à zéro), leur horizon d'apparition. Cette contribution diffère de l'état de l'art principalement par le fait qu'elle détecte automatiquement les événements influenceurs adéquats pour chaque contexte (comme influencer les achats d'un client particulier, le comportement d'un utilisateur particulier dans un blog, etc.) et non pas un ensemble d'événements influenceurs en général, applicable peu importe le contexte. Les expérimentations menées montrent la pertinence de nos modèles, notamment, en détectant des événements qui influencent au même temps plusieurs caractéristiques des événements futurs.

1.4 Application et corpus d'évaluation

Dans le domaine bancaire, les informations sur les clients sont exploitées afin de mieux connaître et comprendre ces clients. Nous partons du constat fait par les banques que les informations officielles dont elles disposent sur les clients, ne sont pas suffisantes ni même forcément sûres (cas d'absence d'un justificatif ou d'information non mise à jour), limitant ainsi la fiabilité de la modélisation. Exploiter d'autres données peut être utile afin d'améliorer cette modélisation, et de mieux connaître les clients : leurs besoins, satisfactions et opinions sur ses produits. En concertation avec le groupe Crédit Agricole S.A. qui finance cette thèse dans le cadre du projet "Armures", nous choisissons d'utiliser une source de données informelle : les réseaux sociaux et les blogs. Les réseaux sociaux et les blogs sont un environnement de partage des opinions et de diffusion de l'information, dans lequel les utilisateurs peuvent cacher leur identité réelle et donc s'exprimer librement et peuvent fournir des informations les concernant. Ainsi, nous faisons l'hypothèse que les réseaux sociaux et les blogs représentent la source contenant les informations dont nous avons besoin. Notre objectif dans cette thèse est d'exploiter, d'analyser les données issues des réseaux sociaux et des blogs et de traiter les besoins applicatifs suivants :

— Gérer la e-réputation : surveiller les discussions et détecter les cas où la banque doit intervenir.
— Extraire les nouveaux goûts, les nouveaux besoins et tendances des clients.
— Détecter les critères de défaut de remboursement et comment les éviter.

En atteignant ces objectifs, nous aidons le groupe Crédit Agricole S.A. à améliorer ces modèles afin de prendre des décisions et de réagir au bon moment.

Corpus de données : Afin d'évaluer les apports de nos contributions, nous avons mené des études expérimentales sur un corpus de données. Le corpus de données utilisé est composé de 27612 messages utilisateurs extraits à partir de deux blogs [3] [4] discutant de sujets financiers et bancaires. Il est important de noter que les messages sont ordonnés par leur date d'apparition. Les messages sont étiquetés en utilisant le logiciel *Temis* [5]. Ce logiciel utilise des cartouches d'étiquetage sémantique, syntaxique et d'opinion. Chaque message est donc représenté par l'ensemble de ses étiquettes (des items).

Prenons par exemple le message suivant : *"ma banque n'est pas la seule à proposer le mieux pour moi, mais aussi les banques en ligne. Dans ce cas, quel est le meilleur moyen pour améliorer mes économies ? avoir plusieurs comptes dans plusieurs banques ? avoir une assurance vie ? merci d'expliquer ça pour moi"*. Ce message est étiqueté avec les items suivants : *{(banque en ligne, négatif), (économie, neutre), (assurance vie, neutre), (besoin, neutre)}*, où chaque item

3. http ://cbanque.com
4. http ://droit-finances.commentcamarche.net
5. http ://www.temis.com

est associé à une polarité de sentiment, parmi trois valeurs : positive, négative et neutre. Un événement dans ce cas représente une idée exprimée dans un message. Il peut être composé d'un ou de plusieurs ensembles (item, sentiment). Ainsi, plusieurs événements peuvent représenter un seul message.

Dans ce corpus, les messages sont étiquetés par en moyenne $4,8$ items, allant de 1 à 50 et la médiane est 4. Il y a 4 480 étiquettes distinctes (items), avec une fréquence moyenne de $88,5$. Presque la moitié (49,4%) de ces items possède une fréquence égale à 1. Ces derniers items seront automatiquement éliminés pendant les phases d'intilialisation des l'algorithmes que nous proposons.

1.5 Présentation du plan de la thèse

Le reste du manuscrit est organisé comme suit : Dans le chapitre 2, nous présentons les concepts et les approches de la littérature liés à nos travaux. Puis, nous introduisons dans le chapitre 3 l'algorithme $DEER$ que nous proposons pour la fouille de règle d'épisode. Ensuite, nous détaillons dans le chapitre 4 l'algorithme EER pour la détection de règles d'épisode émergentes. Nous présentons dans le chapitre 5 l'algorithme IE dédié à la détection des événements influenceurs. Nous concluons ce manuscrit et présentons les perspectives dans le chapitre 6.

Chapitre 2

État de l'art

Sommaire

2.1 Principes de la fouille de données . . . 12
2.1.1 Introduction . . . 12
2.2 Les données simples . . . 13
2.2.1 Les transactions et les séquences . . . 13
2.2.2 Autres types de données : les graphes, les textes, etc. . . . 15
2.3 La fouille de données simples . . . 16
2.3.1 L'extraction de motifs et de règles d'association . . . 16
2.3.2 La classification . . . 20
2.3.3 Le clustering . . . 21
2.4 Vers des données complexes . . . 22
2.4.1 Quelques exemples de données complexes . . . 23
2.4.1.1 Les séquences d'événements . . . 23
2.4.1.2 Les graphes dynamiques . . . 23
2.4.1.3 Les textes bruts . . . 24
2.4.1.4 Les flux de données . . . 25
2.4.2 Les défis de la fouille de données complexes . . . 25
2.4.3 L'impact de données complexes sur la fouille de données : les nouvelles connaissances fouillées . . . 26
2.5 La fouille de séquences d'événements . . . 28
2.5.1 Importance et exemples d'applications . . . 29
2.5.2 Définitions . . . 29
2.5.3 Algorithmes d'extraction d'épisodes dans une séquence d'événements . 32
2.5.3.1 L'algorithme WINEPI . . . 32
2.5.3.2 L'algorithme MINEPI . . . 34
2.5.3.3 Les mesures de fréquence pour l'extraction d'épisodes . . . 36
2.5.4 Extraction des épisodes avec contraintes . . . 37
2.5.5 Extraction des règles d'épisode . . . 38
2.6 Prédiction dans une séquence d'événements . . . 39
2.6.1 Les approches existantes . . . 40
2.6.2 Discussion . . . 41
2.7 Détection de l'émergence dans un flux d'événements . . . 42
2.7.1 Les approches existantes . . . 42
2.7.1.1 La fouille de flux d'événements . . . 42

2.7.1.2 La détection d'émergence 44
2.7.2 Discussion 45
2.8 Influencer les événements futurs 45
2.8.1 Les approches existantes 46
2.8.2 Discussion 48
2.9 Plateformes de traitement de séquences et de flux 49

De nombreuses applications génèrent une grande quantité de données, qui ont la caractéristique d'être structurées et stables (à savoir homogènes au fil du temps). Nous appelons ces données des *données simples*. Il s'agit par exemple des séquences d'achats des clients de magasins, des séries de mesures météorologiques, du graphe de publications scientifiques, etc. Cependant, de plus en plus d'applications récentes génèrent des données non structurées et dynamiques (à savoir hétérogènes au fil du temps) qui ont également la caractéristique d'être extrêmement volumineuses, de l'ordre de téraoctets. Nous appelons ces données des *données complexes* (voir section 1.1). Les données complexes sont diverses et riches : elle portent énormément d'informations (Lohr, 2012). L'apparition de ces nouvelles données : leur diversité, leur volume, leur richesse, etc. rend le domaine de la fouille de données très dynamique, avec de nombreux défis à relever (Aggarwal, 2015).

Les séquences d'événements (voir section 1.1 du chapitre Introduction) sont un type de données largement fouillées dans l'état de l'art. Les séquences d'événements sont devenues, de notre point de vue, les données complexes les plus intéressantes, car elles portent énormément d'informations issues de différents domaines d'application comme le business, la sécurité, la médecine, etc. (Verhoef et al., 2004; Zhang et al., 2006). La fouille des séquences d'événements fait partie des thèmes de recherche d'actualité en fouille de données (Ao et al., 2015; Lin et al., 2015b). Plusieurs solutions ont été proposées dans l'état de l'art avec comme objectifs principaux la prédiction, l'extraction de connaissances, etc.

L'objectif de ce chapitre est tout d'abord de présenter, d'un point de vue global, les principes de la fouille de données : les données simples et les techniques standards de la fouille de données. Puis, nous nous intéressons aux données complexes et leur impact sur le domaine de la fouille de données. Nous nous focalisons ensuite sur la fouille d'un type spécifique de ces données : les séquences d'événements. Nous détaillons enfin les solutions proposées dans l'état de l'art pour la prédiction et l'influence des événements dans les séquences d'événements, qui sont au cœur de notre problématique.

2.1 Principes de la fouille de données

2.1.1 Introduction

La fouille de données est un domaine de l'informatique apparu au début des années 1990. Son objectif est d'extraire des connaissances à partir d'une grande quantité de données (Piateski and Frawley, 1991; Han et al., 1992). Certains chercheurs (Lakhmi and Wu, 2005) considèrent la fouille de données comme une sous-étape de l'extraction de connaissances à partir de données : Knowledge Discovery from Data (KDD). Cependant, d'autres (Aggarwal, 2015) considèrent la fouille de données comme un de ses synonymes. Nous adoptons ce dernier point de vue et nous nous inspirons de la définition de (Aggarwal, 2015) et définissons la fouille de données comme suit :

Définition 2.1.1. *La* **fouille de données** *est le processus qui consiste à collecter, nettoyer, traiter, analyser une grande quantité de données afin de découvrir des connaissances intéressantes,*

inattendues ou précieuses.

Par conséquent, la tâche de fouille de données peut être divisée en plusieurs étapes (Han et al., 2011; Aggarwal, 2015).

1. Pré-traitement des données : c'est l'étape de préparation de données qui consiste au nettoyage de données, à la récupération des données pertinentes et à la transformation des données sous la forme appropriée.
2. Modélisation des données : elle représente l'étape centrale du processus, qui consiste en l'application de méthodes intelligentes dites de fouille pour extraire les connaissances intéressantes, inattendues ou précieuses.
3. Post-traitement des données : c'est l'évaluation des connaissances extraites pour l'identification des connaissances les plus intéressantes et leur représentation en utilisant des outils de visualisation.

Il est important de noter que les étapes de pré-traitement et de post-traitement de données sont parfois divisées en plusieurs sous-étapes (Aggarwal, 2015). Cependant, nous avons choisi de représenter la tâche de fouille de données en seulement trois étapes principales pour montrer l'importance de l'étape de modélisation, puisque dans nos travaux nous nous focalisons uniquement sur cette étape.

2.2 Les données simples

De nombreuses données ont toujours été disponibles. Ces données portent des caractéristiques qui les rendent, de notre point de vue, simples. Nous définissons les données simples comme suit :

Définition 2.2.1. *Les* **données simples** *sont des données structurées et stables (à savoir homogènes au fil du temps), quelle que soit la quantité à la quelle elles sont présentes.*

Dans la littérature, on distingue plusieurs types de données simples. Nous pouvons citer par exemple les bases de transactions ou de séquences, les séquences d'événements, les séries temporelles, les graphes, les textes, etc. Dans les séries temporelles, par exemple, leur structure représentée par une séquence de valeurs d'une ou plusieurs variables, reste stable, c'est-à-dire qu'aucune nouvelle variable ne s'ajoute au fil du temps. Il est important de noter que plusieurs chercheurs (Aggarwal, 2015; Lakhmi and Wu, 2005) considèrent ces types de données comme des données complexes. Cependant, nous considérons que ces types ne portent pas toutes les caractéristiques que nous avons défini pour les données complexes, notamment la variété de données.

Nous allons présenter maintenant les différents types de données simples traditionnellement analysés dans la littérature. À noter que nous nous focalisons sur les notions de transactions et de séquences, ces dernières étant le cœur de nos travaux.

2.2.1 Les transactions et les séquences

La création des bases de transactions (Tung et al., 1999; Han et al., 2000b) a pour origine l'étude des habitudes de consommation des clients de magasins. Ce type d'étude est appelé l'étude du panier de la ménagère (market basket study). Dans ce cadre, les données se présentent sous la forme d'un ensemble de paniers d'achats pour chaque client. Un achat correspond à un produit acheté, un "item". Par conséquent, une transaction correspond aux achats d'un seul panier pour un seul client et est donc composée d'un ensemble d'items.

Définition 2.2.2. *Soit $I = \{i_1, i_2, ..., i_m\}$ un ensemble fini d'items. Une* **transaction** *est un ensemble non vide d'items notée $T = \{i_1, i_2, ..., i_k\}$ où $k \in \{1, ..., m\}$.*

Définition 2.2.3. *Un* **itemset** *est un ensemble d'items : $I_j = \{i_1, i_2, ..., i_j\}$. Lorsque l'itemset I_j est un sous-ensemble de l'itemset I_k, I_j est dit un sous-itemset de I_k : $I_j \subseteq I_k$.*

Il est important de noter que les notions de transaction et d'itemset, ont presque la même définition, mais souvent ont des rôles différents en fouille de données. Par exemple, on parle toujours de l'extraction d'itemsets et non pas de l'extraction de transactions (Aggarwal and Yu, 1998).

Dans les exemples que nous présenterons, un item sera présenté par une lettre minuscule. Un ensemble d'items, par exemple $\{f, g\}$, sera noté fg pour simplifier la notation.

Définition 2.2.4. *Une* **base de transactions** *est un ensemble de transactions notée $Base_T = \{T_1, T_2, ..., T_n\}$, où n est le nombre de transactions dans la base.*

Les séquences : En raison de l'évolution la nature des données dans de nombreuses applications, l'ordre des items dans une transaction peut être considéré (Agrawal et al., 1994; Zaki, 2001), et donc il est également possible d'exploiter l'ordre quand il est disponible. Considérons par exemple l'ordre des produits achetés par un client durant plusieurs sessions d'achat. Les données dans ce cas sont dites séquentielles. Nous distinguons trois types de données séquentielles : les bases de séquences, les séquences d'événements et les séries temporelles, que nous allons détailler dans les définitions suivantes. Un point en commun entre ces trois types est que l'ordre entre les items peut représenter une position temporelle (une date par exemple) ou une position spatiale. La contrainte de l'ordre imposée sur les items dans les séquences rend leur analyse beaucoup plus complexe par rapport à l'analyse des bases de transactions non séquentielles. Cela est dû au fait que les algorithmes de fouille doivent prendre en compte l'ordre des items, ce qui augmente leur complexité.

Nous allons maintenant définir, de manière formelle, la notion de séquence.

Définition 2.2.5. *Une* **séquence de données** *est une liste ordonnée d'itemsets : $S =< I_1, I_2, ..., I_n >$.*

À rappeler que l'ordre peut représenter une position temporelle ou spatiale. Pour un item, le temps de son apparition ou ses coordonnées spatiales sont présentes en général mais seul leur ordre est exploité pour former la séquence. Par exemple, dans un contexte de séquences d'achats, la séquence $< d, ab, c >$ est une séquence ordonnée par date croissante d'achat. À noter que l'ordre n'est pas présent entre entre a et b. Cela signifie que le client a acheté le produit d lors d'un premier passage au magasin, puis les produits a et b lors de son deuxième passage, et enfin le produit c lors d'un dernier passage.

Définition 2.2.6. *Soit $S1 =< a_1, a_2, ..., a_n >$ et $S2 =< b_1, b_2, ..., b_m >$ deux séquences de données. S_1 est une* **sous-séquence** *de S_2 (noté $S_1 < S_2$) s'il existe des entiers $1 \leq i_1 < i_2 < ... < i_n \leq m$ tels que $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, ..., a_n \subseteq b_{i_n}$.*

Définition 2.2.7. *Une* **base de séquences** *est un ensemble de séquences notée $Base_S = \{S_1, S_2, ..., S_n\}$.*

Les séquences d'événements : Les séquences dans une base de séquences sont généralement courtes (Leleu et al., 2004). Cependant, plusieurs applications génèrent un autre type de séquences : les séquences d'événements (Mannila et al., 1997). Une séquence d'événements est

une longue séquence qui peut s'étendre sur plusieurs dizaines voire plusieurs milliers d'unités temporelles (instants ou *timestamps*) sur lesquelles sont apparus plusieurs milliers d'événements (itemsets) (Méger and Rigotti, 2004). Contrairement aux séquences, où le temps d'apparition d'un item n'est pas exploité, dans les séquences d'événements, la notion de temps concret est généralement utilisée et associée à chaque événement.

Nous allons présenter maintenant, de manière plus formelle, les définitions liées aux séquences d'événements.

Définition 2.2.8. *Soit $I_t \subseteq I$ l'ensemble d'items qui apparaît à l'instant t (au timestamp t), on dit que I_t représente un* **événement**.

Définition 2.2.9. *Une* **séquence d'événements** *S est une liste ordonnée de couples (événement, instant), $S =< (t_1, I_{t_1}), (t_2, I_{t_2}), ..., (t_n, I_{t_n}) >$ avec $t_1 < t_2 < ... < t_n$.*

Parmi les exemples de séquences d'événements, nous pouvons citer une séquence d'achats en ligne (qui représente l'historique d'achats et qui diffère de transactions d'achats par la possibilité de tracer l'ordre d'items dans le panier) (Korzaan, 2003), une séquence de clicks sur un site Web (Gündüz and Özsu, 2003), une séquence de processus dans une chaîne de production (Ho et al., 2006), etc. Dans cette thèse, nous nous intéressons particulièrement à ce type de séquences. Un chapitre complet sera consacré à l'analyse des séquences d'événements.

Les séries temporelles : Un troisième type de données séquentielles peut être généré par plusieurs applications : les séries temporelles. Une série temporelle est une séquence de valeurs métriques enregistrées à des intervalles de temps égaux (chaque minute, heure ou jour, etc.) (Geweke and Porter-Hudak, 1983). La représentation des séries temporelles est différente de la représentation des bases de séquences et des séquences d'événements, car elle est basée sur les valeurs de mesure d'une seule variable. Par exemple, la variable peut être une variable en météorologie, qui peut représenter la valeur de température, d'humidité, ou de pression.

Définition 2.2.10. *Soit X une variable qui représente une ou plusieurs mesures : $X = (x_1, x_2, ...x_m)$. Les valeurs de la variable X à l'instant t sont notées X_t. Une* **série temporelle** *S est une suite ordonnée des valeurs de mesures d'une seule variable X, notée $S =< X_{t_1}, X_{t_2}, ...X_{t_n} >$.*

Les séries temporelles sont très utilisées dans le domaine de l'analyse et de la prévision économique (Granger and Newbold, 2014), le contrôle de système de télécommunications (Mastorocostas et al., 2012), le contrôle de qualité d'un système (Štěpánek et al., 2011), etc.

En plus des trois types de données séquentielles déjà mentionnés, plusieurs chercheurs (Han et al., 2011) ajoutent un quatrième type : les **séquences biologiques**. Une séquence biologique est une séquence de protéines ou d'acides aminés (Andreeva et al., 2014; Srinivasan et al., 2013). Ces séquences portent un sens sémantique caché très complexe et très important. Analyser ces séquences est incontournable pour comprendre les structures et les fonctions de diverses molécules et donc pour mieux comprendre et traiter les maladies (Durbin et al., 1998). Cette analyse est cependant complexe en raison de la diversité et du manque d'ontologies standards dans ces séquences (Raza, 2012).

2.2.2 Autres types de données : les graphes, les textes, etc.

En dehors des transactions et des séquences, il existe plusieurs autres types de données simples comme les graphes et les textes.

Un graphe est une structure de données plus générale que les séquences, dans laquelle les données sont représentées par des sommets, et les relations entre les données par des liens : soit

des arcs (pour les graphes orientés) ou des arêtes (pour les graphes non orientés). On distingue deux types de graphes : (i) les graphes homogènes : dans lesquels les nœuds sont de même type (de même pour les liens), citons par exemple le graphe d'un utilisateur et de ses amis dans un réseau : un utilisateur représente un nœud et l'amitié entre deux utilisateurs représente une arête dans le graphe ; (ii) les graphes hétérogènes : dans lesquels les nœuds sont de différents types (de même pour les liens), citons par exemple les graphes de publications scientifiques où un auteur ou un article représente un nœud, et une citation représente une arête. Les applications de graphes sont très nombreuses. Nous pouvons citer par exemple les graphes de systèmes d'informations (Ghosh et al., 2014), les réseaux multimédia (O'Callaghan et al., 2012), etc.

Un autre type de données simples sont les textes. Ils représentent un type de données très classique (Tan et al., 1999). Nous distinguons deux types de données textuelles : (i) les tableaux de données : c'est un type de textes structurés dans lequel une donnée peut correspondre à un produit, une personne, un objet, etc. Chaque donnée (ou instance) est décrite par un certain nombre d'attributs ; (ii) les textes semi-structurés : représentés par les documents XML, qui sont moins structurés qu'un tableau.

Dans la catégorie de données simples, nous pouvons également citer les images (Lowe, 1999), les données spatiales (Samet, 1990), les données spatio-temporelles (Erwig et al., 1999), etc.

Nous allons maintenant présenter les techniques traditionnelles de fouille de données simples.

2.3 La fouille de données simples

Comme nous l'avons mentionné, la modélisation de données représente l'étape fondamentale du processus de fouille de données, et celle sur laquelle la recherche scientifique focalise et avance en continu. On distingue deux grands types de modélisation : descriptive et prédictive. Plusieurs techniques ont été proposées dans la littérature afin de réaliser les deux types de modélisation. Il est connu que quelle que soit la technique de modélisation utilisée, elle peut être utilisée pour des objectifs à la fois descriptifs et prédictifs. Nous nous intéressons maintenant aux techniques de fouille de données standard : (i) l'extraction de motifs et de règles d'association, (ii) la classification et (iii) le clustering. Nous détaillons plus précisément la première technique et la manière dont elle est appliquée sur des données simples, ce qui nous sert d'inspiration pour la fouille de données complexes.

2.3.1 L'extraction de motifs et de règles d'association

L'extraction de motifs et de règles d'association (Agrawal et al., 1993; Srikant et al., 1997) est un domaine classique dans la fouille de données. Cette tâche s'applique généralement sur les bases de transactions et les données séquentielles. Nous allons définir dans cette section les notions liées à l'extraction de motifs et de règles d'association ainsi que les familles d'algorithmes dédiées à la réalisation de cette tâche. À noter que dans ce chapitre, nous ne présentons pas la fouille de séquences d'événements représentée par l'extraction d'épisodes et par l'extraction de règles d'épisode. Nous consacrons un chapitre complet à la fouille des séquences d'événements, qui représente le cœur de cette thèse.

Définissions maintenant les objets recherchés dans une base de transactions ou de séquences : les motifs, et les notions associées.

Un motif M a la même définition qu'une transaction et qu'un itemset, c'est-à-dire un ensemble d'items. Cependant, le rôle des motifs en fouille de données est différent : ils représentent les connaissances fouillées et on s'intéresse à leur fréquence d'apparition.

Définition 2.3.1. *Soit M un motif. Quand l'ordre d'apparition des items est considéré, le motif est appelé un* **motif séquentiel** *(sequential pattern). Le premier item i_1 d'un motif séquentiel représente un préfixe et le dernier item i_m est un suffixe.*

Définition 2.3.2. Une occurrence *d'un motif signifie une apparition de ce motif dans une transaction ou dans une séquence. Soit M un motif et T une transaction (ou S une séquence dans le cas d'une base de séquences). Lorsque M est un sous-ensemble de T : $M \subseteq T$ (ou une sous-séquence de S : $M \subseteq S$), nous considérons que le motif M apparaît dans la transaction (ou dans la séquence) : une occurence de ce motif est trouvée. Le* **support** *de M, noté $supp(M)$ représente le nombre de transactions (ou de séquences) dans lesquelles il apparaît (le nombre de ses occurrences).*

Définition 2.3.3. *Soit M un motif et $supp(M)$ son support. Si le motif M satisfait la fréquence minimum exigée, il est appelé* **motif fréquent** *: si $supp(M) \geq minsupp$, où $minsupp$ est le seuil minimum donné du support.*

L'ensemble de techniques d'extraction de motifs s'appuie sur la notion de support de motifs afin de n'extraire que les motifs fréquents. Cette notion de support représente une contrainte sur les motifs. L'application de cette contrainte de support est nécessaire car elle permet de réduire l'espace de recherche des motifs, et ainsi de limiter la consommation de ressources en temps et en mémoire. Dans la fouille de motifs, une caractéristique exigée lors de l'évaluation du support est l'anti-monotonicité (Agrawal et al., 1994).

Définition 2.3.4. *De manière générale,* **l'anti-monotonicité** *(Uryas'ev, 1988) est définie comme suit : si un élément (ou un objet) viole une contrainte, donc tous ses sur-éléments (ou sur-objets) violent la même contrainte.*

Le support des motifs doit être anti-monotone (Agrawal et al., 1994) : si le motif $< a, b >$ par exemple n'est pas fréquent, aucun de ses sur-motifs ne peut être fréquent. Ainsi, les algorithmes de fouille ne considéreront aucun de ces sur-motifs pendant la recherche des motifs, ce qui représente un gain de temps important. En plus de la contrainte de support, les techniques de recherche des motifs ont souvent un point commun relatif à l'utilisation de deux phases : une phase de génération de candidats (construction des motifs potentiellement fréquents dans la base) et une phase d'évaluation de la fréquence des candidats (application de la contrainte de support afin de savoir si les motifs candidats sont fréquents ou non). Nous allons maintenant présenter en détails les techniques standards pour l'extraction de motifs.

Les différentes familles d'algorithmes : Deux familles d'algorithmes de recherche de motifs sont connues dans la littérature, savoir les algorithmes basés sur le principe de recherche en largeur d'abord et les algorithmes basés sur le principe de recherche en profondeur d'abord. Nous allons introduire ces deux familles d'algorithmes en présentant leur principe général et des exemples d'algorithmes connus.

— **Méthodes basées sur le principe de recherche en largeur d'abord (breadth-first)** : la première famille d'algorithmes d'extraction de motifs suit le modèle de l'algorithme **Apriori** (Agrawal et al., 1994) et adopte une approche pour explorer l'espace de recherche des motifs : la recherche en largeur d'abord (breadth-first search). Dans la recherche en largeur, les motifs de taille $m+1$ ne sont considérés qu'après avoir exploré tous les motifs de taille m (la construction de motifs se fait de motifs de taille plus petite vers les motifs de taille plus grande). Par conséquent, aucun motif fréquent ne peut contenir un sous-motif qui n'est pas fréquent.

Apriori est le premier algorithme de recherche de motifs dans une base de transactions. Depuis l'apparition d'Apriori, de nombreux algorithmes d'extraction de motifs utilisent ce même principe. Les algorithmes de recherche de motifs en largeur d'abord suivent les quatre étapes suivantes :

1. Phase d'initialisation : un passage complet sur la base est effectué afin d'évaluer le support de tous les items i de I (motifs de taille $m = 1$). Les items dont le support est supérieur au support minimum $minsupp$ sont retenus pour former l'ensemble de motifs fréquents de taille 1. À l'issue de cette phase, commence l'itération basée sur la succession des phases de génération de candidats et d'évaluation du support.
2. La génération de candidats : cette phase permet de créer des motifs candidats de taille $m+1$ à partir de l'ensemble de motifs fréquents de taille m. Soit M_m, M'_m deux motifs fréquents de taille m. Si $|M_m \cap M'_m| = m - 1$, le motif issu de la jointure de M_m et M'_m est un motif candidat de taille $m + 1$.
3. Évaluation du support des motifs candidats : un passage sur la base est effectué et à chaque fois que le processus détermine qu'une transaction (ou une séquence) contient un motif candidat, le support de ce motif est incrémenté de 1.
4. L'ensemble de motifs fréquents est augmenté par la sélection de motifs fréquents à la fin de chaque itération. L'itération s'arrête quand aucun motif candidat n'est fréquent.

Concernant l'extraction de motifs séquentiels, les mêmes phases sont appliquées en ajoutant une contrainte d'ordre : l'ordre des items doit être respecté pendant la phase de génération de candidats et la phase d'évaluation du support. La contrainte de l'ordre augmente la complexité algorithmique (dans les étapes 2 et 3) et rend la tâche d'extraction de motifs séquentiels difficile. Pour cette raison, la recherche est très dynamique dans ce domaine. Nous allons présenter quelques algorithmes connus pour l'extraction de motifs séquentiels.

La première proposition dans l'état de l'art pour extraire les motifs séquentiels est l'algorithme **GSP** (Generalized Sequential Patterns) (Srikant and Agrawal, 1996). GSP est également basé sur la recherche en largeur d'abord. Les apports de GSP par rapport à l'algorithme Apriori reposent sur deux idées : (i) la génération de candidats : deux motifs M, M' de taille m peuvent générer un motif de taille $m + 1$, si M sans son préfixe est identique au M' sans son suffixe. Par exemple, les motifs $< c, a >$ et $< a, b >$ permettent de générer le motif candidat $< c, a, b >$. De la même manière, les motifs $< c, a, d >$ et $< a, d, e >$ permettant de générer le motif candidat $< c, a, d, e >$; (ii) la structure utilisée pour représenter les candidats dans la base de séquences est un arbre de hachage pour réduire le nombre de motifs candidats.

En utilisant également le principe de parcours de l'espace de recherche en largeur d'abord, l'algorithme **SPADE** (Zaki, 2001) élague l'espace de recherche en regroupant les candidats par catégorie. SPADE est connu pour sa gestion de données en mémoire : les motifs fréquents ayant leurs premiers items identiques, ce qui permet de décomposer le problème de recherche de motifs en sous-problèmes qui seront traités parallèlement en mémoire.

L'avantage de méthodes basées sur le principe de recherche en largeur d'abord est qu'elles garantissent que l'ensemble final de motifs fréquents est complet (aucun motif fréquent non extrait). Cependant, plusieurs limitations sont présentes. Un très grand nombre de candidats peut être généré, et chaque phase d'évaluation (pour calculer le support des motifs candidats) déclenche un parcours de la base complète. Par conséquent, les ressources en temps et en mémoire peuvent exploser avant même de construire les motifs les plus longs (Aggarwal and Han, 2014). Ces algorithmes, comme tous les algorithmes

basés sur Apriori, ont des difficultés à parcourir de longues séquences. En effet, les longues séquences sont formées à partir d'un nombre important d'itemsets et le nombre de motifs candidats générés augmente de façon exponentielle avec la longueur de ces itemsets.

— **Méthodes basées sur le principe de recherche en profondeur d'abord (depth-first search)** : Afin de traiter les limitations présentes dans les algorithmes basés sur la recherche en largeur d'abord, des méthodes basées sur le principe de recherche en profondeur d'abord ont été proposées. L'idée principale est de faire grandir un motif autant que possible jusqu'à ce que le support de ce motif ne satisfasse plus le seuil de support minimum (Agarwal et al., 2000; Aggarwal and Han, 2014). Contrairement aux méthodes en largeur d'abord, il n'existe pas de phase standard à suivre pour les méthodes de recherche de motifs en profondeur. Cependant, de nombreuses méthodes utilisent une structure en arbre de manière à ce que tout chemin de la racine à une feuille représente un motif candidat.
Parmi les algorithmes connus pour le parcours en profondeur d'abord, nous pouvons citer l'algorithme **PSP** (Prefix Tree for Sequential Pattern) (Masseglia et al., 1998). PSP met en place et exploite un arbre de préfixes pour gérer les candidats. C'est aussi le principe adopté par FreeSpan (Han et al., 2000a) et amélioré par PrefixSpan (Han et al., 2001).
L'algorithme **FreeSpan** (Frequent pattern-projected Sequential pattern mining) (Han et al., 2000a) propose la projection (le partitionnement) de la base de données en fonction des items fréquents. La base est alors projetée en plusieurs bases plus petites et les motifs fréquents grandissent avec le nombre de projections. Le temps de réponse est donc diminué car chaque base projetée est plus petite et facile traiter. FreeSpan a tout de même un défaut : une sous-séquence peut être générée par n'importe quelle combinaison dans une séquence, donc l'algorithme doit conserver la totalité de la séquence dans la base d'origine sans réduire sa taille.
Prefixspan (Han et al., 2001) propose d'analyser les préfixes communs que présentent les séquences dans la base de données. À partir de cette analyse, l'algorithme construit des bases de données intermédiaires qui sont des projections de la base d'origine déduites à partir des préfixes identifiés. Ensuite, dans chaque base obtenue, PrefixSpan cherche à faire croître la taille des motifs découverts en appliquant la même méthode de manière récursive. PrefixSpan est parmi les algorithmes les plus connus dans la recherche de motifs.
Les algorithmes basés sur la recherche en profondeur d'abord comportent tout de même des inconvénients. Lorsque le nombre de motifs est très grand, l'algorithme aura des difficultés en terme de temps pour finir le processus de fouille sur une branche de l'arbre. Pour cela, des contraintes sont nécessaires pour arrêter le processus (Stone and Sipala, 1986). Encore plus, lors de la recherche de l'occurrence d'un motif, le processus peut être infini si l'arbre est très profond.

L'extraction de règles d'association : Dans de nombreuses applications, l'extraction de motifs fréquents est suffisante pour répondre aux objectifs applicatifs, donc il n'y a pas de nécessité de générer les règles d'association. Cependant, dans d'autres applications, notamment celles ayant un objectif de prédiction, les règles d'association sont incontournables. Une règle d'association $R : M \rightarrow N$ est une implication de type : si un élément M (appelé antécédent) apparaît, alors un autre élément N (appelé conséquence) apparaît avec un probabilité relativement élevée. Les règles d'association sont souvent utilisées pour des objectifs de prédiction (Yin and Han, 2003). Par exemple, dans les magasins, les informations sur les achats des clients sont sauvegardées. Les commerçants sont intéressés par la prédiction de ce que les clients vont acheter

(la conséquence de la règle). Ces informations servent à bien mener les campagnes marketing et à améliorer les relations clients par l'exploitation des règles d'association.

La génération de règles d'association à partir de motifs fréquents est une tâche simple dans la littérature et se fait traditionnellement en deux étapes (Agrawal et al., 1994). La première consiste à extraire l'ensemble des motifs fréquents. La deuxième a pour objectif de construire les règles à partir de ces motifs : identifier l'antécédent et la conséquence.

Définition 2.3.5. *Soit F un motif. $\forall M, N : F = M \cup N$, la règle $R : M \rightarrow N$ est une* **règle d'association** *candidate avec $supp(R) = supp(F)$. Le motif M représente l'antécédent de la règle et le motif N représente sa conséquence.*

Définition 2.3.6. *Soit $R : M \rightarrow N$ une règle d'association. La* **confiance** *de R notée $conf(R : M \rightarrow N)$ représente la probabilité d'apparition de la conséquence N de la règle R sachant que l'antécédent M est apparu. La mesure de confiance est représentée par l'équation suivante :*

$$conf(R : M \rightarrow N) = \frac{supp(M \cup N)}{supp(M)} \tag{2.1}$$

Si la confiance de la règle dépasse un seuil minimum $minconf$, la règle est dite une règle confiante. Généralement, seules les règles dites confiantes sont recherchées.

Notons que pour les motifs séquentiels, $F = M \cdot N$, qui considèrent l'ordre entre les items, la confiance représente la probabilité d'apparition de la conséquence N de la règle R après l'antécédent M sachant que ce dernier est apparu.

Selon plusieurs travaux (Liu, 2011), toutes les informations nécessaires pour le calcul de la confiance d'une règle sont disponibles en amont : le motif $F = M \cup N$ à partir duquel la règle $R : M \rightarrow N$ est construite est un motif fréquent, et par le critère d'anti-monotonicité, tous ses sous-motifs sont fréquents et leur support a été calculé pendant le processus d'extraction de motifs (donc on l'a en mémoire). Ceci rend la tâche de génération des règles d'association assez simple.

Les auteurs de l'algorithme Apriori proposent un algorithme pour la génération des règles d'association souvent utilisée pour cette tâche : **GenRules** (Agrawal et al., 1994). Le principe de cet algorithme est le suivant : pour chaque motif fréquent F, l'algorithme commence à construire les règles d'association dans lesquelles les conséquences (les motifs N) sont composés d'un seul item. Puis, une fois une règle d'association validée (confiante), l'itération de l'augmentation de la taille de la conséquence commence : un item de l'antécédent est déplacé pour qu'il fasse partie de la conséquence, et la nouvelle règle d'association est évaluée par la mesure de confiance afin de tester sa validité.

2.3.2 La classification

La classification (qui rentre dans la catégorie de l'apprentissage supervisé) est le processus qui consiste à trouver un modèle ou une fonction qui distingue les catégories (appelées étiquettes) de classes présentes dans les données, à partir des attributs qui caractérisent ces données (Dietterich, 1998). Généralement, la classification est réalisée en deux étapes. On dispose au départ d'un échantillon de données dit exemples d'apprentissage dont les catégories (étiquettes) sont connues (c'est pour cette raison que la classification rentre dans le cadre d'apprentissage supervisé). Cet échantillon est utilisé pour l'apprentissage du modèle. Comme tout algorithme d'apprentissage automatique, il est nécessaire ensuite d'étudier la qualité du modèle appris. Pour cela, on utilise

souvent un deuxième échantillon indépendant, dit de validation ou de test. Le modèle construit, connu sous le nom de classifieur, est ensuite utilisé pour classifier les nouvelles données.

Présentons ces concepts de manière plus formelle :

Définition 2.3.7. *Soit BD une base qui contient des objets : des images, des transactions, des séquences, des graphes, etc. Soit $O = \{o_1, o_2, ...o_m\}$ l'ensemble des objets dans BD. Chaque objet est représenté par une description : l'ensemble des valeurs de ses attributs. Pour l'objet o_i, sa description est $\{d_{i_1}, d_{i_2}, ...d_{i_m}\}$. Soit $C = C_1, C_2, ..., C_c$ l'ensemble des étiquettes des classes. La classification supervisée définit la fonction suivante : chaque objet o_i est associé à une étiquette de classe C_k. Cette fonction est notée $C^f(o_i, C_k)$. La* **classification supervisée** *consiste alors à déterminer une procédure de classification : à partir de la description de l'objet, la procédure détermine l'étiquette de la classe de l'objet : $C^f : \{d_{i_1}, d_{i_2}, ...d_{i_m}\} \rightarrow C_k$.*

Prenons un exemple dans le domaine bancaire. Un banquier veut pouvoir prédire si son client sera capable de rembourser son prêt. Pour cela il collecte des informations sur son client : âge, profession, revenus, etc. Puis, en fonction de ces observations, il accepte ou refuse l'octroi du prêt. La règle d'octroi du prêt est apprise au préalable à partir des clients dont les caractéristiques ainsi que l'étiquette "prêt remboursé" sont connues. En utilisant la fonction de classification, un nouveau client est classé dans une des deux classes : apte à rembourser ou ne pas rembourser le prêt.

Quelle que soit la méthode de classification, un problème important peut influencer la qualité du modèle : le manque de données disponibles pour réaliser l'apprentissage ou la présence de données inadéquates (incertaines ou imprécises), qui empêche la construction d'un modèle fiable. Dans l'état de l'art, nous pouvons distinguer plusieurs familles de méthodes de classification. Parmi les familles de méthodes les plus connues nous citons les arbres de décision (Quinlan, 1986), les réseaux de neurones (Wan, 1989), les machines à support vecteurs (SVM) (Vapnik and Vapnik, 1998), les classificateurs de Bayes (Hanson et al., 1991), etc.

2.3.3 Le clustering

Le clustering (qui rentre dans la catégorie de l'apprentissage non supervisé) est le processus qui consiste à diviser automatiquement les données en groupes ou en classes (appelés clusters). L'ensemble de clusters issus d'un seul processus représente un clustering ou un schéma de clustering (Jain and Dubes, 1988). Contrairement à la technique de classification, un modèle de clustering est appris à partir de données non étiquetées (à savoir les informations sur les catégories de classes ne sont pas connues).

Les objets (les éléments) de chaque cluster sont similaires les uns aux autres, et non similaires aux éléments d'un autre cluster en se basant sur une mesure de distance (Han et al., 2011). Par conséquent, le choix de la mesure de distance ou de similarité est une question primordiale pour les méthodes de clustering car elle influence la nature et la qualité du clustering. À noter que chaque domaine d'application possédant ses propres données, il peut posséder également sa propre mesure de distance ou de similarité. Il faut concevoir alors une mesure différente pour chaque domaine d'application ou chaque type de données, permettant de distinguer au mieux les différences entre les données.

Présentons la tâche de clustering de manière plus formelle :

Définition 2.3.8. *Soit BD défini dans la section précédente (voir section 2.3.2 définition 2.3.7), la classification non supervisée (le clustering) définit la fonction suivante : chaque paire d'objets o_i, o_j est associée à une distance : $(o_1, o_2, dist(o_1, o_2))$, où "dist" est la fonction de distance.*

La **classification non supervisée (clustering)** *consiste alors à déterminer une procédure de regroupement des objets : à partir de la description d'un sous-ensemble d'objets O' et de la fonction de distance, la procédure forme plusieurs clusters, comme le cluster C_i de la manière suivante :* $C^f : O', dist \rightarrow C_i : (\forall (o_i, o_j) \in O' : dist(o_i, o_j)$ *est minimale*$) \wedge (\exists\, O'' \subseteq O : \forall o_k \in O'' : dist(o_i, o_k)$ *est maximale).*

Prenons à nouveau l'exemple du domaine bancaire. Plusieurs observations peuvent être collectées sur les clients, comme par exemple la nature des transactions bancaires. Il existe des profils-type de clients. L'objectif est alors d'une part de découvrir ces profils-type (par la tâche de clustering) à partir de l'information sur les transactions bancaires des clients, et d'autre part de déterminer, pour chaque client, à quel profil il appartient.

Dans la littérature, plusieurs méthodes ont été proposées pour le clustering. Nous distinguons trois types principaux de méthodes : (i) les méthodes de clustering hiérarchique (basées sur la construction d'arbres) (Kaufman and Rousseeuw, 2009), (ii) les méthodes de clustering par partitionnement (basées sur la sélection du meilleur schéma de clustering en comparant plusieurs schémas candidats), comme l'algorithme des k-moyennes (k-means) (MacQueen et al., 1967) qui est sans doute la méthode de partitionnement la plus connue et la plus utilisée dans divers domaines d'application ; (iii) les méthodes probabilistes (basées sur l'estimation de la densité de probabilité) (Dempster et al., 1977).

Il est important de mentionner que les techniques de classification et de clustering, que nous avons brièvement présentées, ne représentent pas le cœur de notre travail. C'est pour cette raison que nous ne les détaillerons pas.

Nous avons présenté dans cette section les techniques de fouilles de données simples. Passons maintenant à celles de données complexes.

2.4 Vers des données complexes

Comme nous l'avons mentionné dans l'introduction de ce manuscrit (voir section 1.1 page 3), les données actuellement générées ne sont en général plus des données simples, elles sont disponibles en immense quantité et sous de nouvelles formes : non structurées et parfois en structures multiples. Par ailleurs, elles portent de nouvelles caractéristiques : vélocité, bruit, etc. Nous choisissons de les appeler données complexes. De notre point de vue, la généralisation des données issues du Web est la raison principale de l'apparition et de la prédominance des données complexes. Au sein du Web (en considérant le contenu, les liens, etc.), nous pouvons collecter plusieurs catégories de données : des séquences, des graphes, des textes, etc. chacune ayant des caractéristiques particulières (Kosala and Blockeel, 2000). C'est pourquoi, la majorité des exemples que nous citons sont issus du Web.

Présentons maintenant la définition de données complexes (introduit de façon informelle dans le chapitre d'introduction de ce manuscrit).

Définition 2.4.1. *Les* **données complexes** *sont des données qui portent au moins une des trois caractéristiques suivantes : (i) très volumineuses : il y a une immense quantité de données générées, (ii) véloces : la fréquence à laquelle les données sont générées, capturées et partagées est très élevée, (iii) variées : les données sont bruitées, brutes et non structurées (des mots, images, vidéos, séquence de signaux de capteurs, etc.) et provenant potentiellement de plusieurs sources.*

Comme nous l'avons mentionné, les données complexes ont un grand pouvoir prédictif en raison de leur richesse en information (Lohr, 2012). Par conséquent, de nombreuses applications exigent désormais l'extraction de connaissances de plus en plus sophistiquées et évoluées au sein

des données complexes. Ainsi, la fouille de données complexes devient une tâche de plus en plus importante en fouille de données.

Dans cette section, nous allons tout d'abord présenter quelques exemples de données complexes ou de données devenues complexes. Ensuite, nous allons introduire les défis de la fouille de données complexes, ainsi que les nouvelles connaissances fouillées.

2.4.1 Quelques exemples de données complexes

Un constat a récemment été fait que plusieurs types de données simples sont devenus complexes et que de nouveaux types sont apparus en tant que données complexes. Il est également important de mentionner qu'un ensemble de données simples peut former des données complexes, car dans ce cas, l'ensemble de données devient hétérogène avec des structures multiples. Prenons l'exemple d'un ensemble "texte-image-vidéo" qui constitue, de notre point de vue, un type fréquent de données complexes. Ces données sont aisément disponibles sur le Web.

Nous allons maintenant présenter les types de données complexes les plus étudiés dans la littérature.

2.4.1.1 Les séquences d'événements

Récemment, les séquences d'événements (voir définition 2.2.9 page 15) ont changé de caractéristiques, principalement en raison du changement de caractéristiques des événements les constituant. Avant, les séquences étaient structurées et stables, désormais, les événements les constituant : (i) ne sont plus un simple ensemble d'items (structurés), ils ont désormais des formes non structurées (comme par exemple le cas d'un événement représenté par une phrase écrite par un utilisateur. La séquence devient alors une séquence de phrases) ; (ii) de plus en plus d'événements sont générés et la séquence devient donc de plus en plus longue, ce qui augmente son volume ; (iii) les événements provenant des sources multiples, n'ont pas tous la même structure. Pour ces raisons, nous considérons que les séquences d'événements deviennent des données complexes et les algorithmes conçus pour la fouille de séquences d'événements simples ne sont plus adaptés. Nous nous intéressons, dans cette thèse, à la fouille de séquences d'événements complexes. Nous allons détailler les algorithmes de la fouille de ces séquences dans une section ultérieure (section 2.5 page 28).

2.4.1.2 Les graphes dynamiques

Comme nous l'avons mentionné (voir section 2.2.2 page 15), nous considérons les graphes comme un type de données simples. Cependant, un type récent de graphes est apparu : les graphes dynamiques, dits également évolutifs (en opposition aux graphes statiques) (Likhachev et al., 2008; Beck et al., 2014). Dans un graphe dynamique, la structure du graphe change au fil du temps : des arêtes et/ou des sommets disparaissent, apparaissent, etc. (Cortes et al., 2012; Desmier et al., 2013). Nous pouvons citer l'exemple d'un graphe d'utilisateurs dans un réseau social, où les liens d'amitié changent au fil du temps. Ce changement permanent de structure est la raison pour laquelle nous considérons les graphes dynamiques comme des données complexes.

Définition 2.4.2. *Un* **graphe dynamique** *est une paire $G = (V, E)$, où V est une fonction $V(t)$ qui associe à tout instant t un ensemble de sommets, et E est une fonction $E(t)$ qui associe à tout instant t un ensemble d'arêtes.*

L'objectif de la fouille de graphes dynamiques est de comprendre les raisons de leur évolution, de prédire cette évolution, de l'empêcher, etc. Les modèles de fouille conçus pour les

graphes non dynamiques (ou statiques) ne sont pas adaptés car, entre autres, ils ne gèrent pas l'aspect dynamique (Ilcinkas et al., 2014). Même les modèles ayant une certaine tolérance aux fautes (c'est-à-dire aux légers changements de structure) deviennent insuffisants : les graphes dynamiques changent en continu, les modèles doivent être adaptés pour capturer un changement dès qu'il se produit (Ilcinkas et al., 2014). Pour cela, des algorithmes dits dynamiques sont proposés (Nikolopoulos et al., 2012; Zakrzewska and Bader, 2015). Un algorithme dynamique considère un graphe dynamique comme étant une suite de graphes statiques. Cet algorithme commence par la détection d'une ou plusieurs propriétés évolutives du graphe (des caractéristiques qui changent au fil du temps). L'évolution des valeurs de ces propriétés permet de "mettre à jour" le résultat de la fouille du graphe (au lieu de le fouiller à nouveau) pour chaque changement dans le graphe, c'est-à-dire pour chaque nouveau graphe statique.

2.4.1.3 Les textes bruts

En dehors des deux types de textes simples précédemment mentionnés (section 2.2.2 page 15), un troisième type de texte devient très courant : le texte brut (Feldman and Sanger, 2007).

Les auteurs définissent un texte brut comme une séquence de caractères prise parmi un ensemble fini de caractères d'où toute mise en forme est absente (Aggarwal and Zhai, 2012; He et al., 2013). Les caractères intègrent toutes les lettres de l'alphabet (peu importe l'alphabet, en préservant la distinction entre majuscule et minuscule), les symboles numériques et mathématiques qui peuvent être tapés sur un clavier d'ordinateur, etc.

Définition 2.4.3. *Soit $I = \{i_1, i_2, ..., i_m\}$ un ensemble fini de caractères. T est un* **texte brut** *: $T =< i_1, i_2, ..., i_n >$ et est souvent appelé un "document".*

L'analyse d'un texte brut commence par la reconnaissance des mots (Aggarwal and Zhai, 2012). Ensuite, c'est l'enchaînement de mots qui est traité. Cependant, un texte brut peut être "standard", c'est-à-dire composé des mots du dictionnaire sans fautes d'orthographes ni de grammaires, mais aussi "non standard", c'est-à-dire qu'il il peut violer de nombreuses règles grammaticales. Prenons l'exemple de messages d'utilisateurs postés dans un réseaux social : un texte écrit par un utilisateur peut être un mélange de français et d'anglais, sans respecter aucune règle grammaticale, utiliser des raccourcis (le langage SMS (Fairon et al., 2007), les smileys (Marcoccia, 2000), etc.). De plus, les documents textuels deviennent de plus en plus volumineux et nombreux (Lamirel and Cuxac, 2013).

L'objectif de la fouille de texte brut est donc de traiter une grande quantité de textes afin d'en comprendre le sens : extraire les motifs et les relations sémantiques entre ces motifs en utilisant les techniques de traitement automatique des langues (TAL) (Gupta and Lehal, 2009). À noter que la fouille de textes simples (tableaux de données ou textes semi-structurés) ne requiert pas l'application de techniques de traitement automatique des langues.

Plusieurs algorithmes ont été proposés pour le traitement de grandes quantités de textes. Cependant, ces solutions ont encore des limites (Rebholz-Schuhmann et al., 2005; Oda et al., 2008; Dai et al., 2010) : (i) l'analyse sémantique est encore coûteuse en temps et en mémoire requis pour les textes très volumineux, ainsi que pour les textes multi-langues ; (ii) l'exploitation des connaissances liées au domaine d'un texte est très importante car elle influence le sens des motifs extraits (Rajpathak et al., 2012; Rajpathak, 2013). Cependant, les textes multi-domaines ne sont pas encore efficacement fouillés (Aggarwal and Zhai, 2012). La fouille de texte brut est donc un domaine dynamique, qui comporte encore de nombreux défis.

2.4.1.4 Les flux de données

Un autre type de données complexes, pour lequel les recherches sont particulièrement actives, sont les flux de données. Un flux de données est une séquence infinie d'éléments générés de façon continue à un rythme rapide (Krempl et al., 2014). Le terme "rapide" signifie que la vitesse d'arrivée des nouveaux éléments est grande relativement aux capacités de traitement et de stockage (Csernel et al., 2005). Il est important de noter que la nature d'un élément dans un flux peut varier. Un élément peut être constitué d'un ou de plusieurs événements (ce qui constitue un flux d'événements) (Cugola and Margara, 2012), être un graphe (ce qui constitue un flux de graphes) (McGregor, 2014), un texte brut (ce qui constitue un flux de textes) (Aggarwal and Zhai, 2012), etc. Nous allons présenter la définition formelle d'un flux de données :

Définition 2.4.4. *Soit E un ensemble d'éléments (événements, graphes, textes, etc.) qui peut être infini. $F =< (t_1, E_{t_1}), (t_2, E_{t_2}), ..., (t_n, E_{t_n}) >$ (avec $t_1 < t_2 < ... < t_n \wedge n \simeq \infty$) est un* **flux de données**. *Le flux a une longueur infinie et sa vélocité peut être déduite en fonction des instants d'apparition t_i de ses éléments.*

Comme nous pouvons le remarquer, la définition d'un flux de données est semblable à celle d'une séquence d'événements. La différence majeure réside dans la vélocité qui n'est pas un facteur important dans une séquence d'événements et dans l'infinité du flux.

La généralisation de données du Web est la raison principale de l'apparition d'un grand nombre de flux de données (Bifet, 2013). Prenons par exemple les flux de messages d'utilisateurs dans un blog, les flux de pages Web (qui forment des graphes en raison des hyperliens), les flux d'articles scientifiques publiés avec les citations, etc. ou bien toutes ces caractéristiques au sein d'un même flux.

Quel que soit le domaine auquel ils sont appliqués, les algorithmes de traitement de flux de données doivent respecter un certain nombre de contraintes (Aggarwal, 2007). (i) ils ne peuvent pas stocker toutes les informations disponibles en raison du volume du flux de données ; (ii) en raison de la vélocité et du volume du flux, les algorithmes de traitement n'ont droit qu'à une seule passe sur chaque élément et, malgré l'infinité du flux, doivent s'exécuter en utilisant une mémoire de taille finie ; (iii) étant donné le rythme élevé d'arrivée des données, le temps de traitement par élément doit être impérativement court.

Dans cette thèse, nous nous intéressons à la prédiction dans un flux de données. Nous allons détailler les techniques de fouille de flux de données dans une section ultérieure.

2.4.2 Les défis de la fouille de données complexes

Pour minimiser le défi associé au volume de données, une des solutions est de simplement améliorer la capacité de mémoire et la puissance de calcul (Wu et al., 2014), ou d'utiliser des plateformes de traitement intensif et en temps réel de données, telles que Apache Hadoop (Apache Hadoop, 2005), ou STORM (Apache Storm, 2012), qui ont émergé avec l'apparition de données complexes. Quel que soit le type de données complexes, leur grand volume n'est pas la seule source de défi qu'elles apportent à la fouille de données. La complexité de la fouille de données complexes est liée à d'autres aspects de ces données (Aggarwal, 2015; Wu et al., 2014) :

— Les données hétérogènes : bien qu'habituellement décrites comme non structurées, les données complexes peuvent également contenir des données structurées ou semi-structurées (par exemple un mélange de texte, d'images et de vidéos) (Fan and Bifet, 2013). Les données à traiter ont donc une structure hétérogène et cela complexifie la tâche de fouille de données car il faut soit proposer une structure unique des données avant leur

modélisation, soit proposer un algorithme de fouille capable de traiter cette hétérogénéité en appliquant des techniques innovantes d'intégration de données (Calì et al., 2013; Vincini et al., 2013). Ces techniques exigent souvent la présence et l'exploitation de métadonnées pour chaque structure de données (Ochsenbein and Ortiz, 2001).

— Les dépendances dans les données : la dépendance représente le fait d'avoir des relations explicites ou implicites entre les données. Prenons l'exemple d'un graphe d'utilisateurs d'un réseau social. Les arêtes représentent une dépendance explicite entre les utilisateurs. Dans une séquence d'événements, les dépendances sont implicites : deux messages successifs dans un blog sont liés l'un à l'autre. Ainsi, les données temporelles déterminent implicitement une dépendance entre des éléments successifs. Comme ces dépendances implicites sont difficiles à trouver, les exploiter est également complexe. Les dépendances dans les données apportent beaucoup d'informations, leur prise en compte est donc importante dans le processus d'extraction de connaissances et d'évaluation de données (Aggarwal, 2015). Ceci ajoute des contraintes sur la fouille de ces données et augmente donc la complexité algorithmique.

— Les associations sémantiques au sein des données : les associations sémantiques sont des connexions, comme par exemple la connexion entre deux phrases successives dans un texte, qui peuvent être considérées comme des dépendances implicites, la différence est qu'elles exigent une connaissance experte (linguistique par exemple) pour les extraire (Jiang et al., 2012; Viswanathan and Krishnamurthi, 2012)). Les multiples sources des données, comme les actualités sur le Web, les commentaires sur Twitter, les images sur Flickr, les vidéos sur YouTube, etc. peuvent toutes discuter en même temps d'un même événement social (par exemple le gagnant d'un prix académique). Il est indéniable qu'il existe une association sémantique très forte entre ces données. Extraire les associations sémantiques à partir d'un ensemble hétérogène de données peut aider à améliorer la performance de plusieurs applications. Cela est le cas par exemple dans les systèmes de recommandation (Wang et al., 2009). Cependant, il est très complexe de découvrir de telles relations sémantiques à partir de données hétérogènes car il faut tout d'abord s'attaquer au problème de l'hétérogénéité dans les données (Rishe et al., 2000; Vincini et al., 2013).

2.4.3 L'impact de données complexes sur la fouille de données : les nouvelles connaissances fouillées

Rappelons que les données complexes contiennent une très grande richesse d'informations non présente dans les données simples, ce qui les rend très prometteuses (Wu et al., 2014). Les données complexes peuvent cacher des informations très importantes qui représentent, par exemple, un risque ou bien une solution future pour l'application dont elles sont extraites. Ces informations, une fois détectées, peuvent aider à améliorer l'application et à mieux comprendre son évolution. Par conséquent, de nouveaux axes de recherches dans le domaine de la fouille de données sont apparus pour extraire des nouvelles connaissances :

— Les événements émergents. La détection d'événements émergents est généralement représentée par la détection des thèmes émergents (appelée parfois détection des tendances) et par la détection des nouvelles de dernière heure dites "breaking news" (Cataldi et al., 2010; Mathioudakis and Koudas, 2010; Alvanaki et al., 2011; Budak et al., 2011). La détection des événements émergents est un domaine récent dans la fouille de données, qui est apparu dans les flux de données issus de réseaux sociaux. Elle consiste en la détection, en temps réel, des événements (thèmes ou tendances) les plus fréquents au cours

d'une période et en la détection des événements devenus fréquents de manière imprévue (breaking news). En principe, la détection de l'émergence exige la surveillance de la fréquence d'apparition des événements au fil du temps, afin de décider leur émergence, bien avant qu'ils soient très fréquents, ce qui permet au système d'agir ou de réagir au bon moment (Cataldi et al., 2010). Cependant, les algorithmes traditionnels, souvent de clustering ou des modèles statistiques, ne sont pas adaptés pour plusieurs raisons (Ma et al., 2013) : (i) ils reposent souvent sur l'exploitation d'un grand historique de données sur une longue durée, ce qui ne permet pas de détecter l'émergence présente sur une période courte par manque d'historique ; (ii) ils ne sont pas capables de gérer le bruit souvent présent dans les données, ce qui peut conduire à la détection d'une fausse émergence ou à la non détection d'une émergence effective. Par conséquent, les nouveaux algorithmes de détection de ces connaissances doivent faire face à l'historique de données et au bruit, en extrayant et en exploitant les dépendances et les associations sémantiques cachées dans les données (Saha and Sindhwani, 2012; Aggarwal and Subbian, 2012). Ces techniques sont détaillés dans une section ultérieure (voir section 2.7).

— Les opinions et les sentiments. Ce domaine, qui fait partie de la fouille de texte, a pour object de découvrir et d'examiner les avis d'utilisateurs (avis positifs, négatifs, neutres, etc.) sur un produit, un site Web, etc. Les algorithmes standards de traitement automatique des langues ne sont pas adaptés pour cette tâche pour plusieurs raisons (Maynard et al., 2012) : (i) ils sont conçus pour l'analyse de textes "standards", c'est-à-dire composés des mots du dictionnaire sans fautes d'orthographes ni de grammaires et non pas pour les textes bruts générés par les utilisateurs ; (ii) ils ont des difficultés à comprendre le sens de textes qui sont souvent très courts et dans lesquels les informations sont contextuelles et implicites. Cependant, plusieurs nouveaux algorithmes sont proposés dans la littérature pour fouiller ce type de connaissances. Ils sont basés par exemple sur la reconnaissance d'événements (Maynard et al., 2012), l'annotation automatique de gros corpus de textes pour améliorer la classification des opinions et des sentiments (Vinodhini and Chandrasekaran, 2012), l'extraction de relations syntaxiques (Greene and Resnik, 2009) et de mots dits "shifter", c'est-à-dire qui influencent fortement la classification (Li et al., 2010), etc.

— Les utilisateurs spammeurs. En opposition aux méthodes traditionnelles de détection de spam (comme dans les courriels électroniques), de nouvelles méthodes sont apparues avec les réseaux sociaux. En effet, dans les réseaux sociaux où les données sont générées par les utilisateurs, il existe des utilisateurs appelés "spammeurs" qui diffusent des mots clés ou des messages indésirables contenant, par exemple, des hyperliens "spam", qui peuvent influencer négativement la pertinence de l'application. Pour détecter ces utilisateurs, les algorithmes traditionnels de détection de spam ne sont pas adaptés, car ils sont souvent basés sur un système de filtrage pour distinguer le contenu (les mots indésirables par exemple) censés augmenter la probabilité d'un message d'être un spam (Cormack, 2007). Cependant, dans les réseaux sociaux, les spammeurs insèrent souvent, dans le contenu diffusé, des mots représentant des événements typiques ou de dernière heure, ce qui encourage les autres utilisateurs à leur faire confiance et à cliquer sur des liens spams, et cela devient donc difficile de les filtrer (Benevenuto et al., 2010). Par conséquent, de nouveaux algorithmes sont proposés basés par exemple sur l'étiquetage de profils d'utilisateurs, à savoir classer chaque profil comme spammeur ou non spammeur, puis sur l'apprentissage automatique, à partir de ces profils, d'un modèle qui sera appliqué pour filtrer les nouveaux profils (Benevenuto et al., 2010; Halpin and Blanco, 2012; Fei et al., 2013). Cependant, deux manières pour l'étiquetage de profils d'utilisateurs sont

connues dans la littérature : (i) en se basant sur le contenu diffusé par chaque profil (Lee et al., 2010), (ii) en se basant sur l'influence de chaque profil dans le réseau social et sur sa confiance attribuée par les autres profils (Ghosh et al., 2012).

— Les liens. C'est un domaine récent, motivé par la généralisation de données hétérogènes, volumineuses et brutes. Ce domaine regroupe l'analyse des liens (Feldman, 2002), la fouille du Web (Eirinaki and Vazirgiannis, 2003), la fouille de données relationnelles (Džeroski, 2010) et la fouille de graphes (Washio and Motoda, 2003). Un lien représente une relation entre objets et peut donc être un hyperlien dans une page Web, une relation d'amitié (entre deux utilisateurs dans un réseau social), une citation entre deux documents, etc. La fouille de liens considère explicitement les liens entre les données dans le processus de fouille (Getoor and Diehl, 2005). Comme dans la fouille d'autres connaissances présentées précédemment, le défi dans la fouille de liens à partir de données complexes, est représenté par le fait que les données sont hétérogènes avec des structures multiples et l'existence de plusieurs types de relations entre eux. Les algorithmes traditionnels, comme la fouille par règles d'association ou par clustering, tentent de trouver des relations entre objets souvent homogènes et liés par une unique relation (Getoor, 2003), ils ne sont donc pas adaptés à la fouille de liens. Par conséquent, de nouveaux algorithmes sont proposés dans la littérature, ils exigent souvent la présence de métadonnées pour chaque objet afin de traiter le problème d'hétérogénéité. Ces algorithmes considèrent les caractéristiques présentes dans les liens et les informations dans le contenu des objets liés (Richardson and Domingos, 2001).
Parmi les tâches les plus connues pour la fouille de liens, nous pouvons citer (Getoor, 2003; Lu and Getoor, 2003) : la classification des liens, la prédiction du nombre des liens, la prédiction du type des liens, la prédiction de liens, la prédiction de l'identité des objets, l'extraction de sous-graphes, etc.

2.5 La fouille de séquences d'événements

Dans cette section, nous nous intéressons à un type particulier de données complexes : les séquences d'événements $S =< (t_1, I_{t_1}), (t_2, I_{t_2}), ..., (t_n, I_{t_n}) >$ avec $t_1 < t_2 < ... < t_n$ (voir définition 2.2.9 page 15). Tout comme l'analyse des bases de séquences ou de transactions, l'analyse de séquences d'événements cherche à comprendre les acteurs principaux (c'est-à-dire les générateurs des événements : un utilisateur ou un système) et à comprendre ou à établir des traitements ou des réactions au sein de cette séquence (Dong and Pei, 2007). (Mannila et al., 1997) est le premier à proposer des méthodes de fouille d'éléments dans une séquence d'événements qu'il a appelés des "épisodes" et des "règles d'épisode". Depuis son apparition, la fouille d'épisodes et de règles d'épisode est devenue un domaine important et actif en fouille de données (Luo and Bridges, 2000; Méger and Rigotti, 2004; Zhu et al., 2010; Gan and Dai, 2011; Tatti and Cule, 2012; Lin et al., 2015b).

La fouille de séquence d'événements diffère de la fouille de bases de séquences par plusieurs points : (i) seul l'ordre des items est pris en compte dans la fouille de bases de séquences, contrairement à la fouille de séquences d'événements où la notion de temps est prise en compte ; (ii) l'occurrence d'un motif est recherchée dans une séquence et bornée par la taille de cette séquence (relativement petite) et une seule occurrence est trouvée dans une seule séquence. Au contraire, toutes les occurrences d'un épisode sont recherchées dans une unique très longue séquence d'événements, et donc une occurrence doit être bornée dans ce que l'on appelle *fenêtre*. Les occurrences peuvent donc se chevaucher, comme nous détaillerons par la suite.

Dans la suite de cette section, nous allons tout d'abord présenter quelques exemples applicatifs pour monter l'importance de la fouille de séquences événements. Nous allons, ensuite, définir les notions principales, détailler l'extraction d'épisodes et de règles d'épisode dans une séquence, et présenter quelques contraintes imposées lors de l'extraction des épisodes et des règles d'épisode.

2.5.1 Importance et exemples d'applications

La fouille de séquences est un domaine qui étudie les techniques nécessaires pour l'extraction de connaissances cachées dans des séquences. Ce domaine est très attractif à la fois pour la recherche scientifique et pour le développement (Dong and Pei, 2007). Nous allons présenter quelques exemples applicatifs liés à la fouille de séquences d'événements :

— Dans le cadre du contrôle des épidémies des maladies, comme par exemple la détection des attaques à l'anthrax (Wong et al., 2005), les informations (consignées dans les fiches des patients : date d'admission, âge, symptômes, traitements proposés, code postal, etc.) dont disposent les hôpitaux, sont exploitées. L'ensemble de ces informations peut former une séquence d'événements : les éléments de fiches patient représentent des événements qui peuvent être ordonnés par la date d'admission pour former une séquence de fiches patient. Dans ce contexte, l'objectif est de comprendre voire de prédire la propagation d'une épidémie. Par exemple, il est important de savoir que si $symptôme_A$ et $symptôme_B$ sont présents dans une même ville humide, cela va conduire à la propagation de $symptôme_A$ vers une ville voisine (ceci forme une règle d'association : ($symptôme_A$, $ville_1$ humide), ($symptôme_B$, $ville_1$ humide) $\rightarrow$ ($symptôme_A$, $ville_2$)). Cela rentre dans le cadre de la **fouille de motifs**.

— Dans un système d'entretien d'avions, une séquence de registre d'état est conservée pour chaque avion, ce qui représente une séquence d'événements par avion. On sait que réaliser l'entretien d'un avion dans un aéroport national est beaucoup moins cher que dans un aéroport international. Pour cela, il est nécessaire de classer un avion en fonction de la nécessité ou non d'un entretien avant son décollage, en prenant en compte sa séquence de registre. Cet objectif s'inscrit dans le cadre d'une **classification de séquences** : un historique de séquences de registre d'avions peut être utilisé pour l'apprentissage d'un modèle qui servira ensuite à classer les avions.

— Dans un cadre médical, un nouveau médicament doit être testé sur des patients avant qu'il ne soit validé. Pour chaque patient, une séquence de réactions médicales (comme par exemple un changement de température, la tension, etc.) est collectée, ce qui représente une séquence d'événements. Dans ce contexte, l'objectif est de détecter l'ensemble des réactions au médicament : connaître les réactions principales et les réactions rares, etc. Pour cela, comme plusieurs patients peuvent avoir des réactions similaires, il est possible de les regrouper en fonction de la similarité entre leur séquences de réactions. Cela rentre dans le cadre d'un **clustering de séquences**. Notons que le clustering de séquences d'événements peut être très similaire au clustering de transactions, sauf si la notion de temps entre les événements est prise en compte pour le clustering d'une séquence d'événements.

2.5.2 Définitions

Pour présenter les différentes notions liées à l'extraction d'épisodes et de règles d'épisode dans une séquence d'événements, nous allons prendre un "exemple jouet" d'une séquence d'événements S, présenté dans la figure 2.1. Cet exemple sera utilisé tout au long de ce manuscrit (voir définition 2.2.8 et définition 2.2.9 page 15 pour la représentation formelle d'un événement et d'une séquence

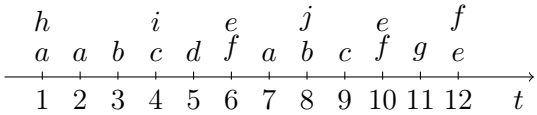


FIGURE 2.1 – Exemple d'une séquence d'événements S.

d'événements). L'événement ef composé de l'itemset $\{e, f\}$, apparaît trois fois : aux instants t_6, t_{10} et t_{12}.

On peut observer que, dans la séquence d'événements S, le motif "l'événement a est apparu régulièrement avant l'événement ef et ce, au plus cinq instants avant ef", apparaît plusieurs fois. Ce type de motif est appelé épisode, il peut être noté $(< a, ef >, 5)$. Les algorithmes dans l'état de l'art n'ont pas pour habitude d'associer la notion de temps à chaque épisode (comme nous l'avons noté) car la valeur de temps associé change en fonction de paramètres de l'algorithme. Nous allons donc reprendre la notation telle qu'elle est dans la littérature et notons un épisode $< a, ef >$. (Mannila et al., 1997) sont les premiers à introduire la notion d'épisodes et à proposer les algorithmes de fouille associés. Ils définissent un épisode comme une collection d'événements *relativement proches*, un épisode peut être comparé à un graphe acyclique où les nœuds représentent les événements et les arêtes représentent la nature de l'ordre entre les événements. Il est important de noter que les motifs peuvent être définis de la même manière que les épisodes, à la différence que dans les épisodes la notion de temps est prise en compte lors de la fouille des épisodes. Selon la nature de l'ordre utilisé, les épisodes peuvent être classés en trois types : les épisodes sériels, les épisodes parallèles et les épisodes composés.

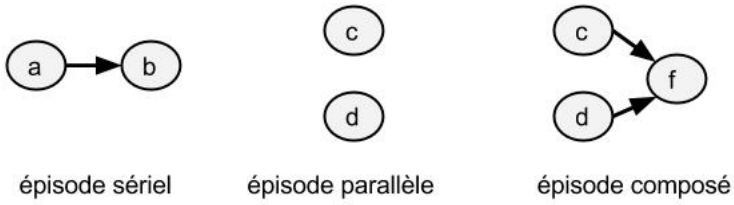


FIGURE 2.2 – Les différents types d'épisodes.

Définition 2.5.1. *Un* **episode** P *est une liste partiellement ordonnée d'événements. Il existe trois types d'épisodes (voir figure 2.2) :*

- *Épisode sériel :* $P =< p_1, p_2, ..., p_k >$ *sur* I^k *avec* $p \subseteq I$. *L'ordre entre les événements est total. Le premier événement* p_1 *de l'épisode représente son* **préfixe** *et le dernier événement* p_k *représente son* **suffixe**.
- *Épisode parallèle :* $P = \{p_1, p_2, ..., p_k\}$ *sur* I^k *avec* $p \subseteq I$. *Il n'y a pas d'ordre entre les événements qui composent l'épisode.*
- *Épisode composé : l'épisode est partiellement sériel et partiellement parallèle, c'est-à-dire qu'il se forme récursivement par une composition sérielle et parallèle des événements.*

Les deux types d'épisodes les plus fouillés et utilisés dans la pratique sont les épisodes sériels et les épisodes parallèles (Mannila et al., 1997; Méger and Rigotti, 2004). Dans cette thèse, nous nous intéressons à l'ordre entre les événements. Pour cela, dans la suite du manuscrit nous nous focalisons sur les épisodes sériels, et que nous les appellerons "épisodes", pour simplifier la dénomination.

Définition 2.5.2. *Soit l'épisode* $P =< p_1, p_2, ..., p_k >$. *La* **taille** *de l'épisode* P *représente le nombre de ses événements, elle est notée* $|P|$: $|P| = k$. *Soit* $Q =< q_1, q_2, ..., q_s >$ *un autre épisode. La* **concaténation** *de l'épisode* P *et de l'épisode* Q *forme l'épisode noté* $P \cdot Q =< p_1, ..., p_k, q_1, ..., q_s >$ *de taille* $k + s$.

Définition 2.5.3. P *est un* **sous-épisode** *de* Q, *noté* $P \subseteq Q$, *si* $\forall_i\ p_i \subseteq q_i$ *(*p_i *peut être* $\emptyset$*). Dans ce cas,* Q *est appelé un* **super-épisode** *de* P.

Prenons un exemple de la figure 2.1. L'épisode $P =< ef >$ est un sous-épisode de l'épisode $Q =< a, ef >$. En revanche, l'épisode $P' =< ae >$ n'est pas un sous-épisode de Q.

Définition 2.5.4. *L'***occurrence** *d'un épisode* P, *notée* $occ_i(P)$, *est la suite des instants (temps) d'apparition, dans la séquence, des événements qui composent cet épisode. Dans la littérature, l'occurrence d'un épisode est généralement notée* $occ_i(P) = (t_{s_i}, t_{e_i})$ *: les instants d'apparition de son préfixe et de son suffixe, car ce sont les événements les plus importants et le plus souvent utilisés par les algorithmes, car ils bornent l'occurence de l'épisode. Pour simplifier la notation, l'occurrence sera notée comme suit :* $occ(P) = (t_s, t_e)$. *La liste de toutes les occurrences d'un épisode* P *est notée* $Occ(P)$.

Définition 2.5.5. *Le* **support** *de l'épisode* P, *noté* $supp(P)$, *représente le nombre de ses occurrences qui sont obtenues selon la mesure de fréquence appliquée :* $supp(P) = |Occ(P)|$. *L'épisode* P *est considéré comme un* **épisode fréquent** *dans la séquence, si* $supp(P) \geq minsupp$, *où* $minsupp$ *est un seuil de support prédéfini.*

Il est important de mentionner que la notion d'occurrence peut varier en fonction des différents contextes d'application. La mesure de fréquence appliquée est la seule à être impactée lors d'un changement de contexte.
Sur le même schéma de la définition des règles d'association à partir de motifs (voir définition 2.3.5 page 20), il est possible de définir des règles d'épisode à partir d'épisodes.

Définition 2.5.6. *Une* **règle d'épisode** R *est une implication* $R : P \rightarrow Q$ *qui signifie que l'épisode* Q, *qui représente la conséquence de la règle, apparaît régulièrement après l'épisode* P, *qui représente l'antécédent. Le terme "régulièrement" signifie que la conséquence apparaît après l'antécédent avec une probabilité relativement élevée. Le* **support de la règle** R *représente le support de la concaténation des deux épisodes* P *et* Q *qui la composent :* $supp(R) = supp(P \cdot Q)$.

Définition 2.5.7. *La* **confiance** *de la règle d'épisode* $R : P \rightarrow Q$ *représente la probabilité que sa conséquence* Q *apparaisse, sachant que son antécédent* P *est apparu, comme présenté dans l'équation (2.2).*

$$conf(P \rightarrow Q) = \frac{supp(P \cdot Q)}{supp(P)} \tag{2.2}$$

Lorsque la confiance de la règle d'épisode dépasse un seuil prédéfini $minconf$, *la règle est considérée comme une* **règle confiante**.

La tâche de fouille de règles d'épisode consiste en règle générale à extraire les règles valides : les règles d'épisode fréquentes et confiantes (Mannila et al., 1997).
Rappelons que la différence entre la fouille de motifs et la fouille d'épisodes réside dans la recherche des occurrences. Plusieurs définitions restent les mêmes : comme le support, la confiance (pour les règles), etc.

2.5.3 Algorithmes d'extraction d'épisodes dans une séquence d'événements

Les deux premiers algorithmes proposés dans la littérature pour l'extraction d'épisodes et de règles d'épisode sont $WINEPI$ et $MINEPI$ (Mannila et al., 1997) qui sont la base de nombreux algorithmes proposés par la suite (Huang and Chang, 2008; Hu et al., 2013; Tseng et al., 2015). Ces deux algorithmes ont trois points communs : (i) l'approche pour former les épisodes : ils commencent tous les deux par l'extraction d'épisodes composés d'un seul événement, et puis étendent itérativement ces épisodes en fusionnant des événements sur leur côté droit. Cette approche est encore utilisée par de nombreux algorithmes récents (Méger and Rigotti, 2004; Laxman et al., 2007; Hu et al., 2013) ; (ii) la sélection des épisodes dits fréquents : les épisodes qui apparaissent suffisamment souvent dans une séquence d'événements, c'est-à-dire les épisodes pour lesquels le nombre d'apparitions dépasse un seuil prédéfini ; (iii) l'extraction de règles d'épisode. Que signifie "apparition suffisante" (connu sous le terme "occurrence") d'un épisode ? La réponse à cette question est multiple et traitée dans la littérature par la proposition de multiples *mesures de fréquence* (de comptage) (Mannila et al., 1995; Mannila and Toivonen, 1996). Deux mesures de fréquence sont proposées dans les deux algorithmes $WINEPI$ et $MINEPI$. Nous allons tout d'abord détailler ces deux algorithmes. Nous présentons ensuite d'autres mesures de fréquence proposées dans la littérature.

2.5.3.1 L'algorithme WINEPI

La mesure de fréquence proposée dans l'algorithme $WINEPI$ (Mannila et al., 1997) repose sur la notion de fenêtre.

Définition 2.5.8. *Un segment dans la séquence S, de taille w, qui commence à l'instant t_s est appelé une* **fenêtre** *et noté $Win(S, t_s, w)$. Un instant t_k est contenu dans la fenêtre $Win(S, t_s, w)$ et noté $t_k \in Win(S, t_s, w)$ si $t_s \leq t_k \leq (t_s + w - 1)$.*

La séquence dans la figure 2.1 contient plusieurs fenêtres dont par exemple : $Win(S, 2, 5)$ qui débute à l'instant t_2, se prolonge sur cinq instants et se termine donc à l'instant t_6. L'instant $t_3 \in Win(S, 2, 5)$, car $t_2 \leq t_3 \leq t_6$.

Définition 2.5.9. *La* **mesure de fréquence basée sur des fenêtres** *(windows-based frequency measure) est définie comme suit : Une occurrence de l'épisode $P =< p_1, p_2, ..., p_k >$ est présente dans une fenêtre $Win(S, t_s, w)$ si*

- *Le préfixe p_1 de l'épisode P apparaît à l'instant t_s (au début de la fenêtre).*
- *$\exists\ t_e \in Win(S, t_s, w)$ tel que le suffixe p_k de l'épisode P apparaît à instant t_e.*
- *$\forall\ p_i, p_{i+1} \in P : p_i, p_{i+1}$ apparaissent aux instants $t_j, t_{j'} \in Win(S, t_s, w)$ respectivement avec $t_j < t_{j'}$.*

Cette occurence est notée $occ(P) = (t_s, t_e) \in Occ(P)$.

Prenons l'exemple des deux fenêtres $Win(S, 1, 5)$ et $Win(S, 7, 5)$ de la séquence S. L'épisode $P =< a, c >$ dispose de deux occurrences dans ces fenêtres : $Occ(< a, c >)= \{(t_1, t_4), (t_7, t_9)\}$. Une fois la mesure de fréquence définie et la taille de fenêtres fixée, il est alors possible de calculer le support de l'épisode, comme défini dans la définition 2.5.5 et donc d'évaluer si l'épisode apparaît souvent ou non dans la séquence d'événements. Pour l'épisode $P =< a, c >$, $supp(a, c) = |Occ(< a, c >)| = 2$.
Comme nous l'avons mentionné, l'algorithme $WINEPI$ est composé de deux sous-algorithmes : (i) le premier est dédié à l'extraction des épisodes fréquents, (ii) à partir de ces épisodes fréquents et de leurs supports, le deuxième algorithme se charge de former les règles d'épisode et de sélectionner les règles confiantes.

Extraction des épisodes fréquents : L'extraction des épisodes fréquents par l'algorithme $WINEPI$ est réalisée en deux phases : une phase de génération des épisodes candidats (Algorithme 1 lignes 5 et 9), et une phase d'évaluation du support de ces candidats (Algorithme 1 ligne 7), de façon à déterminer les épisodes fréquents.
L'algorithme effectue un parcours en largeur, c'est-à-dire qu'une fois que l'ensemble des épisodes candidats d'une taille m donnée est généré (l'ensemble noté C_m dans Algorithme 1) et leur support évalué, il génère l'ensemble des épisodes candidats de taille $m+1$ (ensemble C_{m+1} dans Algorithme 1). Les épisodes de taille 1 sont donc d'abord générés (Algorithme 1 ligne 5), puis itérativement les épisodes de taille plus grande. Par conséquent, $WINEPI$ peut être comparé aux approches de type Apriori (voir section 2.3.1 page 17) : la différence principale réside dans la définition des épisodes et le mode d'évaluation du support de ces épisodes. Rappelons que le support doit être une mesure qui possède la propriété d'anti-monotonicité (voir définition 2.3.4 page 17) : lorsqu'un épisode n'est pas fréquent, aucun de ses super-épisodes ne peut être fréquent, donc aucun épisode candidat ne sera formé à partir de cet épisode. Comme dans l'algorithme Apriori (voir section 2.3.1 page 17), l'algorithme $WINEPI$ s'arrête lorsque plus aucun candidat ne peut être généré (Algorithme 1 ligne 6).

Algorithm 1: $WINEPI$: sous-algorithme 1 extraction des épisodes fréquents

Données: S : la séquence d'événements, w : la taille des fenêtres, $minsupp$: le seuil de support minimal.
Résultat: E : L'ensemble des épisodes fréquents

1 **begin**
2 $E = \emptyset$; `// initialisation de l'ensemble des épisodes fréquents`
3 $m = 1$; `// m représente l'indice de la taille des épisodes`
4 $C_m = \emptyset$; `// initialisation de l'ensemble des candidats de taille 1`
5 calculer $C_m = \{P : |P| = 1\}$; `// génération des candidats de taille 1`
6 **tant que** $(C_m \neq \emptyset)$ **faire**
`// évaluation du support :` $|Occ(P)| \geq minsupp$
`// ` E_m `représente l'ensemble d'épisodes fréquents de taille` m
7 calculer $E_m : \{P : P \in C_m \wedge supp(P) \geq minsupp\}$
8 $E = E \cap E_m$
`// génération des candidats de taille` $m+1$
9 calculer $C_{m+1} = \{P : |P| = m+1 \ \wedge \ (\forall P', P'' \subset P \wedge P', P'' \in E_m : \text{suffixe}(P') = \text{préfixe}(P''))\}$
10 $m = m + 1$

Extraction de règles d'épisode : Le deuxième sous-algorithme de $WINEPI$ a pour but d'extraire les règles d'épisode confiantes (Algorithme 2 qui repose sur Algorithme 1). Il procède en deux étapes : (i) la génération des règles d'épisode candidates : formées à partir de l'ensemble des épisodes fréquents (ligne 3). Soit P un épisode fréquent avec $P = P' \cdot P''$. Une règle candidate est donc formée de manière à ce que P' représente son antécédent et P'' représente sa conséquence (lignes 5 et 6) : $R : P' \rightarrow P''$. Rappelons que si un épisode P est fréquent, alors tous ses sous-épisodes P' et P'' sont fréquents (par la propriété d'anti-monotonicité), et que la construction de règles d'épisode à partir des épisodes fréquents assure donc que les règles candidates formées sont fréquentes ; (ii) l'évaluation de la confiance de chaque règle candidate (ligne 7) : si la confiance d'une règle candidate est supérieure à un seuil prédéfini $minconf$ (voir définition 2.5.7 page 31), alors elle sera une règle résultat de l'algorithme.

Algorithm 2: $WINEPI$: sous-algorithme 2 extraction de règles d'épisode

Données: E : L'ensemble des épisodes fréquents, $minconf$: le seuil de support minimal.
Résultat: $E_{\text{règles}}$: L'ensemble des règles d'épisode

```
1 begin
2     E_règles = ∅ // initialisation de l'ensemble de règles d'épisode
3     pour chaque ( P ∈ E ) faire
4         pour chaque ( P' ⊂ P ) ;                    // génération de candidats
5         faire
6             calculer P'' : P = P'.P''
7             si conf(P' → P'') ≥ mincof // évaluation de candidats
8             alors
9                 E_règles = E_règles ∩ {P' → P''}
```

Impact de la mesure de fréquence basée sur des fenêtres : Revenons sur l'évaluation du support des épisodes candidats. Cette évaluation nécessite la détermination d'une taille de fenêtre w dans lesquelles les occurrences de l'épisode seront recherchées, ce qui permet de fixer la durée maximale sur laquelle peut s'étendre l'occurrence d'un épisode dans la séquence. Les algorithmes gèrent parfaitement les valeurs de w et sont capables à choisir les bonnes valeurs. Cependant, l'importance de cette durée varie d'une application à l'autre. Pour cela, il est préférable que la taille de fenêtres soit fixée par un expert du domaine.
Il est important de noter que, en changeant la taille de fenêtre, le support d'un épisode peut être modifié : plus la taille de fenêtre est grande, plus on trouvera d'occurrences de l'épisode, et vice versa. Prenons par exemple l'épisode $< a, f >$. Pour des fenêtres de taille $w = 4$, une seule occurrence est trouvée dans la séquence S : $Occ(< a, f >) = \{(t_7, t_{10})\}$, $supp(< a, f >) = 1$. Cependant, pour des fenêtres de taille $w = 5$, deux occurrences sont trouvées : $Occ(< a, f >) = \{(t_2, t_6), (t_7, t_{10})\}$, $supp(a, f) = 2$. Pour des fenêtres de taille encore plus grande $w = 6$, quatre occurrences sont trouvées : $Occ(a, f) = \{(t_1, t_6), (t_2, t_6), (t_7, t_{10}), (t_7, t_{12})\}$, $supp(a, f) = 4$.
Nous pouvons donc voir l'utilisation des fenêtres comme une contrainte sur les occurrences des épisodes. Cette contrainte permet de réduire la consommation de ressources en temps et en mémoire lorsque la taille de fenêtre est réduite : pour un épisode candidat, elle permet de réduire le nombre de ses occurrences, ce qui permet à son tour de diminuer le nombre d'épisodes fréquents issus d'une seule itération, et donc de réduire le nombre d'épisodes candidats lors de l'itération suivante. À noter qu'en limitant la taille de fenêtres, on risque de limiter la richesse des épisodes extraits, à savoir le nombre d'événements les composant. Enfin, il est important de noter que l'évaluation des supports des épisodes candidats exige une passe complète sur la séquence d'événements.

2.5.3.2 L'algorithme MINEPI

Le deuxième algorithme proposé dans (Mannila et al., 1997) est l'algorithme $MINEPI$. $MINEPI$ diffère principalement de l'algorithme $WINEPI$ par la définition de la mesure de fréquence. Dans $MINEPI$, l'occurrence d'un épisode n'est pas basée sur l'apparition de cet épisode dans une fenêtre de taille prédéfinie. La fenêtre, implicite dans $MINEPI$, doit être telle qu'il n'y a pas d'autre fenêtre plus petite dans laquelle apparaît ce même épisode. Cette fenêtre est considérée dans ce cas comme contenant une occurrence minimale de l'épisode. $MINEPI$ est basé

sur les listes des occurrences minimales. Comme l'algorithme $WINEPI$, l'algorithme $MINEPI$ procède à une première phase de génération d'épisodes candidats (Algorithme 3 lignes 5 et 9) puis à une phase d'évaluation du support de ces épisodes (Algorithme 3 ligne 7).

Définition 2.5.10. *Soit $occ(P) = (t_s, t_e)$ une occurrence de l'épisode P. Cette occurrence est considérée comme une* **occurrence minimale**, *notée $mo(P)$, si elle ne contient pas une autre occurrence du même épisode : si $\nexists occ'(P) = (t'_s, t'_e) : \{(t'_s, t'_e) \subseteq (t_s, t_e), i.e.\ (t_s < t'_s \wedge t'_e \leq t_e) \vee (t_s \leq t'_s \wedge t'_e < t_e)\}$. La liste des occurrences minimales d'un épisode P est notée $Mo(P)$, et le support de l'épisode : $supp(P) = |Mo(P)|$.*

Une fois l'ensemble des occurrences minimales d'un épisode identifié, le support de l'épisode est évalué de la même manière que dans l'algorithme $WINEPI$: par le nombre des occurrences minimales de l'épisode. Le principe de génération des règles d'épisode est le même que celui dans $WINEPI$ (voir définition 2.5.6 page 31), nous ne le détaillons pas ici. La différence réside principalement dans la génération des épisodes fréquents.

Extraction des épisodes fréquents : L'algorithme $MINEPI$ fonctionne comme l'algorithme $WINEPI$ pour l'extraction des épisodes. La seule différence réside dans la recherche des occurrences qui sont minimales dans $WINEPI$. À noter que pour chercher l'occurrence minimale d'un épisode de taille $m + 1$, les occurrences de deux épisodes de taille m sont exploitées en effectuant une opération de jointure temporelle.
Soient deux motifs séquentiels P, Q. La jointure temporelle (Zaki, 1998; Zaki, 2001; Ayres et al., 2002) de ces deux motifs est une opération qui consiste, à partir des listes d'occurrences de ces deux motifs, à créer des occurrences d'un motif séquentiel $P \cdot Q$ (la concaténation de P et Q). Nous allons présenter la définition de la jointure temporelle en général, et sa définition en cas de calcul d'une occurrence minimale :

Définition 2.5.11. *Soient P et Q deux épisodes, avec $occ(P) = (t_s, t_e) \in Occ(P)$, $occ(Q) = (t'_s, t'_e) \in Occ(Q)$. Pour construire la liste des occurrences de l'épisode $P \cdot Q$, une* **jointure temporelle** *des listes d'occurrences des deux épisodes P et Q est effectuée, comme suit : pour $Occ(P), Occ(Q) : Occ(P \cdot Q) = Occ(P) \cdot Occ(Q) = \{(t_s, t'_e) : t_s < t'_e\}$.*

Pour construire une liste des occurrences minimales d'un épisode $P \cdot Q$ en utilisant une jointure temporelle, une condition supplémentaire est ajoutée à la définition de la jointure temporelle :

Définition 2.5.12. *La* **jointure temporelle avec occurrence minimale** *est : $Mo(P \cdot Q) = Mo(P) \cdot Mo(Q) = \{mo(P \cdot Q) = (t_s, t'_e) : (t_s < t'_e) \wedge (\nexists mo'(P \cdot Q) : mo'(P \cdot Q) \subseteq mo(P \cdot Q))\}$.*

Dans la séquence d'événements S, considérons les deux épisodes $< a >$ et $< e >$. $Occ(< a >) = \{(1,1), (2,2), (7,7)\}$ et $Occ(< e >) = \{(6,6), (10,10), (12,12)\}$. Afin de construire la liste d'occurrences de l'épisode $< a > \cdot < e >$, l'opération de jointure temporelle est effectuée sur les listes d'occurrences de $< a >$ et de $< e >$, on obtient ainsi la liste d'occurrences suivante : $Occ(< a > \cdot < e >) = Occ(< a >) \cdot Occ(< e >) = \{(1,6), (1,10), (1,12), (2,6), (2,10), (2,12), (7,10), (7,12)\}$. Lorsqu'une occurrence minimale est exigée, la liste des occurrences est réduite et devient : $mo(< a > \cdot < e >) = mo(< a >) \cdot mo(< e >) = \{(2,6), (7,10)\}$ (à noter que dans cet exemple $occ(< a >) = mo(< a >)$ et $occ(< e >) = mo(< e >)$, puisque $< a >$ et $< e >$ sont des épisodes composés d'un unique événement).
Rappelons qu'une fois les épisodes fréquents identifiés, ces épisodes sont utilisés afin de former les règles d'épisode (fréquentes et confiantes) de manière identique à l'algorithme $WINEPI$.

Algorithm 3: $MINEPI$: sous-algorithme 1 extraction des épisodes fréquents

Données: S : la séquence d'événements, $minsupp$: le seuil de support minimal.
Résultat: E : l'ensemble des épisodes fréquents

```
1  begin
2      E = ∅ ;                    // initialisation de l'ensemble des épisodes fréquents
3      m = 1 ;                    // m représente l'indice de la taille des épisodes
4      C_m = ∅ ;                  // initialisation de l'ensemble des candidats de taille 1
5      calculer C_m = {P : |P| = 1} ;   // génération des candidats de taille 1
6      tant que ( C_m ≠ ∅ ) faire
           // évaluation du support |Mo(P)| ≥ minsupp
7          calculer E_m : {P : P ∈ C_m ∧ supp(P) ≥ minsupp}
8          E = E ∩ E_m
           // génération des candidats de taille m + 1
9          calculer C_{m+1} = {P : |P| = m + 1 ∧ ( ∀ P', P'' ⊂ P ∧ P', P'' ∈ E_m : suffixe(P') =
           préfixe(P''))}
10         calculer Mo(P) : P ∈ C_{m+1}
11         m = m + 1
```

2.5.3.3 Les mesures de fréquence pour l'extraction d'épisodes

En plus des deux mesures de fréquences présentées ci-dessus, celle basée sur des fenêtres (proposée dans l'algorithme $WINEPI$) et celle basée sur l'occurrence minimale (proposée dans l'algorithme $MINEPI$), plusieurs autres mesures ont été proposées dans l'état de l'art afin de satisfaire des besoins applicatifs et des contraintes techniques.
Dans la mesure de fréquence sans recouvrement (non-overlapped frequency measure) (Laxman et al., 2005), deux occurrences sont dites non-chevauchées si aucun événement d'une occurrence n'apparaît entre les événements de l'autre. En restreignant de manière remarquable l'espace requis pour rechercher les occurrences d'un épisode, cette mesure est connue pour être peu consommatrice de ressources en temps et en mémoire. Elle est définie comme suit :

Définition 2.5.13. *Soient $occ(P), occ'(P)$ deux occurrences de l'épisode P : $occ(P) = (t_s, t_e), occ'(P) = (t'_s, t'_e)$. Ces deux occurrences sont considérées comme des* **occurrences non-chevauchées** *(non-overlapped) si $t_e < t'_s$ (c'est-à-dire l'instant d'apparition t_e du suffixe dans $occ(P)$ précède l'instant d'apparition t'_s du préfixe dans $occ'(P)$) ou $t_s > t'_e$ (c'est-à-dire l'instant d'apparition t_s du préfixe dans $occ(P)$ précède l'instant d'apparition t'_e du suffixe dans $occ'(P)$). Si les deux occurrences sont chevauchées, c'est la première qui est généralement considérée par l'état de l'art lors du calcul du support de l'épisode.*

Un combinaison entre la mesure de fréquence basée sur l'occurrence minimale et la mesure de fréquence non chevauchée est proposée dans l'algorithme $Manepi$ (Zhu et al., 2010). Cette mesure est introduite dans l'objectif de découvrir les cas de causalité au sein de la séquence.

La mesure de fréquence basée sur l'occurence distincte (distinct occurrence-based frequency measure) (Joshi et al., 1999) est principalement proposée pour répondre aux objectifs applicatifs. Deux occurrences d'un épisode sont dites distinctes si elles ne partagent aucun instant :

Définition 2.5.14. *Soient $occ(P), occ'(P)$ deux occurrences de l'épisode P : $occ(P) = (t_s, t_e)$, $occ'(P) = (t'_s, t'_e)$. Ces deux occurrences sont des* **occurrences distinctes** *si $\forall\ t_i \in$*

(t_s, t_e), $t'_j \in (t'_s, t'_e)$ *(à savoir* t_i *est un instant d'un événement dans une occurrence de l'épisode,* t'_j *est un instant d'un événement dans une autre occurrence du même épisode) :* $t_i \neq t'_j$.

2.5.4 Extraction des épisodes avec contraintes

Comme nous l'avons mentionné, les deux algorithmes $WINEPI$ et $MINEPI$ sont des algorithmes pionniers du domaine et sont la base de plusieurs autres algorithmes. Ces algorithmes respectent tous la contrainte de support minimal (Laxman and Sastry, 2006; Achar et al., 2012). Cependant, l'utilisation de cette seule contrainte a ses limites, à la fois du point de vue applicatif mais aussi du point de vue algorithmique (Méger and Rigotti, 2004). Du point de vue applicatif, extraire des épisodes fréquents ne donne pas toujours une réponse intéressante. En effet, de très nombreux épisodes fréquents sont souvent obtenus, dont peu se révèlent intéressants pour l'application. Tout cela sans compter l'effort à fournir pour isoler les quelques épisodes intéressants parmi les autres épisodes. Ainsi, spécifier à l'avance la nature des épisodes recherchés (en imposant les contraintes nécessaires), par exemple en utilisant la connaissance du domaine d'application, permet d'obtenir des résultats à la fois plus pertinents par rapport aux besoins de l'application, et aussi des épisodes dont le volume n'est pas démesuré par rapport à la capacité d'analyse et d'interprétation de l'utilisateur.
D'un point de vue algorithmique, une contrainte peut être appliquée de deux façons : (i) en post-traitement : tous les épisodes fréquents sont tout d'abord générés puis seuls ceux qui satisfont la contrainte prédéfinie sont retenus. Cette façon de procéder a l'inconvénient de consommer beaucoup de ressources de calcul ; (ii) en intégrant la contrainte dans le processus d'extraction des épisodes, ce qui permet de concentrer les efforts de calcul sur les épisodes susceptibles de satisfaire la contrainte imposée et d'ainsi réduire les ressources de calcul nécessaires. C'est cette deuxième façon de procéder qui est la plus souvent utilisée (Méger and Rigotti, 2004; Achar et al., 2013; Cule et al., 2013).
Dans la littérature, plusieurs types de contraintes sont imposés pour l'extraction d'épisodes. Nous pouvons les classer en deux familles : les contraintes temporelles et les contraintes sur l'ensemble résultat des épisodes. Nous allons maintenant présenter ces contraintes :

Les contraintes temporelles : Ces contraintes s'appliquent sur chaque occurrence des épisodes. (Méger and Rigotti, 2004) les intègrent dans la construction des épisodes, tandis que (Harms and Deogun, 2004) les imposent au niveau de l'évaluation du support des épisodes. Deux contraintes temporelles sont connues dans la littérature :
(i) une contrainte de durée, appelée *span*, qui spécifie la durée minimale ou maximale entre le premier et le dernier événement de chaque occurrence de l'épisode. Le span, qui représente souvent la durée maximale de taille *sp* (Achar et al., 2013), a été introduit pour réduire le temps d'exécution de l'algorithme.

Définition 2.5.15. *Soit* $occ(P) = (t_s, t_e)$ *une occurrence de l'épisode* P. $occ(P)$ *satisfait la contrainte de* **span maximal** *de taille* sp *si* $(t_e - t_s) \leq sp$.

(ii) une contrainte appelée *gap* qui spécifie la durée minimale ou maximale entre deux événements consécutifs dans chaque occurrence d'un épisode (Méger and Rigotti, 2004; Achar et al., 2013). Un gap représente souvent une durée maximale de taille g (Achar et al., 2013) au sein d'une occurrence de l'épisode, comme suit :

Définition 2.5.16. *Soit* $occ(P) = (t_s, t_e)$ *une occurence de l'épisode* $P =< p_1, p_2, ..., p_k >$. $occ(P)$ *satisfait la contrainte de* **gap maximal** *de taille* g *si* $\forall i = 1, ..., k : (t_{i+1} - t_i) \leq g$.

Il est important de noter que, lors d'une opération de jointure temporelle (voir définition 2.5.11) sur des occurrences avec contraintes (span ou gap), la définition de la jointure temporelle doit être modifiée afin de respecter la contrainte imposée. Présentons la définition de la jointure temporelle en cas de l'imposition de la contrainte de gap maximal (notons que pour la contrainte de span, la jointure temporelle est modifiée de la même manière) :

Définition 2.5.17. *Soient P, Q deux épisodes et $occ(P) = (t_s, t_e) \in Occ(P)$, $occ(Q) = (t'_s, t'_e) \in Occ(Q)$. Une* **jointure temporelle avec la contrainte de gap maximal** *de taille g, est réalisée comme suit : Pour $Occ(P), Occ(Q)$: $Occ(P \cdot Q) = Occ(P) \cdot Occ(Q) = \{(t_s, t'_e) : (t_e < t'_s) \wedge (t'_s - t_e \leq g) \wedge (t'_e - t_s \leq sp)\}$.*

Dans la séquence d'événements S, afin de construire la liste des occurrences minimales pour l'épisode $< a > \cdot < e >$ en satisfaisant la contrainte de gap maximal $g = 3$, une jointure temporelle est réalisée comme suit : $Occ(< a > \cdot < e >) = Occ(< a >) \cdot Occ(< e >) = (7, 10)$.
Enfin, il est important de mentionner que les contraires d'occurrence minimale, de gap et de span peuvent être appliquées simultanément, et que les contraintes de span et de gap sont souvent appliquées dans le cas où les épisodes doivent exprimer une causalité (Achar et al., 2013).

Les contraintes sur l'ensemble résultat des épisodes : Ce type de contraintes peut être appliqué sur les épisodes dans le but d'obtenir un ensemble condensé et significatif des épisodes. Deux contraintes sont proposées dans la littérature : (i) l'ensemble des épisodes maximaux : un épisode fréquent est dit maximal si aucun de ses super-épisodes n'est fréquent (Iwanuma et al., 2005) ; (ii) l'ensemble des épisodes fermés : un épisode est dit fermé si aucun des ses super-épisodes n'a le même support que le sien (Zhou et al., 2010). Dans la littérature, il est généralement admis que peu d'information importante est perdue lors de l'application de ce type de contrainte (Pasquier et al., 1999) (Zhou et al., 2010).

Bien évidemment, il est possible de combiner plusieurs contraintes à la fois. On obtient alors un ensemble d'épisodes plus concis, et le temps requis pour la fouille de ces épisodes est réduit.

2.5.5 Extraction des règles d'épisode

Tout comme les règles d'association, l'extraction de règles d'épisode est une tâche simple et se fait traditionnellement en deux étapes, comme nous l'avons montré dans les algorithmes $WINEPI$ et $MINEPI$.
À notre connaissance, aucun algorithme de l'état de l'art ne propose l'extraction des règles d'épisode sans passer par l'étape d'extraction des épisodes fréquents. Par conséquent, concernant l'intégration de contraintes, l'extraction de règles avec contraintes consiste d'abord en l'extraction des épisodes fréquents avec ces contraintes, puis en la génération des règles. C'est le cas dans la plupart des algorithmes, comme dans l'algorithme proposé par (Méger and Rigotti, 2004) pour l'extraction des règles d'épisode avec un gap maximal (contrainte temporelle), et l'algorithme proposé par (Gan and Dai, 2011) pour les règles d'épisode dites "non-déduites" (non dérivables), c'est-à-dire des règles qu'on ne peut pas déduire à partir d'autres règles (contrainte sur l'ensemble). Il est important de noter qu'en cas de contraintes uniquement sur les règles (comme par exemple une contrainte sur l'antécédent et la conséquence de la règle), il devient impossible de générer les contraintes au moment de l'extraction des épisodes.
Plusieurs algorithmes dans la littérature se sont intéressés à l'extraction des règles en se focalisant sur les caractéristiques de ses deux parties : l'antécédent et la conséquence. Certains auteurs proposent l'extraction des règles avec un antécédent long (constitué de nombreux événe-

ments) (Pasquier et al., 1999). D'autres proposent le contraire : l'extraction des règles avec un antécédent petit pour former des règles dites "minimales"[6] (Rahal et al., 2004), qui sont considérées comme suffisantes, c'est-à-dire aucune nouvelle connaissance n'est ajoutée par des règles avec un antécédent plus long. Les règles minimales ont été proposées dans la littérature pour réduire la consommation des ressources en temps et en mémoire au moment de la construction de règles et pour éviter la redondance dans l'ensemble final de règles (Neeraj and Swati, 2012).
Focalisons nous maintenant sur les conséquences de règles et surtout sur leur taille, à savoir le nombre d'événement les composant. Pour un épisode donné P, les algorithmes de l'état de l'art proposent la construction et l'évaluation itérative de règles avec toutes les tailles possibles de conséquences (voir algorithme 2 pour plus de détail). Cette manière de construire les règles est très consommatrice de ressources surtout lorsque les épisodes sont maximaux (plus longs). Pour répondre à ces limites, plusieurs auteurs proposent de restreindre les événements dans la conséquence d'une règle en fixant à l'avance les événements qui peuvent faire partie de la conséquence (Rahal et al., 2004), ou en limitant le nombre d'événements dans la conséquence afin de diminuer les ressources consommées à condition que cela ne diminue pas la qualité et le sens de règles extraites.

2.6 Prédiction dans une séquence d'événements

Rappelons que la prédiction est un des objectifs de la fouille de données. Un modèle prédictif permet non seulement, comme tout autre modèle, de comprendre un phénomène et de retenir les informations pertinentes noyées dans la masse de données, mais également de rechercher des connaissances qui peuvent par la suite être généralisées et employées afin de prédire l'occurrence d'information et/ou prendre des décisions futures adéquates (Weiss and Indurkhya, 1998). Par conséquent, l'extraction de connaissances est une étape qui précède la prédiction et qui aide à déterminer la décision et non pas à la prendre.
Nous avons présenté dans le chapitre Introduction (voir section 1.1) deux types de modèles connus dans la fouille de données : (i) les modèles descriptifs qui visent à décrire le mécanisme de génération de données, comme par exemple les modèles de la fouille de séquences et d'extraction de motifs ou d'épisodes présentés dans les sections précédentes ; (ii) les modèles prédictifs qui visent à prédire des valeurs dans les données. Il est important de noter que dans la littérature, la majorité des modèles prédictifs reposent sur des modèles descriptifs. Nous allons maintenant nous focaliser sur les modèles des sections précédentes et sur la manière dont ils sont exploités dans la littérature pour construire des modèles prédictifs.
Le domaine médical représente le domaine d'application d'origine des modèles prédictifs (DETSKY, 1995; Steyerberg, 2008). Une difficulté généralement constatée dans ce domaine réside dans le manque d'outils pour transcrire efficacement les données médicales, ce qui les rend très bruitées. Les connaissances extraites sont donc pauvres et la prédiction très difficile en raison du bruit et de la richesse de ses données, ce qui les rend pauvres en connaissances extraites (Soni et al., 2011). Plus récemment, la prédiction a été appliquée à d'autres domaines comme la finance (Bogle and Potter, 2015), les télécommunications (Ahuja and Chakka, 2014), l'industrie pharmaceutique (Cumming et al., 2013), etc. Une des applications les plus récentes est la prédiction du risque-client (Bilbrey Jr et al., 2013), utilisée dans l'ensemble des services financiers. Dans ce cadre, les modèles de prédiction analysent l'historique de chaque client, afin de les classer selon la probabilité de chacun de rembourser leur(s) prêt(s) dans le temps fixé.

6. Une règle "minimale" (avec un antécédent petit) n'a aucun lien avec la mesure d'occurrence "minimale" utilisée pour identifier les occurrences des épisodes.

Nous allons maintenant présenter les approches de l'état de l'art utilisées pour créer des modèles prédictifs.

2.6.1 Les approches existantes

La classification est la technique de fouille de données la plus connue pour la génération de modèles descriptifs qui peuvent être utilisés comme des modèles prédictifs. Dans un modèle de classification, les étiquettes de la classe, qui représentent les informations ou les événements prédits, doivent être informatives et indicatives et non pas seulement descriptives de la classe. Prenons l'exemple de la classification de séquences de registre d'état pour les avions. Les étiquettes peuvent être : manque carburant, entretien moteur, nécessité de repos de l'équipage, non satisfaction de passagers, etc. Un modèle prédictif basé sur la classification permet donc de prédire l'étiquette (la classe) de nouvelles données.
La sélection des attributs les plus pertinents pour la détermination de la classe est un défi largement étudié dans le domaine de classification (Samb et al., 2012). Ces attributs peuvent représenter des événements pour la classification dans des séquences d'événements. Cependant, dans les séquences d'événements le nombre d'événements est très grand voire infini, ce qui rend, de notre point de vue, la tâche de sélection des attributs pour la classification très complexe voire inefficace en terme de temps de calcul.
L'approche appelée "classification associative" (Yin and Han, 2003; Zhang and Jiao, 2007) intègre classification et fouille de règles d'association. La classification associative utilise les algorithmes de fouille de règles d'association pour générer tout d'abord un ensemble complet de règles. Ensuite, les règles ayant les confiances les plus élevées sont sélectionnées pour construire le modèle de classification, de manière à ce que les conséquences des règles représentent les étiquettes des classes, donc utilisées dans un but de prédiction. Les approches de classification associative sont connues pour être plus performantes que les approches de classification classique (Yin and Han, 2003). Cependant, le grand nombre de règles générées, même si elles sont générées avec contraintes (et donc en nombre réduit), et l'effort requis pour sélectionner les meilleures règles représentent les limites les plus connues de ces approches. À noter que dans le cas de la classification, les conséquences des règles, qui représentent les étiquettes des classes, sont connues à l'avance, et donc les événements prédits, représentés par les conséquences des règles, sont également connus à l'avance, ce qui peut être une limite.
Depuis leur apparition, les techniques basées sur les règles d'association sont parfaitement employées pour la prédiction (Agarwal et al., 1999; Wang and Jafari, 2005; Jain et al., 2012; Abdi and Giveki, 2013). Ces techniques sont adaptées de manière à être capable de gérer le grand nombre d'événements (Wang et al., 2000), ce qui n'était pas le cas dans les techniques de classification. Par conséquent, elles sont plus adaptées pour la prédiction dans une séquence d'événements.
Rappelons qu'une règle d'épisode a le format suivant $R : P \rightarrow Q$ (voir section 2.5.2). Pour que cette règle soit exploitée dans un contexte de prédiction, on doit attendre l'apparition complète de l'antécédent P pour pouvoir prédire l'apparition de la conséquence Q. Par conséquent, la difficulté dans les modèles prédictifs à base de règles réside dans l'apprentissage du modèle, à savoir la construction de règles (comme nous l'avons vu dans la section précédente), et non pas dans l'utilisation de ces règles pour la prédiction. En plus, vu le grand nombre de règles construites, un nombre réduit de ces règles est en pratique utilisé pour la prédiction. C'est le cas dans de nombreux domaines où l'expert détermine la politique de choix de règles à utiliser pour prédire certains événements, comme par exemple les événements les plus probables (les conséquences des règles les plus confiantes), les événements les plus prédits (les conséquences de

plusieurs règles à la fois), etc.
Focalisons nous maintenant sur la prédiction par règles d'épisode dans une séquence d'événements. Comme nous l'avons mentionné précédemment, la tâche d'extraction de règles d'épisode est souvent composée de deux sous-tâches : l'extraction d'épisodes et la génération de règles. Une règle est générée en considérant quelques événements ou le dernier événement dans l'épisode comme la conséquence de la règle, le reste des événements composant son antécédent. Par conséquent, se baser sur ces deux sous-tâches génère des règles dans lesquelles la conséquence est proche temporellement de l'antécédent. Cela est dû au fait que l'épisode, qui est l'origine d'une règle, est composé d'événements relativement proches (voir section 2.5.2). Ces règles permettent donc de prédire des événements temporellement proches, ce qui est nécessaire pour de nombreuses applications. Nous pouvons par exemple citer la prédiction de comportement utilisateurs sur un site Web (Laxman et al., 2008), la prédiction de comportement client dans un magasin (Nakahara and Yada, 2012), etc.

2.6.2 Discussion

Comme nous l'avons mentionné ci-dessus, les approches proposées dans la littérature permettent de prédire les événements proches temporellement de l'occurrence de l'antécédent. Cependant, dans certaines applications, il est nécessaire de prédire des événements temporellement distants, voire loin dans le futur. C'est le cas lorsqu'il est nécessaire d'avoir une période de temps entre le moment où l'événement est prédit et son apparition effective. Prenons l'exemple d'une séquence de messages d'utilisateurs dans un réseau social où des règles d'épisode peuvent être extraites et utilisées pour prédire des informations sur les futurs messages de réclamation. Si les messages de réclamation sont prédits suffisamment tôt, il sera possible d'intervenir pour empêcher leur apparition par exemple. C'est également le cas pour la prédiction dans une séquence d'actions réalisées par un client d'une banque où les règles d'épisode extraites peuvent être utilisées pour prédire les cas de surendettement ou pour prédire les éventuelles demandes de prêt. Si ces messages sont prédits suffisamment tôt, il sera possible de réagir pour prévenir le surendettement en recommandant par exemple un prêt au client. De manière générale, lorsque la prédiction est réalisée tôt, un temps suffisamment long sera disponible afin de contrôler l'occurrence des événements prédits.
De nombreux algorithmes sont conçus pour extraire un ensemble complet d'épisodes ou de règles d'épisode (Huang and Chang, 2008). Ces algorithmes ne sont donc pas adaptés pour les données complexes car le nombre d'épisodes ou de règles d'épisode extraites est énorme, un grand effort sera donc consacré pour traiter et filtrer les épisodes ou les règles résultantes. Comme nous l'avons mentionné, l'imposition de contraintes (temporelles ou sur l'ensemble résultant) réduit le nombre final d'épisodes ou de règles d'épisode, ce qui paraît une solution évidente dans le cas de données complexes, face au grand volume de données. Concernant les contraintes temporelles, un gap *maximal* a été proposé dans la littérature afin de contraindre la distance entre les événements de l'épisode de manière à ce qu'ils soient proches l'un de l'autre, réduisant ainsi à la fois le nombre de règles et le temps d'exécution requis par l'algorithme. À notre connaissance, aucun algorithme ne propose un gap *minimal* entre les événements d'un épisode. Un gap minimal permet d'agrandir la distance entre les événements de l'épisode et peut être vu comme une piste pour la prédiction très tôt.
Comme nous l'avons mentionné, les règles minimales (voir section 2.5.5) ont été proposées dans la littérature pour réduire la consommation des ressources en temps et en mémoire au moment de la construction de règles et pour éviter la redondance au sein de l'ensemble final de règles (Neeraj and Swati, 2012). Rappelons qu'une règle est utilisée pour la prédiction de la manière suivante :

une fois que le préfixe de son antécédent apparaît, il faut attendre l'apparition de tous les événements de cet antécédent avant de pouvoir prédire l'apparition de la conséquence (avec une probabilité représentée par la confiance de cette règle). Par conséquent, de notre point de vue, les règles minimales peuvent être adéquates pour la prédiction au plus tôt dans certains cas, car peu d'événements sont à attendre (les événements à attendre correspondent ceux qui constituent l'antécédent) avant de pouvoir prédire la conséquence.
À notre connaissance, l'intégration de contraintes temporelles spécifiquement entre l'antécédent et la conséquence d'une règle d'épisode n'a jamais été proposée dans la littérature. Cependant, certains auteurs ont introduit l'extraction des motifs puis la concaténation de deux motifs pour former une règle en imposant un gap entre eux. De notre point de vue, cette approche n'est pas efficace car elle est réalisée en post-traitement et non pas au cours du processus de fouille des règles, ce qui consomme beaucoup de ressources. Rappelons qu'il est connu dans la littérature que l'intégration des contraintes au sein du processus de fouille est plus efficace que la réalisation de cette tâche durant une étape de post-traitement (Srikant et al., 1997). Comme nous l'avons mentionné, les mesures de fréquence proposées dans la littérature (voir section 2.5.3) définissent l'occurrence d'un épisode de manière à ce que ses événements soient les plus proches possibles l'un de l'autre. Comme la conséquence d'une règle est souvent composée des derniers événements de l'épisode (Mannila et al., 1997), la durée temporelle entre l'antécédent et la conséquence est plutôt petite. Pour élargir cette distance, et donc pour utiliser la règle pour prédire très tôt, la solution peut être d'imposer un gap minimal entre l'antécédent et la conséquence de la règle.
Lorsque les données sont complexes, comme les séquences d'événements, le grand volume et la richesse de données rendent la prédiction plus fiable (McAfee et al., 2012). Cependant, le nombre d'informations extraites (dans notre cas il s'agit de règles d'épisode) rend l'évaluation de la qualité de ces règles très difficile et exige souvent une intervention humaine experte afin de sélectionner les règles les plus pertinentes (cette tâche est généralement effectuée en définissant la politique de choix des règles à utiliser pour la prédiction).

2.7 Détection de l'émergence dans un flux d'événements

Dans cette section nous nous intéressons à la détection de l'émergence dans un flux d'événements. Dans un premier temps, nous allons détailler les techniques de fouille de flux d'événements. Dans un deuxième temps, nous nous focalisons sur la détection de l'émergence dans ces flux.

2.7.1 Les approches existantes

2.7.1.1 La fouille de flux d'événements

Comme nous l'avons mentionné dans la section 2.4.1.4, la fouille de flux de données est soumise à plusieurs contraintes comme l'impossibilité de stocker toutes les données et la nécessité de réaliser la fouille par un passage unique sur les données et en un temps impérativement court. Par conséquent, la fouille de flux de données est une tâche très complexe (Tseng et al., 2006; Gao et al., 2011) et les algorithmes traditionnels, conçus pour le traitement de séquences par exemple, ne sont pas capables de gérer ces contraintes. Ceci rend obligatoire la proposition d'algorithmes dédiés à la fouille de flux de données.
Soit un flux de données : $F =< (t_1, E_{t_1}), (t_2, E_{t_2}), ..., (t_n, E_{t_n}) >$ (avec $t_1 < t_2 < ... < t_n \wedge n \simeq \infty$), E est un élément (une donnée) qui peut représenter un texte, un graphe, un événement, etc. (voir définition 2.4.4). Dans ce manuscrit, nous choisissons de nous intéresser uniquement aux événements, nous allons donc étudier les flux d'événements.

Comme tout flux, un flux d'événements est infini, on ne peut donc pas le fouiller dans son ensemble. Il faut définir une technique de suivi de données dans le flux, à savoir diviser la tâche de fouille en sous-tâches, chaque sous-tâche s'intéressant à fouiller une partie du flux. Déterminer la partie du flux sur laquelle porte une sous-tâche représente un des défis du traitement de flux (Zhu and Shasha, 2003).

La littérature de la fouille de flux d'événements a montré que les approches basées sur des fenêtres sont les plus efficaces en terme de ressources consommées (Zhu and Shasha, 2003). Une fenêtre $Win(F, t_s, w)$ d'un flux F correspond à un instant de début t_s et possède une taille w qui permet de déterminer son instant de fin, comme nous l'avons défini dans la section traitant de la fouille de séquence d'événements (voir définition 2.5.8). Trois modèles de fenêtres sont proposés dans la littérature (Zhu and Shasha, 2002; Golab and Özsu, 2003; Csernel et al., 2005) :

— Une fenêtre glissante : la taille de la fenêtre est fixe, l'instant de début se décalant dans le temps chaque fois qu'un événement arrive. Dans ce cas, le traitement du flux est toujours réalisé sur la dernière position de la fenêtre glissante. On peut exprimer les événements dans une fenêtre grâce à l'ordre d'arrivée des événements du flux, comme par exemple une fenêtre portant sur les quinze derniers événements. On parle alors de fenêtre logique (Muthukrishnan, 2005). Cependant, lorsque plusieurs événements arrivent au même temps, la fenêtre peut porter sur plusieurs instants représentés par des secondes, des minutes, des jours, etc. Comme par exemple une fenêtre qui porte sur les trois derniers jours. On parle alors de fenêtre physique par opposition aux fenêtres logiques. À noter que les fenêtres physiques sont les plus souvent utilisées dans la littérature (Csernel et al., 2005).

— Une fenêtre point de repère : l'instant de début de la fenêtre est fixe, la taille de la fenêtre n'est pas fixe car l'instant de fin est représenté par l'instant actuel t_{now} (où "*now*" signifie maintenant). Par exemple une fenêtre entre le premier mars et maintenant.

— Une fenêtre amortie : est similaire à la fenêtre point de repère (la taille de la fenêtre n'est pas fixe, mais l'instant de début de la fenêtre est fixe), mais l'importance et l'impact des événements, généralement représentés par une mesure d'utilité ou de poids, diminue de façon exponentielle en fonction de leur ancienneté.

À noter que ce sont les fenêtres glissantes qui sont le plus souvent utilisées pour fouiller les flux d'événements (Csernel et al., 2005). De notre point de vue, ce type de fenêtre est plus efficace en terme de précision et de qualité de la fouille car toutes les informations sont traitées sans aucune perte, ce qui n'est pas le cas pour les autres types.

Confrontés à la taille infinie des flux d'événements, les algorithmes de fouille de données n'ont pas d'autre choix que de stocker des "résumés" du flux s'ils veulent permettre une analyse portant sur la totalité du flux (Csernel et al., 2005). Un résumé est une structure de données recensant les informations les plus importantes et essentielles du flux, c'est-à-dire les informations nécessaires pour comprendre le contexte du flux. Un résumé est mis à jour au fur et à mesure de l'arrivée d'événements dans le flux permettant de répondre *a posteriori* et approximativement à n'importe quelle requête portant sur le flux, et au mieux compte tenu des contraintes de ressources (Midas et al., 2010). La question qui se pose alors est quelle information stocker et comment adapter la capacité de stockage limitée au caractère infini du flux (Aggarwal et al., 2003). Stocker un résumé du flux semble une bonne solution pour les algorithmes de classification (Csernel et al., 2005). De notre point de vue, cela est valable uniquement si le résumé est pertinent, c'est-à-dire s'il contient tous les attributs nécessaires pour créer un modèle de classification efficace. Les algorithmes d'extraction de motifs ou de règles peuvent également se baser sur un résumé (en extrayant les motifs ou les règles uniquement à partir des informations contenues dans le résumé). Cependant, un résumé ne garantit pas de contenir toutes les occurrences des motifs ou des règles, ce qui amène à l'incomplétude du résultat final. Par ailleurs, les motifs ou les règles

résultant de ces algorithmes ont des valeurs de support approximatives (Giannella et al., 2003; Gaber et al., 2005).
L'extraction de motifs ou d'épisodes dans un flux d'événements a récemment suscité un grand intérêt (Töws et al., 2015; Yun and Lee, 2016). Prenons quelques exemples dans lesquels les fenêtres dites glissantes sont utilisées : (Chi et al., 2004) ont proposé un algorithme pour l'extraction de motifs fermés (rappelons qu'un motif est fermé si aucun des ses super-motifs n'a le même support que le sien, voir section 2.5.4) dans un flux d'événements en se basant sur des fenêtres glissantes et en utilisant un modèle d'arbre pour maintenir les informations sur tous les motifs candidats (fermés ou non fermés). Cependant, dans cet algorithme, les informations stockées sur les motifs candidats non fermés sont inutiles, ce qui augmente la consommation en temps et en mémoire. Toujours dans le cadre de l'extraction de motifs fermés, (Jiang and Gruenwald, 2006a) proposent un algorithme plus efficace en stockant uniquement les motifs fermés (qu'ils soient fréquents ou non) ce qui diminue les ressources consommées. (Gao and Wang, 2009) proposent un algorithme pour l'extraction de motifs fréquents en se basant sur une matrice afin de mettre à jour le support des motifs candidats au fil du temps. Cependant, ces deux derniers algorithmes ont également des limites en temps et en mémoire en cas d'utilisation d'un seuil de support bas.

2.7.1.2 La détection d'émergence

Comme nous l'avons mentionné (voir section 2.4.3), l'émergence d'événements fait partie des nouvelles connaissances fouillées dans les données complexes de type flux de données (ou d'événements). Un événement est dit émergent (Dong and Li, 1999) lorsque son support augmente de manière significative au fil du temps. Dans ce cas, il est possible de prédire que le support de cet événement va encore augmenter dans le temps.

Définition 2.7.1. *Soit $I_j = \{i_1, i_2, ..., i_j\}$ un événement qui apparaît dans le flux d'événements F. Soient $supp_{t'}(I_j)$ et $supp_{t''}(I_j)$ les supports de l'événement I_j jusqu'à l'instant t' et t'' respectivement. I_j est considéré comme un* **événement émergent** *à l'instant t'', si : $supp_{t''}(I_j) \gg supp_{t'}(I_j)$.*

L'augmentation du support d'un événement est souvent surveillée et donc traitée dans la littérature de deux manières : (i) soit l'événement émergent n'est pas encore fréquent à l'instant t'' : $supp_{t''}(I_j) < minsupp$, et donc on détecte qu'il va devenir fréquent ; (ii) soit l'événement émergent est déjà fréquent à l'instant t'' : $supp_{t''}(I_j) >= minsupp$ (ou tout simplement on ne s'intéresse pas au fait qu'il est fréquent) et on détecte qu'il va être encore plus fréquent. Ce deuxième cas est le plus souvent traité dans la littérature.
Plusieurs techniques ont été proposées pour détecter les événements émergents, comme les techniques basées sur le taux de croissance, représenté par le rapport du support de l'événement à deux instants différents (Dong and Li, 1999; Zhu et al., 2015). Si ce taux dépasse un seuil prédéfini, l'événement est considéré comme émergent.

Définition 2.7.2. *Le* **taux de croissance** *de l'événement I_j entre les instants t' et t'' est dénoté : $tauxCroissance_{t',t''}(I_j) = supp_{t''}(I_j)/supp_{t'}(I_j)$. I_j est considéré comme un événement émergent lorsque : $tauxCroissance_{t_i,t_j}(I) >= min_{croissance}$, où $min_{croissance}$ est un seuil de croissance prédéfini.*

Il est important de mentionner que le seuil minimal de croissance et le temps entre les deux instants considérés pour détecter l'émergence sont des paramètres liés à l'application. Cela permet de détecter une émergence forte (seuil élevé) ou une émergence faible (seuil bas, et les deux instants éloignées).

(Luhr et al., 2005) proposent de détecter l'émergence de règles d'association en se basant, comme pour la détection de l'émergence d'événements, sur le taux de croissance. Dans l'algorithme proposé, les règles fréquentes et confiantes dans un premier temps sont extraites à partir d'un corpus de test (c'est-à-dire que les règles apprises). Puis, ces règles (connues à l'avance) sont surveillées dans le flux afin de détecter l'instant auquel elles émergent. Cette approche représente la solution classique pour la détection de l'émergence de règles.

2.7.2 Discussion

Dans de nombreuses applications, les effets consécutifs à l'émergence sont souvent connus à l'avance par des experts humains. Citons par exemple un cas classique : on sait pertinemment que l'émergence de symptômes va avoir pour effet le déclenchement d'une maladie et on sait qu'il faut faire intervenir des experts pour traiter ces effets (Wong et al., 2005). Cependant, dans d'autres contextes, ces effets sont moins maîtrisés. Dans le cadre des réseaux sociaux par exemple, les effets liés aux événements peuvent être très vastes et nombreux. Dans ce cas, nous considérons qu'il faut détecter non seulement l'émergence d'événements mais également l'émergence d'associations d'événements (des règles d'association) qui représentent l'émergence des liens entre les événements (l'antécédent de règle) et l'effet qu'ils produisent (la conséquence de règle).
Revenons au cas de détection de l'émergence de règles où les règles apprises auparavant sont surveillées dans le flux. Notons que surveiller uniquement les règles déjà apprises ne permet pas d'apprendre de nouvelles règles ni de détecter l'émergence de nouvelles associations (règles) non vues auparavant, ce qui ne permet pas d'anticiper l'occurrence de certains événements (ceux qui n'ont pas été appris au travers des règles initiales). D'autre part, un taux de croissance élevé, qui représente une augmentation significative du support de la règle, ne signifie pas forcément que la règle reste confiante. De notre point de vue, cette technique de détection de l'émergence de règles ne permet pas de garantir que les règles dites émergentes sont toujours confiantes au moment de leur détection, ce qui est cependant primordial.
Nous avons précédemment cité les techniques de fouille de flux d'événements qui s'appuient sur un résumé du flux, ces techniques souffrent le plus souvent de l'incomplétude de l'ensemble de résultat. Elles ne sont donc pas adaptées à la détection de l'émergence dans un flux de données. En effet, la détection de l'émergence et particulièrement de l'émergence très tôt et avec un faible support, exige de considérer chaque occurrence du motif ou de la règle dans le flux car chaque occurrence est importante et peut être celle qui permet de dire que le motif ou la règle est émergent (Gao et al., 2011).

2.8 Influencer les événements futurs

Dans la littérature, peu de domaines s'intéressent à la tâche d'influencer des événements futurs. Nous pouvons citer par exemple le domaine de l'automatique (Christen and Busch, 2015) pour influencer une trajectoire (Lu, 1997; Morris and Smith, 2008; Xiong et al., 2013) : pour le décollage d'une fusée on doit asservir tous les moteurs de la fusée pour qu'elle respecte bien sa trajectoire pour sortir de l'attraction terrestre. Cependant, dans le domaine de fouille de données, qui est le domaine qui nous intéresse, très peu de travaux se sont penchés sur cette tâche. Le seul lien que nous pouvons trouver entre la fouille de données et l'influence des événements réside dans l'extraction de motifs dits importants, d'une grande utilité ou intéressants. Pour cela, dans la suite de cette section, nous allons détailler les travaux de la littérature dédiés à l'extraction de tels motifs que nous appellerons motifs d'une grande utilité, pour simplifier. Présentons maintenant la définition de ces motifs (Kim and Yun, 2016).

Définition 2.8.1. *L'utilité d'un item est une valeur numérique initialement fixée par l'utilisateur. L'utilité d'un motif représente la somme des utilités de ces items. Lorsque cette utilité dépasse un seuil prédéfini, le motif est considéré comme un* **motif d'une grande utilité**.

De notre point de vue, un motif d'une grande utilité peut influencer les événements futurs de la manière suivante : lorsque l'on apprend, par un expert ou au travers de règles par exemple, que la présence de ce motif dans les données peut maximiser l'objectif final, à savoir influencer un futur événement, on peut envisager d'injecter ce motif dans les données arrivantes afin d'influencer le futur événement.
Dans de nombreuses applications, des informations, qu'elles soient des items, des motifs, des événements, des règles, etc. sont reconnues comme d'une grande utilité (Hu and Mojsilovic, 2007). Pour cela, il est important pour l'application de surveiller, de détecter l'occurrence, ou de prédire ces informations afin de maximiser voire de "contrôler" l'objectif final lié à l'application, comme par exemple l'augmentation de la rentabilité d'un magasin, l'augmentation de la satisfaction des clients dans un site Web, etc.
Bien que la fouille de motifs et de règles joue un rôle très important dans de nombreuses applications de la fouille de données, elle est confrontée à plusieurs limites (Ahmed et al., 2009) : (i) les algorithmes de fouille de motifs considèrent que tous les items ont la même importance, le même poids, la même utilité etc. (ii) ces algorithmes prennent uniquement en compte l'apparition ou non d'un item, et ne considèrent pas d'autres informations liées à cet item, comme son poids prédéterminé ou son influence sur les autres items. Ces limites contraignent donc les avantages apportés par les motifs. Prenons l'exemple du panier de la ménagère, chaque item dispose d'une fréquence d'achat et d'une rentabilité qui lui sont propres, certains items fréquents peuvent ne pas être rentables ou au contraire très rentables. Par conséquent, l'extraction de motifs fréquents sur la seule base de leur fréquence ne suffit dans le cadre d'application visant à maximiser la rentabilité. Par conséquent, il devient important d'extraire les motifs les plus utiles, parmi les motifs fréquents. Cela motive la proposition de nombreux modèles pour l'extraction de motifs qualifiés par la littérature de motifs d'une grande utilité (Ahmed et al., 2011; Tseng et al., 2013; Lin et al., 2015a), où la définition du degré d'intérêt ("interestingness") (Tan et al., 2002; Freitas, 1999) ou d'utilité ("utility") (Hu and Mojsilovic, 2007) d'un motif dépendent de la formulation de la problématique et de la connaissance du domaine d'application. Il est important de noter que les motifs d'une grande utilité sont à l'origine détectés dans des données transactionnelles (Tseng et al., 2010; Liu and Qu, 2012).
Nous allons maintenant présenter les travaux de l'état de l'art liés à la détection des motifs d'une grande utilité.

2.8.1 Les approches existantes

Plusieurs chercheurs se sont intéressés à la détection des motifs d'une grande utilité dits pondérés où chaque motif est associé à un poids représenté par le poids (prédéterminé) de chacun de ses items (Yun and Ryu, 2013; Yun et al., 2015; Nguyen, 2015).

Définition 2.8.2. *Soit $I_j = \{i_1, i_2, ..., i_j\}$ un motif. Les items le composant sont associés respectivement à des poids prédéterminés : $\{poids_{i_1}, poids_{i_2}, ..., poids_{i_j}\}$. Le motif peut être considéré comme un* **motif pondéré**, *où son poids est calculé au travers de l'équation (2.3) :*

$$poids(I_j) = \frac{poids_{i_1} + poids_{i_2} + ... + poids_{i_j}}{j} \tag{2.3}$$

Dans ce cas, le support de chaque motif devient fonction des poids de ses items, et se définit comme la multiplication du support traditionnel du motif par son poids (définit par l'équation 2.3) (Cai et al., 1998). Le support d'un motif est donc défini comme suit (équation 2.4) :

Définition 2.8.3. *Le* **support d'un motif pondéré** *est calculé comme suit :*

$$supp_{pondr}(I_j) = supp(I_j) \times poids(I_j) \tag{2.4}$$

Pour extraire les motifs pondérés, une technique reposant sur la notion de "transactions pondérées" est souvent appliquée sur les bases de transactions (Wang et al., 2004; Liu et al., 2005; Ahmed et al., 2009). Cette technique consiste tout d'abord à extraire les motifs fréquents (de manière traditionnelle sans leur poids), ensuite à effectuer une étape de post-traitement : calculer le poids de chaque transaction comme la somme des poids des items la composant (de la même façon que le poids d'un motif est calculé), puis identifier les motifs candidats d'une grande utilité si la somme des poids de leur transaction (c'est-à-dire les transactions dans lesquelles les motifs apparaissent) dépasse un seul minimum (ce qui représente le poids maximum du motif), ensuite calculer le poids exact de chaque motif candidat afin de ne retenir que les motifs d'une grande utilité. Rappelons qu'une étape de post-traitement est reconnue comme étant peu efficace en temps.
Plusieurs algorithmes qui n'exigent pas une étape de post-traitement ont été proposés dans la littérature pour extraire les motifs pondérés. Par exemple, certains algorithmes exigent plusieurs passages (ce qui reste tout de même peu efficace) sur les données (Yun and Leggett, 2005), d'autres algorithmes traitent un flux de données en affectant un poids plus important aux items les plus récents (Zhang et al., 2003; Jiang and Gruenwald, 2006b), ou encore certains algorithmes laissent le choix à l'utilisateur de fixer le poids des items les plus récents (Tsai, 2009). Récemment, plusieurs travaux ont proposé de considérer qu'un poids évolue dans le temps (Nguyen, 2015), ce qui reflète la réalité dans plusieurs applications, comme par exemple le poids représenté par le prix d'un item qui peut ne pas être stable et qui change au fil du temps.
Dans la littérature, un motif d'une grande utilité peut également être caractérisé par des mesures statistiques (Li et al., 2009; Li et al., 2005) comme le rapport de chances (appelé aussi rapport de cotes ou odds ratio), le risque relatif, etc. L'utilisation de ces mesures requiert plusieurs bases de transactions, ce qui limite leur application aux séquences ou aux flux. Rappelons que les motifs d'une grande utilité sont à la base détectés dans des données transactionnelles (Tseng et al., 2010; Liu and Qu, 2012). Cependant, l'intégration de l'utilité dans des données séquentielles (Ahmed et al., 2010) et des séquences d'événements (Wu et al., 2013) a été récemment explorée.
En règle générale, les applications s'intéressent à détecter l'ensemble des motifs qui maximise une fonction objectif liée à la problématique de l'application (Hu and Mojsilovic, 2007). Prenons un exemple dans le domaine des chaînes de production où plusieurs composants (qui représentent les items ou les événements) forment un produit final. Il est possible que l'application ne s'intéresse pas à la détection des composants individuellement utiles, mais plutôt à la détection d'un ensemble utile de composants qui maximise la probabilité d'avoir un produit final de bonne qualité, de manière à ce que cet ensemble représente l'antécédent d'une règle et le produit final représente sa conséquence. Dans la littérature, plusieurs modèles ont été proposés pour atteindre cet objectif (Yao et al., 2004; Liu et al., 2005) souvent basés sur un processus de jointure entre plusieurs items pour évaluer l'utilité de leur ensemble. Cependant, la détection des ensembles utiles de motifs est connue comme étant un problème très complexe en raison du grand nombre de jointures à effectuer (Hu and Mojsilovic, 2007), c'est un problème combinatoire. Ces modèles nécessitent l'élaboration d'heuristiques pour atteindre leur objectif en consommant moins de ressources. Certains chercheurs proposent de détecter uniquement les k meilleurs ensembles

utiles de motifs pour réduire les ressources consommées (Chan et al., 2003). D'autres proposent l'utilisation des structures d'arbres pour réduire les calculs non nécessaires, lors de la mise-à-jour de données par exemple (Ahmed et al., 2009).
Récemment, plusieurs travaux se sont intéressés à l'extraction des épisodes d'une grande utilité dans une séquence d'événements (Guo et al., 2012; Wu et al., 2013; Guo et al., 2014; Lin et al., 2015b). Cependant, intégrer la notion d'utilité dans le processus de la fouille d'épisodes est une tâche qui comporte beaucoup de défis (Wu et al., 2013) :

— L'utilité d'un épisode n'est pas forcément une caractéristique anti-monotone, elle peut être égale, plus grande, ou plus petite que l'utilité de ses super-épisodes ou de ses sous-épisodes. Par conséquent, les heuristiques basées sur l'anti-monotonicité (comme par exemple le fait de ne pas extraire les super-épisodes lorsque l'épisode n'est pas fréquent) habituellement appliquées pour élaguer l'espace de recherche ne sont pas adaptées pour la fouille des épisodes d'une grande utilité.
— Dans une séquences d'événements, plusieurs événements issus de plusieurs sources peuvent apparaître en même temps (au même instant). Par conséquent, les contextes de ces événements sont parfois mélangés et chevauchés, ce qui complexifie la tâche de la fouille des épisodes d'une grande utilité.
— La fouille des épisodes étant réalisée sur une séquence d'événements unique et longue, voire infinie (dans le cas d'un flux), l'adaptation de la technique la plus souvent utilisée pour extraire les motifs pondérés, à savoir l'utilisation des transactions pondérées, est difficile dans le cas d'une séquence unique, car cette technique est adaptée pour la recherche des motifs dans les transactions (une unique occurrence est recherchée par transaction).
— Dans une séquence d'événements, le nombre d'épisodes candidats peut être tel qu'il devient ingérable dans le cas de l'application de ces techniques traditionnelles qui reposent sur la génération de candidats. Pour cela, il faut concevoir des heuristiques pour diminuer le nombre de candidats pendant le processus de la fouille des épisodes.

Pour répondre à ces limites, plusieurs algorithmes ont été proposés. (Lin et al., 2015b) introduisent une structure appelée "arbre de règles d'une grande utilité" pour stocker les informations importantes sur les événements afin d'extraire les règles d'une grande utilité sans passer par une étape de génération de candidats. (Wu et al., 2013) proposent une technique appelée l'utilisation des épisodes pondérés. L'algorithme fait un passage sur la séquence d'événements pour extraire les épisodes de taille 1 et calcule directement leur poids, puis d'autres passages sur la séquence sont réalisés afin d'extraire les épisodes les plus longs et de calculer leur poids. Cet algorithme n'est pas adapté en cas d'une séquence très longue voire infinie (flux), car il exige plusieurs passages sur la séquence.

2.8.2 Discussion

Les travaux précédemment cités se focalisent sur l'extraction, soit des motifs (ou épisodes) individuellement d'une grande utilité, soit des ensembles utiles de motifs. Ces motifs sont souvent extraits dans le but d'analyser les données par l'identification des combinaisons utiles et intéressantes qui sont ensuite interprétées par un expert du domaine et non pas pour des objectifs liés aux événements futurs ou à influencer des événements. Il serait opportun que les événements d'une grande utilité extraits dans la littérature puissent être exploités pour influencer les événements futurs : en les injectant dans les données pour maximiser l'objectif final de l'application.
À notre connaissance, aucun travail ne s'intéresse à l'utilité portée par un motif (ou épisode) sur un autre motif (ou épisode), ou à l'utilité portée par un item (ou un événement) sur les autres items d'un motif (ou d'un épisode). Cela pourrait être intéressant dans le cadre d'une application

qui cherche à atteindre plusieurs objectifs à la fois. Dans ce cas, il est possible d'extraire les items d'une grande utilité liés à chaque objectif, en prenant en compte le fait qu'un item est utile pour atteindre un seul objectif et qu'il n'est pas forcément utile pour atteindre les autres objectifs.

2.9 Plateformes de traitement de séquences et de flux

Nous avons présenté dans les sections précédentes les algorithmes proposés dans la littérature pour fouiller les séquences et les flux d'événements. Plusieurs outils ont été développés dans l'état de l'art pour faciliter l'exécution des algorithmes de fouille : soit en implémentant ces algorithmes, soit en donnant la possibilité à un utilisateur d'implémenter lui même son propre algorithme et de faciliter l'importation/exportation de données et la visualisation de résultats. Parmi ces outils, nous pouvons citer *WEKA* (Holmes et al., 1994) pour le traitement de séquence (processus de lots "batch") et *MOA* (Bifet et al., 2010) pour le traitement de flux (voir figure 2.3, branche de droite).
Cependant, le traitement lourd d'une séquence et le volume accompagné de la vélocité d'un flux représentent les principaux défis pour leur fouille. Plusieurs plateformes de traitement distribué de séquence et de flux ont été donc développées. La plateforme *Hadoop* (Apache Hadoop, 2005) est une plateforme qui implémente l'architecture *MapReduce* (Dean and Ghemawat, 2008) où dans cette dernière sont effectués des calculs distribués de données volumineuses. Parmi les outils exécutables sur *Hadoop* qui rassemblent de nombreux algorithmes de fouille de données, nous pouvons citer *Mahout* (Apache Mahout, 2011) (voir figure 2.3).

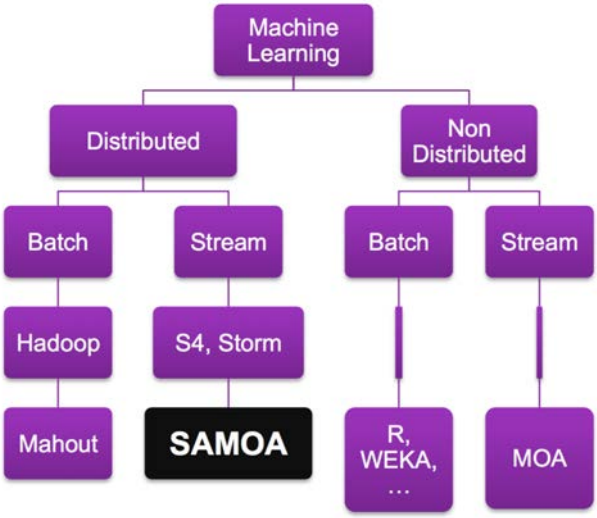


FIGURE 2.3 – Schéma des outils de fouille de données (De Francisci Morales, 2013).

MapReduce est dédié à la base au traitement des séquences (en mode batch), il est relativement peu efficace sur le traitement de flux de données, car les données du flux peuvent être beaucoup plus véloces que le traitement effectué par *MapReduce* (Feldman et al., 2010). Plusieurs extensions de *Hadoop* dédiées aux flux de données ont été proposées, comme par exemple *HaLoop* (Bu et al., 2010) et *DEDUCE* (Kumar et al., 2010). Cependant, ces plateformes restent des hybridations qui comportent toujours des limites, plutôt que de nouvelles approches innovantes dédiées pour le traitement de flux (De Francisci Morales, 2013).
Par conséquent, plusieurs plateformes ont été proposées dans la littérature pour le traitement

distribué de flux de données (voir figure 2.3). Nous pouvons notamment citer : $S4$ (Neumeyer et al., 2010) et $STORM$ (Apache Storm, 2012), qui sont toutes les deux inspirées de $MapReduce$. Parmi les outils qui s'exécutent sur $STORM$, nous pouvons citer $SAMOA$ (De Francisci Morales, 2013) qui regroupe plusieurs algorithmes de fouille de flux.
Les plateformes et les outils de traitement distribué de flux sont largement exploités dans l'état de l'art. De nombreux algorithmes de fouille de flux ont été conçus pour s'exécuter sur ces plateformes et outils (Aniello et al., 2013; Yang et al., 2013; Bifet and Morales, 2014; Prasad and Agarwal, 2014).
Nous allons maintenant présenter en détail les contributions apportées dans ce manuscrit. Nous commençons par présenter l'étude que nous menons dans le but de prédiction au plus tôt dans une séquence d'événements, ce qui sera présenté dans le chapitre suivant.

Chapitre 3

Un algorithme de fouille de règles d'épisode : DEER

Sommaire

3.1 Introduction **51**
3.1.1 Les défis de la prédiction d'événements distants 52
3.1.2 Principe de l'algorithme *DEER* 54
3.2 Nouveaux concepts et redéfinitions **56**
3.3 Les étapes de l'algorithme *DEER* **58**
3.3.1 Identification du préfixe 59
3.3.2 Identification de la conséquence 59
3.3.3 Complétion de l'antécédent 60
3.3.4 La confiance temporelle 61
3.4 Discussion comparative : efficacité de *DEER* en terme de temps d'exécution **64**
3.4.1 Choix d'algorithme de l'état de l'art 64
3.4.2 Comparaison sur des exemples 65
3.4.3 Synthèse : *DEER* vs. un algorithme de la littérature 66
3.5 Expérimentations **66**
3.5.1 Caractéristiques des règles extraites 67
3.5.1.1 Phase d'initialisation 67
3.5.1.2 Impact de seuil de support et de confiance sur les règles extraites 67
3.5.1.3 Impact de gap inséré entre l'antécédent et la conséquence de règles 70
3.5.2 Évaluation de la confiance temporelle 70
3.5.3 Performance en prédiction 72
3.5.4 Comparaison des règles formées par *DEER* à celles de l'état de l'art . 73
3.5.5 Temps d'exécution 74
3.5.6 Discussion 75
3.6 Synthèse **76**

3.1 Introduction

Dans ce chapitre, nous nous intéressons à la prédiction d'événements distants. Comme nous l'avons présenté dans le chapitre précédent, les règles d'épisode extraites à partir d'une séquence

d'événements peuvent être exploitées pour la prédiction d'événements : les conséquences des règles sont prédites (Mannila et al., 1997; Achar et al., 2012). La prédiction des événements est un moyen d'effectuer des recommandations (Letham et al., 2013), c'est-à-dire de recommander soit un événement prédit soit un autre événement selon le contexte applicatif. Prenons quelques exemples :

— Dans le contexte de l'analyse de comportement utilisateurs dans un site Web, des règles d'épisode peuvent être extraites de l'historique d'interactions entre les utilisateurs (qui peut représenter une séquence d'événements) (Laxman et al., 2008) et donc être utilisées pour prédire le comportement utilisateurs. Cette prédiction peut à son tour être exploitée pour recommander des pages Web à précharger par le navigateur Web.
— Dans le contexte de l'analyse de trafic routier (Cho et al., 2008), les règles extraites (à partir de la séquence d'informations de trafic) peuvent être utilisées pour prédire l'embouteillage dans certaines rues. Des recommandations peuvent donc être effectuées afin de limiter la circulation dans les rues amenant à celles où l'embouteillage a été prédit.
— Dans l'analyse de données dans une chaîne de production (Laxman et al., 2009), les règles peuvent être utilisées pour prédire l'état du produit final (à partir de la séquence composée des composants ajoutés et/ou des processus effectués), et donc des solutions peuvent être recommandées afin d'améliorer le produit final et ainsi augmenter la rentabilité.

Rappelons que la tâche d'extraction de règles d'épisode est souvent composée de deux sous-tâches (Mannila et al., 1997) : l'extraction d'épisodes fréquents et l'extraction de règles confiantes. Les règles sont générées en considérant quelques événements dans l'épisode ou les derniers événements comme la conséquence de la règle, et le reste des événements comme son antécédent. Former une règle en se basant sur l'extraction d'épisodes fréquents permet d'extraire des règles dont la conséquence est proche de l'antécédent. Ces règles permettent donc de prédire des événements dans un futur proche, ce qui est pertinent pour de nombreuses applications. De plus, fouiller des règles dans lesquelles les événements sont proches a l'avantage de diminuer la complexité de algorithme en limitant la consommation de ressources en temps et en mémoire pendant le processus de la fouille.

3.1.1 Les défis de la prédiction d'événements distants

Nous allons maintenant présenter les défis auxquels nous nous sommes confrontés dans le contexte de la prédiction d'événements distants en se basant sur les règles d'épisode. Nous présentons également les besoins applicatifs qui nous amènent à répondre à ces défis.

Défi 1 : La fouille de règles distantes : Dans de nombreuses applications, il est préférable de prédire des événements distants, à savoir prédire loin dans le futur. C'est notamment le cas lorsqu'il est important ou nécessaire d'avoir du temps avant l'apparition effective de l'événement prédit. Afin d'expliquer cette nécessité de prédiction des événements distants, prenons deux exemples :

— Rappelons qu'une suite de messages d'utilisateurs dans un blog dans le domaine bancaire peut être considéré comme une unique séquence de messages et donc des règles d'épisode peuvent être fouillées et utilisées pour la prédiction de futurs messages (futurs événements). Parmi les prédictions les plus intéressantes, nous pouvons citer celles liées au surendettement des clients. Si la prédiction est effectuée très tôt, la banque concernée aura du temps pour réagir et empêcher le surendettement effectif du client.
— Considérons à nouveau l'exemple sur la prédiction de l'état final d'un produit dans une chaîne de production. Supposons que vingt jours soient nécessaires pour fabriquer un

produit, la fabrication étant une suite d'événements journaliers (représentés par des composants ou des processus particuliers). Il peut être intéressant de prédire la défaillance dans l'état du produit final très tôt, afin de pouvoir rectifier rapidement cette défaillance, ce qui permettra de réduire le coût de production.

Nous nous intéressons donc à la prédiction d'événements distants au travers de la fouille de règles d'épisode. Nous considérons que, afin de prédire des événements distants, les règles fouillées doivent refléter des relations distantes, et plus particulièrement une relation distante entre leur antécédent et leur conséquence. Nous sommes donc confrontés au défi de **la fouille de règles d'épisode avec une conséquence distante**, que nous appellerons *règles d'épisode distantes*.

Voici un exemple d'une règle pertinente pour ce défi : *[problème de carte bancaire, s'intéresser à une offre d'une banque concurrente, relation tendue avec le conseiller bancaire]* $\rightarrow$ *[demande de fermeture de compte]* avec une distance de *15 jours* entre l'antécédent et la conséquence de la règle. Cette règle signifie que lorsqu'un client rencontre un problème de carte bancaire, et qu'il s'intéresse à une offre proposée par une banque concurrente, en plus de sa relation tendue avec son conseiller bancaire, ce client va probablement fermer son compte bancaire dans sa banque actuelle après 15 jours. Cette règle est intéressante car elle offre la possibilité de prédire l'événement *[demande de fermeture de compte]* 15 jours avant la réelle apparition de cet événement. Par conséquent, la banque dispose de 15 jours pour réagir afin d'empêcher l'apparition de cet événement.

Contrairement à la règle d'épisode précédente, la règle : *[problème de carte bancaire, s'intéresser à une offre d'une banque concurrente, ne pas répondre aux courriels du client, relation tendue avec le conseiller bancaire, conversation téléphonique tendue]* $\rightarrow$ *[demande de fermeture de compte]* avec une distance de *2 jours* , n'est pas une règle qui nous intéresse, car la prédiction sera effectuée trop tard, et la banque n'aura pas le temps suffisant (2 jours) pour empêcher la demande de fermeture de compte client.

Les algorithmes traditionnels de l'état de l'art ne sont pas adaptés à la fouille de règles distantes, car les règles formées ont une conséquence proche de l'antécédent De plus, ces algorithmes ne peuvent pas être facilement modifiés pour former des règles distantes. En effet, lors de la construction des épisodes lors de l'ajout d'un nouvel événement, l'algorithme ne peut pas savoir si cet événement fera partie de la conséquence ou de l'antécédent de la règle formée à partir de l'épisode. Par conséquent, un tel algorithme ne peut pas décider s'il doit contraindre la distance entre l'événement ajouté à l'épisode et le reste de l'épisode. Le seul moyen pour ces algorithmes traditionnels pour fouiller des règles avec une conséquence distante repose sur deux étapes : (i) fouiller tous les épisodes et former toutes les règles confiantes en appliquant un grand span pour borner l'occurrence des épisodes et donc des règles épisode, (ii) filtrer les occurrences de ces règles en éliminant celles qui ne respectent pas la distance minimale. Cette manière de fouiller des règles distantes consomme énormément de ressources en temps en raison du large span appliqué et du post-traitement.

Défi 2 : La fouille de règles essentielles : En plus de la prédiction d'événements distants par règles d'épisode, nous nous intéressons également à effectuer cette prédiction au plus tôt, afin d'avoir un temps maximum pour réagir une fois la conséquence prédite.

Considérons la règle précédente : R : *[problème de carte bancaire, s'intéresser à une offre d'une banque concurrente, relation tendue avec le conseiller bancaire]* $\rightarrow$ *[demande de fermeture de compte]* avec une distance de *15 jours*. Considérons une autre règle R' avec un antécédent plus petit : R' : *[problème de carte bancaire, s'intéresser à une offre intéressant d'une banque concurrente]* $\rightarrow$ *[demande de fermeture de compte]*, pour laquelle l'antécédent est composé uniquement des deux premiers événements de l'antécédent de la règle R. Si R' est également une règle ca-

pable à prédire de manière confiante l'événement *[demande de fermeture de compte]*, dans ce cas nous considérons que R' est beaucoup plus utile que R. En effet, nous pouvons remarquer que la distance temporelle entre l'antécédent et la conséquence dans la règle R' est par définition plus grande que celle dans la règle R, car il n'est pas utile d'attendre l'apparition du troisième événement de l'antécédent *[relation tendue avec le conseiller bancaire]* avant de pouvoir prédire la conséquence. Par conséquent, en utilisant la règle R', la conséquence peut être prédite plus tôt qu'avec la règle R. Nous considérons que, dans ce cas, un antécédent plus petit, en terme de nombre d'événements, permet de trouver des occurrences de la règle avec une distance plus grande entre l'antécédent et la conséquence, et donc de prédire la conséquence au plus tôt.
Nous nous sommes donc confrontés au défi de **la fouille de règles d'épisode avec un antécédent le plus petit possible**, c'est-à-dire un antécédent composé d'un nombre minimum d'événements. Nous proposons d'appeler ces règles les *règles essentielles*.
Comme nous l'avons mentionné, les algorithmes traditionnels extraient des règles minimales en terme de temps, c'est-à-dire en appliquant une mesure de fréquence basée sur l'occurrence minimale et un span qui borne l'occurrence de la règle. Cependant, ces algorithmes ne sont pas capables de garantir la construction d'un antécédent minimal en nombre d'événements au sein de la règle. Rappelons que ces algorithmes déterminent les événements faisant partie de l'antécédent et ceux faisant partie de la conséquence une fois l'épisode concerné est entièrement formé. Par conséquent, il est impossible de contraindre le nombre d'événements dans l'antécédent durant le processus de la fouille. Pour extraire telles règles minimales, les algorithmes traditionnels doivent se reposer à nouveau sur une étape de post-traitement afin de filtrer les règles extraites, ce qui n'est pas efficace en terme de ressources consommées.

3.1.2 Principe de l'algorithme $DEER$

Pour atteindre l'objectif de fouille de règles d'épisode distantes et essentielles (défis 1 et 2), nous partons de hypothèse suivante :
Hypothèse : Il existe une dépendance temporelle, distante ou proche, entre les événements dans une séquence d'événements. Cette dépendance peut être exploitée et intégrée dans la modélisation des associations entre des événements.
En partant de cette hypothèse, tout en évitant de passer par une étape de post-traitement très consommatrice en temps, nous proposons un nouvel algorithme appelé $DEER$: Distant and Essential Episode Rules. $DEER$ n'utilise ni une étape de post-traitement ni l'étape traditionnelle d'extraction d'épisodes.
L'originalité de l'algorithme $DEER$ réside dans le moment où la conséquence de la règle est identifiée : très tôt dans le processus. C'est cette caractéristique qui permet de garantir que la conséquence est distante de l'antécédent. Cette originalité permet également l'évaluation de la confiance de chaque règle durant le processus de la fouille car la conséquence est connue. Elle permet aussi de former des règles essentielles, sans s'appuyer sur une étape de post-traitement. De plus, nous estimons que l'identification de la conséquence très tôt dans le processus de la fouille entraînera une diminution remarquable du temps d'exécution de l'algorithme comparé aux algorithmes de l'état de l'art, car les occurrences non adéquates de la règle (dans lesquelles la conséquence ne respecte pas la distance requise) seront filtrées tôt dans le processus. D'autre part, nous choisissons de fouiller des règles avec une conséquence composée d'un seul événement, ce qui est commun dans la littérature (Agrawal et al., 1993). À noter que l'algorithme $DEER$ peut facilement s'adapter à la fouille des règles avec une conséquence composée de plusieurs événements.
$DEER$ fouille des règles d'épisode en effectuant un parcours de recherche en profondeur en pro-

cédant en quatre étapes :
Premièrement, le préfixe de la règle est fixé, tout comme dans les algorithmes traditionnels.
Deuxièmement, la conséquence est fixée, ce qui permettra non seulement de contrôler sa distance à l'antécédent, mais également de former des règles essentielles et de calculer leur confiance durant tout le processus de la fouille.
Troisièmement, l'antécédent est complété en ajoutant itérativement des événements à sa droite, tout comme dans les algorithmes traditionnels. À n'importe quelle étape de l'algorithme, le support de la règle d'épisode est évalué. Étant donné que la conséquence de la règle est également connue à ce stade, la confiance peut également être évaluée, ce qui n'est pas réalisable par les algorithmes traditionnels. Si la règle d'épisode est fréquente et confiante, la complétion de l'antécédent s'arrête, et la règle formée est ajoutée à l'ensemble résultat de règles d'épisode. Si la règle est non fréquente, l'itération s'arrête et la règle est écartée. Si la règle est fréquente mais non confiante, la complétion de l'antécédent continue dans la limite d'un span maximal prédéfini (afin de garantir une distance maximale entre l'antécédent et la conséquence) jusqu'à avoir une règle confiante (la règle sera ajoutée à l'ensemble résultat de règles) ou une règle non fréquente (la règle sera écartée). Cette stratégie garantit de former des règles essentielles et distantes.
Quatrièmement, la distance entre l'antécédent et la conséquence est exploitée par une nouvelle mesure de confiance : la confiance temporelle. À partir de cette mesure, les occurrences de la règle d'épisode sont filtrées afin de maintenir celles dans lesquelles la conséquence apparaît seulement à la distance requise de l'antécédent. Cette étape est optionnelle, elle est en fonction des exigences applicatives, comme sera détaillé après.

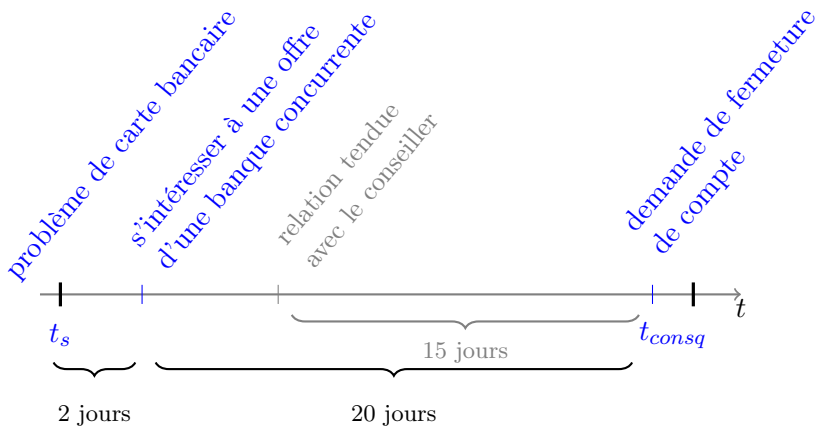


FIGURE 3.1 – Un exemple d'une règle essentielle avec une conséquence distante : *R : [problème de carte bancaire, s'intéresser à une offre d'une banque concurrente]* → *[demande de fermeture de compte]*; la conséquence est 20 jours loin de l'antécédent, la règle est essentielle car il n'y a pas besoin d'ajouter l'événement *relation tendue avec le conseiller* (en couleur gris) pour réaliser une prédiction fiable.

Dans la figure 3.1, nous présentons un exemple de règle (en couleur bleue) que nous souhaitons extraire. À noter que cette règle a été extraite à partir d'une séquence de messages utilisateurs étiquetés issus d'un blog discutant de sujets financiers et bancaires, où un message est représenté par plusieurs étiquettes. *R : [problème de carte bancaire, s'intéresser à une offre d'une banque concurrente]* → *[demande de fermeture de compte]*. La distance entre l'antécédent et la conséquence est de *20 jours*. Cette règle est essentielle.

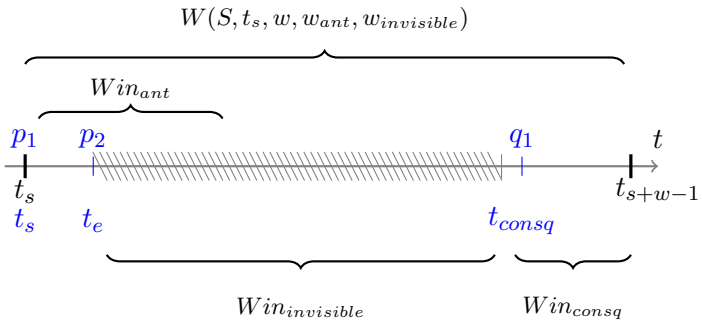


FIGURE 3.2 – Présentation des sous-fenêtres contenant un exemple d'une occurrence de la règle d'épisode $p_1, p_2 \rightarrow q_1$.

Le reste du chapitre est organisé come suit : dans la section 3.2 nous introduisons les concepts sur lesquels s'appuie l'algorithme *DEER*. Ensuite, nous détaillons les étapes de l'algorithme dans la section 3.3. Une discussion est menée dans la section 3.4 pour comparer l'efficacité de notre algorithme par rapport à un algorithme traditionnel. Nous introduisons les études expérimentales en section 3.5. Enfin, nous concluons en section 3.6.

3.2 Nouveaux concepts et redéfinitions

Nous souhaiterons préciser que plusieurs notions définies dans la section 2.5 du chapitre précédent de l'état de l'art seront utilisées dans ce chapitre. Nous allons maintenant présenter plusieurs nouveaux concepts et redéfinir quelques notions afin qu'elles s'adaptent aux caractéristiques des règles fouillées par *DEER*.
Dans la littérature, l'occurrence d'une règle est représentée par l'instant d'apparition de préfixe et l'instant d'apparition du suffixe de l'épisode sous-jacent (voir définition 2.5.4). En raison de la contrainte de distance que nous imposons sur les règles, nous proposons de représenter l'occurrence $occ(R)$ d'une règle d'épisode par trois instants : l'instant d'apparition de son préfixe t_s, l'instant d'apparition de l'unique événement dans la conséquence t_{consq} (qui peut représenter l'instant d'apparition du suffixe), et l'instant d'apparition du dernier événement de l'antécédent t_e (l'instant qui détermine la fin de l'antécédent) qui permet de vérifier si la contrainte de distance entre l'antécédent et la conséquence est acquise : Soient $P =< p_1, p_2, ..., p_k >$ et $Q =< q_1 >$, $occ(R : P \rightarrow Q) = (t_s, t_e, t_{consq}) : (p_1 \in I_{t_s}) \wedge (p_k \in I_{t_e}) \wedge (q_1 \in I_{t_{consq}})$.
Cette représentation peut être facilement modifiée afin de représenter l'occurrence d'une règle dans laquelle la conséquence est composée de plusieurs événements. Dans ce cas, quatre instants peuvent être utilisés : l'instant de début et celui de fin de l'antécédent, et l'instant de début et celui de fin de la conséquence.

Définition 3.2.1. *Soit $Win(S, t_s, w)$ une fenêtre dans la séquence S. Cette fenêtre est utilisée dans les algorithmes traditionnels pour représenter le span maximal qui borne les occurrences des règles à fouiller. Cette fenêtre a le même rôle dans notre algorithme. Cependant, nous proposons de diviser cette fenêtre en trois* **sous-fenêtres** *comme présenté dans la figure 3.2 :*

- *$Win(S, t_s, w_{ant})$, que nous appellerons Win_{ant}, est un segment de la fenêtre $Win(S, t_s, w)$ d'une taille fixe : $w_{ant} \ll w$, et qui commence à l'instant t_s. Win_{ant} représente le span maximal de l'antécédent de la règle d'épisode en cours de construction.*
- *$Win(S, t_e, w_{invisible})$, que nous appellerons $Win_{invisible}$, est un segment d'une taille fixe $w_{invisible}$ qui glisse au cours du processus de fouille. $Win_{invisible}$ débute juste après la fin de*

l'antécédent de la règle en cours de construction. $Win_{invisible}$ représente une sous-fenêtre invisible dans $Win(S, t_s, w)$ dans laquelle la conséquence d'une règle n'est pas recherchée. $Win_{invisible}$ représente un gap minimal entre l'antécédent et la conséquence de la règle.

— *$Win(S, t_e + w_{invisible} + 1, w_{consq})$, que nous appellerons Win_{consq}, est le dernier segment de la fenêtre $Win(S, t_s, w)$ de la taille : $w_{consq} \ll w$, qui commence à l'instant $t_e + w_{invisible} + 1$. Win_{consq} représente le span de la conséquence de la règle en cours de construction. Sa taille n'est pas fixe : elle évolue en fonction de la taille de l'antécédent de la règle.*

Afin de considérer ces sous-fenêtres, nous proposons de représenter une fenêtre $Win(S, t_s, w)$ comme suit : $W(S, t_s, w, w_{ant}, w_{invisible})$ où $w_{ant} + w_{invisible} + w_{consq} \geq w$. Pour simplifier les annotations, cette fenêtre est notée W.

Définition 3.2.2. *Une* **occurrence** *de la règle d'épisode R est une apparition de R dans W, où l'antécédent de R commence à l'instant t_s (le préfixe apparaît à l'instant t_s), le suffixe de l'antécédent apparaît dans la sous-fenêtre Win_{ant} (et donc le reste de l'antécédent apparaît dans Win_{ant}), et la conséquence apparaît dans la sous-fenêtre Win_{consq}. Cette occurrence satisfait la contrainte de span de l'antécédent et la contrainte de gap entre l'antécédent et la conséquence. La liste de toutes les occurrences de R est notée $Occ(R)$. Formalisons maintenant cette définition : $occ(R) = (t_s, t_e, t_{consq}) \in Occ(R) : (t_e - t_s \leq w_{ant}) \wedge (t_e + w_{invisible} \leq t_{consq} \leq t_s + w)$.*

Définition 3.2.3. *Une occurrence de la règle R (qui respecte les caractéristiques de la définition 3.2.2) est considérée comme une* **occurrence minimale**, *notée $mo(R)$, si elle ne contient pas une autre occurrence de la même règle, c'est-à-dire qu'il n'existe aucune autre occurrence dans $Occ(R)$ dont l'antécédent commence après le début de l'antécédent dans $occ(R)$ (après t_s) et finit avant lui (avant t_e), et la conséquence finit avant la conséquence dans $occ(R)$ (avant t_{consq}), comme suit : $occ(R) = mo(R) : \nexists occ'(R) \in Occ(R) = (t'_s, t'_e, t'_{consq}) : (t'_s, t'_e, t'_{consq}) \subseteq (t_s, t_e, t_{consq})$, i.e. $(t_s \leq t'_s \wedge \ t_e \geq t'_e \wedge \ t_{consq} \geq t'_{consq})$. La liste de toutes les occurrences minimales de la règle R est notée $Mo(R)$.*

Rappelons que la mesure de fréquence basée sur l'occurrence minimale permet de fouiller des règles d'épisode dans lesquelles les événements sont les plus proches possibles l'un de l'autre. Dans notre cas, cette mesure est appliquée uniquement sur l'antécédent de la règle (elle pourra aussi être appliquée sur les événements de la conséquence dans le cas où celle-ci est composée de plusieurs événements). Ainsi, elle permet de construire des règles dans lesquelles les événements de l'antécédent sont les plus proches possibles l'un de l'autre.
Dans la figure 3.3, nous présentons deux occurrences de la règle d'épisode $R : p_1, p_2 \rightarrow q_1$. Ces deux occurrences respectent la contrainte de gap entre l'antécédent et la conséquence. Une de ces deux occurrences est minimale. L'occurrence en rouge (t_s, t_e, t_{consq}) n'est pas une occurrence minimale car elle contient une autre occurrence : l'occurrence en bleu (t'_s, t'_e, t'_{consq}). Cette dernière occurrence est donc une occurrence minimale.
Dans le chapitre précédent, nous avons utilisé une séquence jouet pour expliquer les exemples fournis. Dans ce chapitre, la même séquence S (répétée dans la figure 3.4) sera utilisée pour expliquer le déroulement de l'algorithme $DEER$. Présentons un exemple d'une occurrence minimale dans la séquence S. Pour les contraintes : $w_{ant} = 2, w_{invisilbe} = 2, w = 6$ (et donc $W(S, t_s, 6, 2, 2)$), la liste des occurrences de la règle d'épisode $R : A \rightarrow E$ est $Occ(R) = \{(1, 1, 6), (2, 2, 6), (7, 7, 12)\}$. Cependant, sa liste d'occurrences minimales est $Mo(R) = \{(2, 2, 6), (7, 7, 12)\}$, car $(2, 2, 6) \subset (1, 1, 6)$.

Définition 3.2.4. *Une règle fréquente et confiante $R : P \rightarrow Q$ est considérée comme une* **règle**

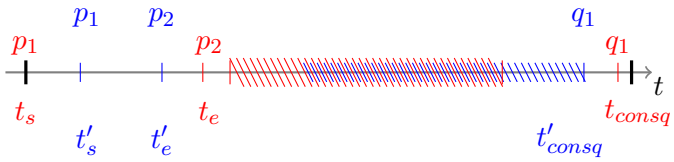


FIGURE 3.3 – Deux exemples d'occurrences de la règle $R : p_1, p_2 \rightarrow q_1$: une occurrence en couleur rouge et une autre en bleu. L'occurrence en bleu est une occurrence minimale.

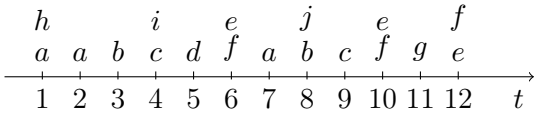


FIGURE 3.4 – Exemple d'une séquence d'événements S.

essentielle *s'il n'existe pas d'autre règle fréquente et confiante $R' : P' \rightarrow Q$ pour laquelle l'antécédent est un sous-épisode de l'antécédent de R, comme présenté dans l'équation 3.1.*

$$R : P \rightarrow Q \text{ est essentielle} : \begin{cases} R \text{ est fréquente} \\ R \text{ est confiante} \\ \nexists R' : P' \rightarrow Q : R' \text{ est fréquente} \wedge R' \text{ est confiante} \wedge P' \subset P \end{cases} \tag{3.1}$$

3.3 Les étapes de l'algorithme $DEER$

$DEER$ débute par une phase d'initialisation qui a pour but l'extraction de tous les événements fréquents dans la séquence d'événements S. Un événement fréquent représente un épisode de taille 1. Il sera noté p et sera associé avec une liste d'occurrence $Occ(p)$.
Dans le tableau 3.1, nous présentons l'ensemble des épisodes de taille 1 dans la séquence S, pour un seuil de support minimal $minsupp = 2$ (cette valeur sera utilisée dans tous les exemples présentés par la suite).

TABLE 3.1 – Les épisodes de taille 1 dans la séquence S et leur liste d'occurrences.

épisode p de taille 1	liste d'occurrences $Occ(p)$
A	(1,1), (2,2), (7,7)
B	(3,3), (8,8)
C	(4,4), (9,9)
E	(6,6), (10,10), (12,12)
F	(6,6), (10,10), (12,12)
EF	(6,6), (10,10), (12,12)

Nous allons maintenant présenter les trois étapes sur lesquelles repose l'algorithme $DEER$. La quatrième étape est optionnelle. Un pseudo-code de l'algorithme $DEER$ est présenté dans Algorithme 4.

3.3.1 Identification du préfixe

Chaque épisode de taille 1 de la phase d'intialisation est considéré comme un préfixe de l'antécédent d'une règle d'épisode à construire $R : P \rightarrow Q$. À cette étape, l'antécédent de R est donc l'épisode P (voir Algorithme 4 ligne 3).
Prenons par exemple l'événement A. Cet événement est un préfixe candidat pour une règle d'épisode, sa liste d'occurrence est $Occ(A) = \{(1,1),(2,2),(7,7)\}$ (voir tableau 3.1).

3.3.2 Identification de la conséquence

Soit P un antécédent composé d'un seul événement de l'étape précédente. Pour identifier les conséquences potentielles Q à partir de l'ensemble des épisodes candidats de taille 1, dont on dispose de la liste d'occurrences (voir Algorithme 4 ligne 5), une jointure temporelle doit être effectuée. En raison de la contrainte de gap minimal entre l'antécédent et la conséquence que nous imposons sur les règles à fouiller, la définition de la jointure temporelle présentée dans la littérature (définition 2.5.11) doit être adaptée. Nous proposons donc une opération appelée une jointure temporelle de la conséquence, définie comme suit :

Définition 3.3.1. *La jointure de la liste d'occurrences de l'épisode P avec la liste d'occurrences de la conséquence candidate Q, en satisfaisant la contrainte du gap minimal, est appelée une* **jointure temporelle de la conséquence**. *Le résultat de cette opération est la liste d'occurrences de la règle d'épisode $R : P \rightarrow Q$. Présentons maintenant la définition formelle :*
Soient les deux occurrences de deux épisodes $(t_{s_p}, t_{e_p}) \in Occ(P), (t_{s_q}, t_{e_q}) \in Occ(Q)$: $Occ(P \rightarrow Q) = Occ(P) \cdot_{consq} Occ(Q) = \{occ_i(P \cdot Q) = (t_{s_p}, t_{e_p}, t_{s_q}) : (t_{e_p} < t_{s_q}) \wedge (t_{e_p} + w_{invisible} < t_{s_q} < t_{s_p} + w)\}$.

À partir de la liste d'occurrences obtenue, la liste d'occurrences minimales (qui doit respecter la définition 2.5.10) est construite (voir Algorithme 4 ligne 7), elle sera utilisée pour calculer le support de la règle en cours de construction. À noter que toutes les occurrences (même les non minimales) doivent être conservées, comme souligné par (Méger and Rigotti, 2004), car elles seront utilisées pour compléter l'antécédent (la prochaine étape dans l'algorithme). Ceci permet de ne manquer aucune occurrence intéressante de la règle d'épisode, qui pourrait être perdue si on conserve uniquement les occurrences minimales.
L'algorithme continue et le support de la règle $R : P \rightarrow Q$ est calculé (voir Algorithme 4 ligne 9). Si la règle $R : P \rightarrow Q$ est non fréquente (son support est inférieur à $minsupp$), Q ne peut pas être sa conséquence. L'itération s'arrête donc et la règle est écartée puisqu'il n'y a pas besoin de compléter l'antécédent : quelques soient les événements utilisés pour compléter son antécédent, la règle obtenue ne pourra être fréquente. L'algorithme va donc itérer sur une autre conséquence candidate. Au contraire, si $R : P \rightarrow Q$ est fréquente, nous procédons au calcul de sa confiance (voir équation 2.2). Si la règle est confiante, elle est donc essentielle (voir définition 3.2.4 et Algorithme 4 ligne 11), l'itération est arrêtée et la règle sera donc ajoutée à l'ensemble résultat de règles d'épisode car elle remplit les caractéristiques requises. Si la règle n'est pas confiante, son antécédent doit être complété (voir Algorithme 4 ligne 16), comme présenté dans la section suivante.
Considérons la fenêtre $W(S, t_s, 6, 2, 2)$ de la séquence S. Pour la règle R ayant l'événement A comme préfixe, l'événement E est une conséquence potentielle : la jointure temporelle de la conséquence sur la règle $R : A \rightarrow E$ donne les trois occurrences suivantes : $Occ(A \rightarrow E) = \{(1,1,6),(2,2,6),(7,7,12)\}$. L'occurrence $(1,1,6)$ ne respecte pas la mesure de fréquence basée sur l'occurrence minimale et elle n'est donc pas considérée (voir définition 3.2.3), car $(2,2,6) \subset$

$(1, 1, 6)$. Par conséquent, $Mo(A \rightarrow E) = \{(2, 2, 6), (7, 7, 12)\}$. Calculons maintenant le support et la confiance de R : $supp(R) = 2$, $conf(R) = 2/3 = 0,67$. Pour $minsupp = 2$ et $minconf = 0,7$, la règle d'épisode $R : A \rightarrow E$ est fréquente mais non confiante. Elle est donc non essentielle. Par conséquent, son antécédent doit être complété, comme présenté dans la section suivante.

3.3.3 Complétion de l'antécédent

L'antécédent P de la règle en cours de construction est itérativement complété (Algorithme 4 ligne 17), c'est-à-dire qu'il est étendu avec des épisodes de taille 1, qui seront placés à sa droite, dans la limite du span maximal de l'antécédent : la sous-fenêtre Win_{ant}. À ce stade, une opération de jointure temporelle standard est effectuée entre l'épisode P et un épisode candidat P' placé directement à la droite de P (Algorithme 4 ligne 20). Nous appelons cette opération une jointure temporelle de l'antécédent, et sera définie comme suit :

Définition 3.3.2. *Soit P' un épisode de taille 1 : $P' =< p'_1 >$, et R une règle d'épisode R : $P \rightarrow Q$. Pour construire la liste d'occurrences de la règles d'épisode R : $P \cdot P' \rightarrow Q$, une opération de* **jointure temporelle de l'antécédent** *est effectuée entre R et l'épisode P', de manière à satisfaire la contrainte de span de l'antécédent (la sous-fenêtre Win_{ant}) comme suit : Soient $(t_s, t_e, t_{consq}) \in Occ(R), (t'_s, t'_e) \in Occ(P')$: $Occ(P \cdot P' \rightarrow Q) = Occ(R) \cdot_{ant} Occ(P') = \{occ_i(P.P' \rightarrow Q) = (t_s, t'_s, t_{consq}) : (t_e < t'_s < (t_s + w_{ant})\}$*

Il est important de mentionner que la définition de jointure temporelle de l'antécédent est similaire à la définition de jointure temporelle présentée dans l'état de l'art (voir définitions 2.5.11 et 2.5.12) lorsqu'elle est appliquée uniquement sur l'antécédent de la règle.
À ce stade, une fois un nouvel épisode ajouté à l'antécédent de la règle en cours de construction, le gap (représenté par la sous-fenêtre $Win_{invisible}$) glisse vers la droite de manière à ce qu'il commence juste après l'occurence de l'épisode ajouté (le dernier événement de l'antécédent). Cependant, il pourrait arriver que la contrainte de gap ne soit plus respectée entre l'occurrence de l'antécédent et la conséquence, c'est le cas lorsque $Win_{invisible}$ couvre l'occurrence de la conséquence t_{consq}. C'est pour cette raison, entre autres, qu'il est important de conserver toutes les occurrences afin de ne manquer aucune occurrence intéressante, comme illustré dans les figures 3.5 et 3.6.
Dans la figure 3.5, nous présentons un exemple d'une occurrence de la règle $R : p_1, p_2 \rightarrow q_1$. Dans le span Win_{consq} de la conséquence, deux occurrences de q_1 sont donc observées, toutes les deux sont conservées. Cependant, uniquement la première occurrence (en couleur bleu) de q_1 est considérée car elle est une occurrence minimale de la règle. Dans la figure 3.6, lorsque l'antécédent de la règle est complété par l'épisode p_3 (recherché dans la limite de Win_{ant}) qui apparaît à l'instant t'_s, la sous-fenêtre $Win_{invisible}$ glisse à droite de manière à ce qu'elle commence à l'instant $t'_s + 1$, et la taille de la sous-fenêtre Win_{consq} est mise à jour. Dans ce cas, la première occurrence de la conséquence (qui est devenue en couleur gris maintenant) n'apparaît plus dans la sous-fenêtre Win_{consq}, mais elle apparaît maintenant dans $Win_{invisilble}$. Dans ce cas, nous devons considérer la deuxième occurrence de q_1 (en couleur bleu maintenant) comme une occurrence de la conséquence de la règle $R : p_1, p_2, p_3 \rightarrow q_1$.
Une fois les occurrences de la règle d'épisode trouvées, une vérification du support et de la confiance, similaire à celle menée dans l'étape 2, est effectuée (Algorithme 4 lignes 21 et 23). L'antécédent est ensuite complété itérativement (itération sur l'étape 3) jusqu'à l'obtention d'une règle confiante.
Par conséquent, si la règle fréquente R est confiante (elle a une confiance supérieure à $minconf$), elle est ajoutée à l'ensemble résultat de l'algorithme : elle remplit les caractéristiques d'une règle

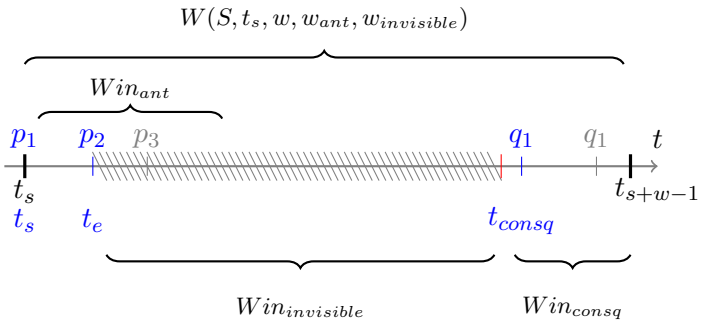


FIGURE 3.5 – Exemple d'une occurrence de la règle $R : p_1, p_2 \rightarrow q_1$.

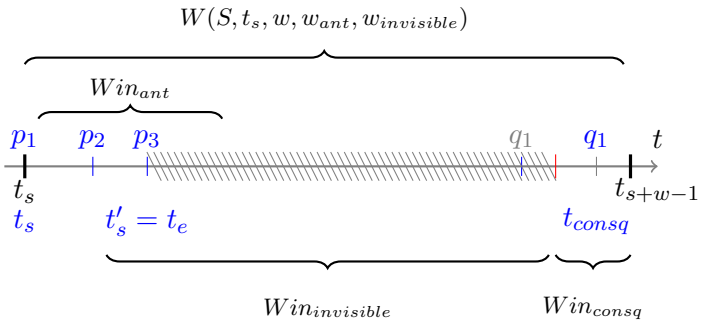


FIGURE 3.6 – Exemple de la complétion de l'antécédent de la règle $R : p_1, p_2 \rightarrow q_1$ avec l'épisode p_3 et du glissement de $Win_{invisible}$.

essentielle. Sinon, c'est-à-dire si la règle fréquente n'est pas confiante, l'antécédent P de cette règle est complété jusqu'à obtenir une règle confiante ou jusqu'à ne plus trouver de candidat pour la compléter ou jusqu'à ce que la règle devienne non fréquente.
Considérons $minsupp = 2$ et $minconf = 0.7$. Pour la règle fréquente et non confiante $R : A \rightarrow E$, sa liste d'occurrences est $Occ(R) = \{(1, 1, 6), (2, 2, 6), (7, 7, 12)\}$. B est un épisode de taille 1 candidat pour compléter l'antécédent de $R : Occ(B) = \{(3, 3), (8, 8)\}$. Une opération de jointure temporelle de l'antécédent est donc effectuée pour construire la liste d'occurrences de la règle $R : A \cdot B \rightarrow E$, comme suit : $Occ(R : A \cdot B \rightarrow E) = Occ(A \rightarrow E) \cdot_{ant} Occ(B) = \{(2, 3, 6), (7, 8, 12)\}$. Après vérification, ces occurrences sont minimales : $Occ(R) = Mo(R)$. Le support et la confiance de R sont donc calculés : $supp(R) = 2$, $conf(R) = 2/2 = 1$. La règle R est maintenant confiante, sa confiance est supérieure à $minconf$. L'étape de complétion de l'antécédent de R s'arrête pour cette règle. Cette règle est ainsi ajoutée à l'ensemble résultat de règles d'épisode fouillées par $DEER$, car elle remplit les objectifs requis.

3.3.4 La confiance temporelle

La mesure traditionnelle de confiance (équation 2.2) exploitée dans les étapes précédentes représente la probabilité que la conséquence de la règle apparaisse à la distance prédéfinie du préfixe de l'antécédent (dans la sous-fenêtre Win_{consq}), sachant que l'antécédent est apparu. Cette mesure n'exploite aucune information concernant l'occurrence de la conséquence dans la sous-fenêtre $Win_{invisible}$ (comme c'est le cas dans l'exemple présenté dans la figure 3.6). Cela rend la complexité de l'algorithme indépendante de la taille de $Win_{invisible}$. Par conséquent, même si la probabilité que la conséquence de la règle apparaisse à la distance prédéfinie est élevée (la

confiance de la règle dépasse $minconf$), la conséquence peut également apparaître plus proche de l'antécédent, ce qui peut être important pour plusieurs applications.
Dans l'exemple de la séquence de messages d'utilisateurs dans un réseau social et notamment sur la prédiction d'événements portant une polarité de sentiment négatif, il est important de ne fouiller que des règles avec une conséquence qui n'apparaît jamais dans la sous-fenêtre $Win_{invisible}$. En effet, il est inutile, même parfois dangereux pour l'application, de prédire une conséquence à une distance donnée, sachant qu'elle peut apparaître plus proche de l'antécédent.
Prenons l'exemple suivant : la banque qui gère un réseau social peut s'intéresser à découvrir que l'occurrence de la suite d'événements *[problème de carte bancaire , s'intéresser à une offre d'une banque concurrente]* résulte en l'événement que l'utilisateur généralement demande la fermeture de son compte bancaire (l'événement *[demande de fermeture de compte]*) 15 jours plus tard. Par conséquent, durant le processus de la fouille de règles, il devient non adéquat, par rapport aux objectifs fixés, de considérer la suite : considérer l'occurrence dans laquelle l'événement *[demande de fermeture de compte]* apparaît avant 15 jours, comme ayant la même importance que l'occurrence où cet événement apparaît uniquement à la distance requise (15 jours plus tard). De notre point de vue, ces deux occurrences n'ont pas la même importance, ce qui doit impacter la confiance de la règle.
Au contraire, considérons un exemple sur une séquence d'achats de clients et la prédiction de futurs achats afin de recharger les stocks du magasin au bon moment. Il reste tout de même utile de prédire les achats de clients à la distance prédéfinie, même si ces mêmes achats peuvent être réalisés entre temps.
Par conséquent, nous proposons une mesure de confiance complémentaire, que nous appelons la *confiance temporelle.*

Définition 3.3.3. *La* **confiance temporelle** *représente la probabilité que la conséquence apparaisse dans la sous-fenêtre Win_{consq}, sachant qu'elle apparaît dans la fenêtre W (le span de la règle) après l'antécédent. La définition de confiance temporelle est formalisée comme suit :*

$$conf_t(P \rightarrow Q) = \frac{supp_{consq}(P \cdot Q)}{supp(P \cdot Q)} \tag{3.2}$$

La confiance temporelle est égale à 1 si dans toutes les occurrences de la règle d'épisode $R : P \rightarrow Q$, Q apparaît uniquement dans Win_{consq}. Si $conf_t(P \rightarrow Q) \geq minconf_t$, où $minconf_t$ est un seuil prédéfini de confiance temporelle, la règle est dite confiante temporellement.

La quatrième étape de l'algorithme $DEER$, qui est optionnelle, évalue la confiance temporelle des règles afin de ne garder que celles dans lesquelles la conséquence apparaît majoritairement dans Win_{consq}. Nous avons choisi de calculer la confiance temporelle uniquement à la fin du processus de la fouille de règles, c'est-à-dire sur les règles essentielles (Algorithme 4 lignes 13 et 25), et non pas durant les trois premières étapes, pour des raisons de complexité ainsi elle sera évaluée uniquement sur les règles essentielles (résultat de la troisième étape et donc en nombre très réduit).
Prenons maintenant un exemple sur la confiance temporelle. Soit la fenêtre $W(S, t_s, 6, 2, 2)$, et la règle essentielle $R : A \rightarrow E$ résultat de l'étape 3 de l'algorithme. La confiance temporelle de cette règle dépend du nombre d'occurrences de l'événement E dans la sous-fenêtre $Win_{invisible}$, qui est égal à 1, car E apparaît à l'instant t_{10} dans $Win_{invisible}$. Par conséquent, $conf_t(R : A \rightarrow E) = 1/2 = 0,5$. Pour $minconf_t = 0,5$, R est confiante temporellement.

Algorithm 4: $DEER$: Distant and Essential Episode Rules

Données: S : la séquence d'événements, $minsupp$, $minconf$, $minconf_t$, w, w_{ant}, $w_{invisible}$, w_{consq}

Résultat: ER : l'ensemble des règles d'épisode ;
ER_t : l'ensemble des règles d'épisode temporellement confiantes

1 **Procédure** *Fouille de règles d'épisode*
2 Initialisation : extraction d'épisodes fréquents de taille 1 ;
3 **pour chaque** $p_i \in$ *épisodes fréquents de taille* 1 **faire**
4 $P \leftarrow p_i$;
5 **pour chaque** $p_j \in$ *épisodes fréquents de taille* 1 **faire**
6 $Q \leftarrow p_j$;
7 Construire $Occ(P \cdot Q)$ par jointure temporelle de la conséquence ;
8 `Conséquence` $(P \rightarrow Q)$

9 **Procédure** *Conséquence*$(P \rightarrow Q)$
10 **si** $P \rightarrow Q$ *est fréquente* **alors**
11 **si** $P \rightarrow Q$ *est confiante* **alors**
12 $ER = ER \cap \{P \rightarrow Q\}$ `/* règle essentielle */` ;
`/* Condition optionnelle */` ;
13 **si** $P \rightarrow Q$ *est confiante temporellement* **alors**
14 $ER_t = ER_t \cap \{P \rightarrow Q\}$
15 **sinon**
16 `Antécédent` $(P \rightarrow Q)$

17 **Procédure** *Antécédent*$(P \rightarrow Q)$
18 **pour chaque** $p'_i \in$ *épisodes fréquents de taille* 1 **faire**
19 $P' \leftarrow p'_i$;
20 Construire $Occ(P.P' \rightarrow Q)$ par jointure temporelle de l'antécédent ;
21 **si** $P.P' \rightarrow Q$ *est fréquente* **alors**
22 $P \leftarrow P, P'$;
23 **si** $P \rightarrow Q$ *est confiante* **alors**
24 $ER = ER \cap \{P \rightarrow Q\}$ `/* règle essentielle */` ;
`/* Condition optionnelle */` ;
25 **si** $P \rightarrow Q$ *est confiante temporellement* **alors**
26 $ER_t = ER_t \cap \{P \rightarrow Q\}$
27 **sinon**
28 `Antécédent` $(P \rightarrow Q)$

3.4 Discussion comparative : efficacité de $DEER$ en terme de temps d'exécution

Afin de montrer l'efficacité de l'algorithme $DEER$ et notamment l'impact de l'utilisation de la fenêtre $Win_{invisible}$ qui représente le gap minimal entre l'antécédent et la conséquence, nous allons comparer son efficacité à un algorithme de l'état de l'art. Dans un premier temps, nous étudions le choix de l'algorithme de l'état de l'art avec qui porte cette comparaison. Puis, nous montrons l'efficacité de $DEER$ par rapport à un algorithme de l'état de l'art au travers des exemples, pour enfin présenter une synthèse de la performance de $DEER$.

3.4.1 Choix d'algorithme de l'état de l'art

Dans cette section, nous allons étudier le choix de l'algorithme avec qui nous allons comparer $DEER$. En effet, ce choix n'est pas évident, état donné que $DEER$ procède directement à la fouille de règles (sans fouiller des épisodes), applique une mesure de fréquence basée sur l'occurrence minimale et impose une contrainte de gap entre l'antécédent et la conséquence.
Rappelons que les algorithmes de fouille de règles d'épisode sont divisées par familles d'algorithmes en fonction de mesure de fréquence utilisée (voir section 2.5.3.3 de l'état de l'art pour plus de détail) : mesure de fréquence basée sur des fenêtres, mesure de fréquence basée sur l'occurrence minimale, mesure de fréquence sans recouvrement, mesure de fréquence basée sur l'occurrence distincte, etc. L'algorithme $DEER$, utilise une mesure de fréquence basée sur l'occurrence minimale et fait donc partie de cette famille. Par conséquent, il est inévitable de comparer $DEER$ avec un algorithme de la même famille.
De plus, comme nous l'avons présenté dans la section 2.5.4 de l'état de l'art, la plupart des algorithmes imposent de contraintes sur les occurrences des règles, comme par exemple la contrainte de gap, de span, etc. La manière et la partie de règles d'épisode sur laquelle ces contraintes sont imposées diffèrent extrêmement de $DEER$ et donnent un ensemble de règles avec des caractéristiques complètement différentes, ce qui rend impossible de les filtrer pour obtenir les mêmes règles que $DEER$. Afin de pouvoir comparer ces algorithmes avec $DEER$, il faut, de notre point de vue, modifier le cœur de ces algorithmes afin d'extraire des règles qui peuvent être filtrées (dans une étape de post-traitement) pour obtenir les mêmes règles résultat dans $DEER$. Cela consomme beaucoup plus de ressources en temps par rapport à l'utilisation d'un algorithme "générique" qui n'impose pas de contraintes sur les épisodes, et pour qui uniquement une étape de post-traitement, qui filtre les règles résultats, est suffisante pour obtenir les mêmes résultats que $DEER$.
C'est la raison pour laquelle nous choisissons de comparer notre algorithme avec un algorithme "générique" qui n'impose pas de contraintes sur les épisodes. Cela permettra d'étudier l'impact de l'imposition de nos propres contraintes sur l'algorithme générique (impact sur le nombre de règles, le temps d'exécution, etc). Nous choisissons de noter un tel algorithme de la littérature $Algo_{littérature}$ qui sera utilisé dans les deux sections suivantes (sections 3.4.2 et 3.4.3).
L'algorithme $MINEPI$ (voir section 2.5.3.2) est l'unique algorithme générique pour la fouille de règles d'épisodes avec la mesure de fréquence basée sur l'occurrence minimale (il est bien le fondateur de cette mesure) et sans contraintes. Par conséquent, nous allons comparer $DEER$ avec $MINEPI$ dans la section d'expérimentations (section 3.5).

3.4.2 Comparaison sur des exemples

Dans cette section, nous allons étudier le fonctionnement de $DEER$ sur trois exemples de règles confiantes, en comparaison d'un algorithme de la littérature qui se repose sur l'extraction des épisodes pour former des règles sans contraintes $Algo_{littérature}$.
Supposons que trois règles d'épisode fréquentes et confiantes R_1, R_2 et R_3 ont un support identique de 14 mais elles diffèrent par l'emplacement de l'apparition de leur conséquence (leur antécédent apparaît dans la sous-fenêtre Win_{ant} mais leur conséquence peut être plus ou moins proche de l'antécédent). Dans le tableau 3.2, nous présentons la fréquence de chaque règle selon l'emplacement de la conséquence : dans les sous-fenêtres Win_{ant}, $Win_{invisible}$ ou dans Win_{consq}.
Pour la règle R_1, la conséquence apparaît la plupart de temps proche de l'antécédent (dans Win_{ant} ou $Win_{invisible}$). Pour la règle R_3, la conséquence apparaît la plupart du temps loin de l'antécédent (dans Win_{consq}). La conséquence de la règle R_2 apparaît, quant à elle, dans la moitié des cas dans Win_{consq}.
La conséquence de la règle R_1 apparaît très proche de l'antécédent et donc elle ne fait pas partie de règles que nous recherchons.

Table 3.2 – Un cas d'étude.

Règle d'épisode	nombre d'occurrences de la règle où la conséquence est dans :	
	$Win_{ant}/Win_{invisible}$	Win_{consq}
R_1	13	1
R_2	7	8
R_3	1	13

Considérons que W le span utilisé dans les deux algorithmes que nous souhaitons comparer ($DEER$ et $Algo_{littérature}$) a la même taille et que $minsupp = 7$.
$Algo_{littérature}$ trouve chaque occurrence des trois règles, sous la condition que la conséquence apparaît dans le span maximal fixé.
Étudions d'abord le cas de la règle R_1. $Algo_{littérature}$ extrait les 14 occurrences de l'épisode qui correspond à la règle R_1. Ensuite, dans une étape de post-traitement, il élimine 13 occurrences et conserve uniquement la seule occurrence pertinente (celle dans laquelle la conséquence apparaît dans Win_{consq}). Puis, le support de la règle est évalué. La règle ne sera donc pas considérée comme fréquente car son support ne dépasse pas $minsupp$. Ainsi, un algorithme traditionnel $Algo_{littérature}$ consomme beaucoup de temps à extraire les 13 occurrences non nécessaires qui sont filtrées à la fin. L'algorithme $DEER$, quand à lui, identifie la conséquence au début du processus de la fouille de règles, il trouve donc uniquement la seule occurrence pertinente de la règle. La règle est alors écartée immédiatement en raison de son support qui ne dépasse pas $minsupp$. La règle R_1 est donc éliminée à ce stade et la gain est double : (i) $DEER$ ne voit jamais les 13 occurrences de R_1, (ii) il s'arrête lorsque l'antécédent de R_1 n'a qu'une taille de 1 (un seul événement). Ainsi, l'algorithme $DEER$ est significativement plus rapide qu'un algorithme traditionnel $Algo_{littérature}$, pour la fouille de règles avec les mêmes caractéristiques que R_1.
Dans le cas de la règle R_2, $Algo_{littérature}$ trouve les 14 occurrences de cette règle, puis élimine les 7 occurrences dans lesquelles la conséquence n'apparaît pas dans Win_{consq}. L'algorithme $DEER$, trouve uniquement les 8 occurrences pertinentes de la règle R_2 et considère donc la règle comme fréquente. Ainsi, sur des règles similaires à la règles R_2 (en terme de caractéristiques d'occurrences), $DEER$ reste plus rapide.

Pour la règle R_3, $Algo_{littérature}$ extrait les 14 occurrences de la règle, puis élimine l'unique occurrence non pertinente. $DEER$ extrait les 13 occurrences pertinentes de la règle R_3 sans même vérifier l'unique occurrence non pertinente. $DEER$ est dans ce dernier cas similaire à un algorithme traditionnel $Algo_{littérature}$ par rapport au temps d'exécution.

3.4.3 Synthèse : $DEER$ vs. un algorithme de la littérature

Nous allons maintenant présenter la conclusion tirée de la discussion précédente (effectuée sur des exemples) concernant la performance et l'efficacité de $DEER$ par rapport un algorithme traditionnel de fouille de règles d'épisode sans contraintes $Algo_{littérature}$, ce qui est présenté dans le tableau 3.3.
À noter que la performance d'un algorithme représente ici le temps d'exécution estimé par le volume de calculs nécessaires pour extraire les règles souhaitées. C'est-à-dire que lorsque l'algorithme n'extrait pas une partie de règles ou nécessite un traitement supplémentaire pour filter les règles extraites, cela représente un volume de calculs et donc un temps d'exécution supplémentaire, ce qui rend l'algorithme moins performant.
$DEER$ est plus ou moins performant qu'un algorithme de la littérature $Algo_{littérature}$, en fonction de caractéristiques de problème de recherche, comme suit (voir tableau 3.3) :

1. Les problèmes qui se caractérisent par la fouille des associations de type règles d'épisode avec : (i) une modélisation sous contrainte de distance prédéfinie (de l'antécédent à la conséquence) et, (ii) un antécédent (le déclencheur, voir définition 1.2.1) le plus petit possible. Pour ce genre de problème, la performance de $DEER$ par rapport à $Algo_{littérature}$ est en fonction de fréquence d'apparition de la conséquence à la distance prédéfinie au seins des occurrences de la règle (à noter que cela varie d'un corpus à l'autre), comme suit :
   — Lorsque la conséquence apparaît peu de temps à la distance prédéfinie (elle est proche de l'antécédent en dehors de la distance prédéfinie), $DEER$ est *beaucoup plus performante* que $Algo_{littérature}$.
   — Lorsque la conséquence apparaît une partie du temps à la distance prédéfinie, $DEER$ est *plus performante* que $Algo_{littérature}$.
   — Lorsque la conséquence apparaît la plupart du temps à la distance prédéfinie, $DEER$ a la *même performance* que $Algo_{littérature}$.
2. Les problèmes qui se caractérisent par la fouille des associations de type règles d'épisode sans aucunes contraintes : sans tenir compte de distance des conséquences et sans avoir pour but la prédiction au plus tôt à travers un déclencheur le plus petit possible. Pour ce genre de problème, la performance de $DEER$ par rapport $Algo_{littérature}$ est comme suit :
   — $DEER$ est *moins performant* que $Algo_{littérature}$ car il n'extrait pas toutes les règles d'épisode ou toutes leurs occurrences. En effet, $DEER$ est conçu pour extraire uniquement les règles d'épisode avec des contraintes prédéfinies.

3.5 Expérimentations

Dans cette section, nous évaluons expérimentalement l'algorithme $DEER$ en comparaison d'un algorithme traditionnel. Nous étudions les caractéristiques des règles d'épisode formées, les performances dans une tâche de prédiction et le temps d'exécution.

TABLE 3.3 – Comparaison de performance entre un algorithme de la littérature $Algo_{littérature}$ pour la fouille de règles d'épisodes sans contraintes et l'algorithme $DEER$.

Caractéristiques de problème	Occurrences de règles : conséquence apparue à la distance prédéfinie ?	Performance	
		$Algo_{littérature}$	$DEER$
fouille de règles d'épisode avec : (i) distance prédéfinie (ii) antécédent le plus petit	peu du temps	+	+ + +
	une partie du temps	+	++
	la plupart du temps	+	+
fouille de règles d'épisode sans contraintes	- -	+	–

3.5.1 Caractéristiques des règles extraites

3.5.1.1 Phase d'initialisation

Le corpus, présenté dans la section 1.4, est utilisé pour évaluer l'algorithme $DEER$. Dans cette phase, les événements fréquents sont extraits (rappelons qu'un événement est composé d'un ou plusieurs items). Le seuil de support minimum $minsup$ est fixé à 30, ce qui donne 652 événements fréquents. Dans le tableau 3.4, nous montrons que ces événements sont composés d'un à trois items seulement. Les événements sont principalement (76% d'entre eux) composés d'un seul item, et ont un support élevé ($149,7$ en moyenne).

TABLE 3.4 – Taille et fréquence des événements fréquents (épisodes de taille 1)

taille (#items)	nombre (%)	support			
		min	max	moyenne	médiane
1	498 ($76,4$)	30	797	$149,7$	89
2	147 ($22,6$)	30	376	71,5	55
3	7 (1)	32	54	44,4	46

Afin d'évaluer les règles d'épisode extraites par l'algorithme $DEER$, nous allons étudier l'influence de $minsupp$, $minconf$ et $w_{invisible}$, en faisant varier un seul paramètre à la fois, et en fixant les autres paramètres.
Dans cette expérimentation, nous choisissons d'utiliser des fenêtres logiques (voir section 2.7), c'est-à-dire une fenêtre de taille 100 par exemple est composée de 100 messages.
Les événements fréquents de la phase d'initialisation seront considérés comme des épisodes de taille 1 et seront utilisés pour construire les règles d'épisode. Rappelons qu'une règle est composée de deux éléments : l'antécédent et la conséquence, chacun des ces deux éléments étant composé d'au moins un épisode de taille 1.

3.5.1.2 Impact de seuil de support et de confiance sur les règles extraites

Impact de seuil de support : Nous allons faire varier $minsupp$ afin d'étudier son influence sur le nombre de règles obtenues. Les autres paramètres restent fixes : $minconf = 0,4$, $w = 40$ (avec $w_{ant} = 20$ et $w_{invisible} = 10$). Nous avons fixé ces valeurs en fonction de nos remarques sur les caractéristiques de règles obtenues et sur la performance de l'algorithme. Le nombre de règles obtenues est présenté dans la figure 3.7. Comme attendu, plus $minsupp$ est petit, plus le nombre de règles est grand. Par exemple, lorsque $minsupp = 10$, le nombre de règles est élevé (10^4). Rappelons que le nombre d'épisodes de taille 1 est de seulement 652. Cependant, le nombre de règles diminue considérablement lorsque $minsupp$ augmente : seulement 1 200 règles pour $minsupp = 15$ et 250 règles pour $minsupp = 20$. Le nombre de règles formées lorsque $minsupp$ excède 25 est presque nul. Ce petit nombre de règles est dû à la moyenne faible de la fréquence des épisodes de taille 1 (environ 88). Cela peut également être expliqué par le fait que les règles formées représentent une dépendance distante entre l'antécédent et la conséquence : au moins 10 instants dans cette expérimentation (la valeur de $w_{invisible}$). En étudiant ces règles en détails, nous pouvons remarquer que leur confiance moyenne augmente avec l'augmentation de $minsupp$, ce qui est tout à fait logique (plus le nombre d'occurrences d'une règle est élevé, plus probable que sa confiance soit plus élevée). Dans les expérimentation suivantes, $minsupp$ sera fixé à 20.

Impact de seuil de confiance : Dans l'algorithme $DEER$, ainsi que dans tous les algorithmes traditionnels, $minconf$ est utilisé pour déterminer si la règle est acceptée (lorsqu'elle a une confiance supérieure à $minconf$). Dans $DEER$, $minconf$ est également utilisé pour savoir si l'antécédent de la règle doit être complété (sous la contrainte que la règle est toujours fréquente). Dans l'expérimentation suivante, nous allons faire varier $minconf$ (avec $minsupp = 20$, $w = 40$ ($w_{ant} = 20$, $w_{invisible} = 10$)). L'évaluation du nombre de règles est présentée dans la figure 3.8. Nous pouvons remarquer que le nombre de règles est particulièrement élevé pour $minconf = 0,2$ ($3,2 \cdot 10^4$). L'augmentation du nombre de règles entre $minconf = 0,1$ et $minconf = 0,2$ est expliqué par la façon dont les règles sont formées : lorsque la confiance d'une règle est inférieure à $minconf$, l'antécédent est complété jusqu'à avoir une règle essentielle. $DEER$ tente d'ajouter chacun des 652 événements (représenté par les épisodes de taille 1) à l'antécédent, ce qui résulte en un grand nombre de règles candidates. Certains d'entre elles sont confiantes, ce qui explique l'augmentation de nombre de règles. Cependant, lorsque $minconf$ dépasse $0,2$, le nombre de règles diminue, conformément à ce qui était attendu.
Dans le tableau 3.5, nous présentons la taille de l'antécédent des règles, en fonction de la valeur de $minconf$. La taille maximum obtenue pour les antécédent est de 3, ce qui était attendu étant donné que $DEER$ forme des règles essentielles (avec un antécédent minimal).
De plus, la taille moyenne de l'antécédent augmente avec l'augmentation de $minconf$: pour $minconf = 0,1$, la plupart de règles possède un antécédent de taille 1. Pour $minconf = 0,3$, la plupart de règles possède un antécédent de taille 2. Ce résultat était attendu : lorsque la confiance d'une règle fréquente est inférieure à $minconf$, son antécédent est complété, jusqu'au moment où la règle est confiante ou non fréquente. Par conséquent, plus $minconf$ est élevé, plus l'antécédent est grand. Une étude approfondie montre que la taille moyenne de la fenêtre qui borne l'occurrence effective de l'antécédent est égal à 8 instants (pour les antécédents de taille 2 et 3), ce qui est plus petit que le span prédéfini de l'antécédent ($w_{ant} = 20$). Cela veut dire que l'algorithme $DEER$ extrait des règles d'épisode avec un antécédent non seulement minimal en taille (en nombre d'événements) mais aussi en temps (en nombre d'instants sur lesquels l'antécédent s'étend).
Nous pouvons conclure que $DEER$ a réussi à former des règles essentielles avec une conséquence distante et une confiance relativement élevée.

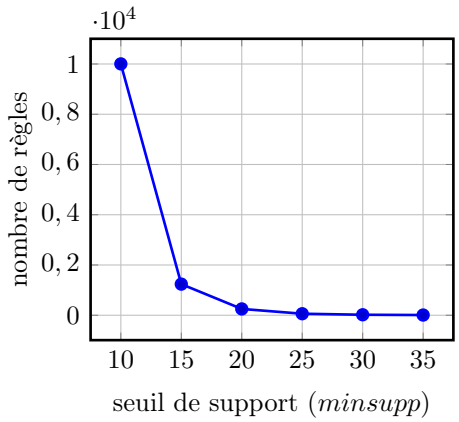


FIGURE 3.7 – Nombre de règles en fonction de seuil de support (*minsupp*).

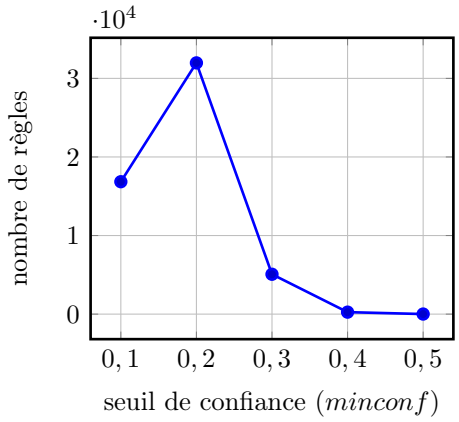


FIGURE 3.8 – Nombre de règles en fonction de seuil de confiance (*minconf*).

TABLE 3.5 – Taille de l'antécédent en faisant varier le seuil de confiance *minconf*.

minconf	#Règles	%Règles		
		antécédent de taille 1	antécédent de taille 2	antécédent de taille 3
0, 1	16 850	57, 2	42, 84	0
0, 2	31 972	4, 2	95, 6	0, 2
0, 3	5 072	0, 9	97, 5	1, 6
0, 4	251	0, 8	90, 9	8, 3
0, 5	9	0	100	0

3.5.1.3 Impact de gap inséré entre l'antécédent et la conséquence de règles

Comme nous l'avons présenté dans la section 3.2, nous proposons de diviser $Win(S, t_s, w)$ en trois sous-fenêtres, parmi lesquelles $Win_{invisible}$ de taille $w_{invisible}$, qui correspond au gap minimal entre l'antécédent et la conséquence des règles formées. Tout d'abord, nous allons étudier le nombre de règles formées en fonction de la valeur de ce gap (les autres paramètres w et w_{ant} restent fixes). Il est important de mentionner que pour pouvoir comparer $DEER$ à un algorithme traditionnel dans la suite de ces expérimentations, nous n'avons pas d'autre choix que de fixer la taille de la fenêtre W. Dans ce cas, lorsque la taille de $Win_{invisible}$ augmente, la taille de Win_{consq} diminue et donc $DEER$ identifie la conséquence de la règle dans une sous-fenêtre plus petite. Cela peut influencer la performance de l'algorithme en terme de précision et de rappel de prédiction (comme nous allons le présenter plus tard).
Dans la figure 3.9, nous présentons le nombre de règles obtenues, pour $minsupp = 20$ et $minconf = 0,4$. Deux valeurs de w sont étudiées : $w = 40$ et $w = 100$. Dans les deux cas, $w_{ant} = 20$. Il est important de noter que lorsque $w_{invisible} = 0$, les configurations sont presque similaires à celles de l'état de l'art. Nous remarquons que plus $w_{invisible}$ est grand, plus le nombre de règles est petit. Deux raisons peuvent expliquer cette diminution de nombre de règles : (i) lorsque $w_{invisible}$ augmente, w_{consq} (qui représente la taille de la sous-fenêtre dans laquelle la conséquence est recherchée) diminue en raison de la valeur fixe de w, et donc le nombre de conséquences candidates diminue ; (ii) plus $w_{invisible}$ est grand, plus la conséquence est distante, et donc plus petite est la probabilité d'avoir une dépendance entre l'antécédent et la conséquence. En effet, il est naturel que le plus grand nombre de dépendances dans un corpus se passe entre événements proches.
Cependant, même avec une valeur grande de $w_{invisible}$, plusieurs règles sont obtenues : 210 règles pour $w_{invisible} = 70$ (avec $w = 100$). Nous pouvons conclure qu'il existe effectivement une dépendance temporelle entre les messages utilisateurs dans le corpus utilisé. Lorsque le gap minimal est fixé à 50, plus de $140K$ règles sont obtenues : une forte dépendance existe entre les messages avec une telle distance.
Nous allons maintenant présenter un exemple d'une telle règle d'épisode avec $w = 100$ et $w_{invisible} = 50$: R : *(prix, positif), (information, positif)* $\rightarrow$ *(achat, positif)*. Cette règle veut dire que lorsqu'un utilisateur parle du prix d'un article et puis demande quelques informations sur cet article, il va l'acheter après certain temps ($w_{invisible} = 50$). Nous pouvons donc lui recommander, entre temps, un prêt pour acheter des articles similaires.

3.5.2 Évaluation de la confiance temporelle

Dans cette section, nous nous focalisons sur la confiance temporelle des règles obtenues. Rappelons que la confiance temporelle représente la probabilité que la conséquence apparaisse dans Win_{consq}, sachant qu'il apparaît dans W. Dans le tableau 3.6, nous présentons l'évolution de la confiance temporelle en fonction de $w_{invisible}$ pour $w = 100$ et $w_{ant} = 20$. Comme attendu, plus $w_{invisible}$ est petit, plus la confiance temporelle est élevée : la conséquence apparaît rarement dans $Win_{invisible}$ lorsque sa taille est petite. Pour $w_{invisible} = 30$, la confiance temporelle moyenne est $0,6$, ce qui reste relativement élevé. Une analyse approfondie montre que parmi les $27 \cdot 10^4$ règles obtenues pour $w_{invisible} = 30$, environ 40 règles ont une confiance temporelle égale à 1 (c'est-à-dire que la conséquence n'apparaît jamais dans $Win_{invisible}$), et que 1 400 règles ont une confiance temporelle supérieure à $0,9$ (la conséquence apparaît dans $Win_{invisible}$ dans moins de 10% des cas). Ce résultat montre que dans le corpus utilisé dans cette expérimentation, une dépendance temporelle forte est présente entre les événements.

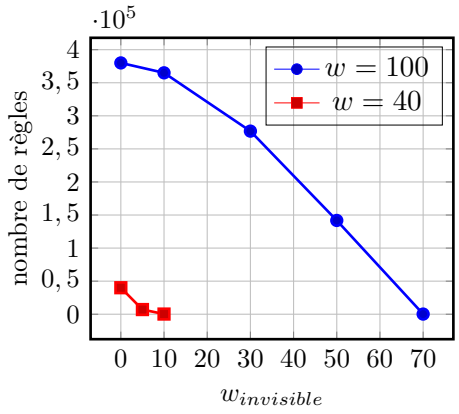


FIGURE 3.9 – Nombre de règles en fonction de taille de *gap* entre l'antécédent et la conséquence ($w_{invisible}$).

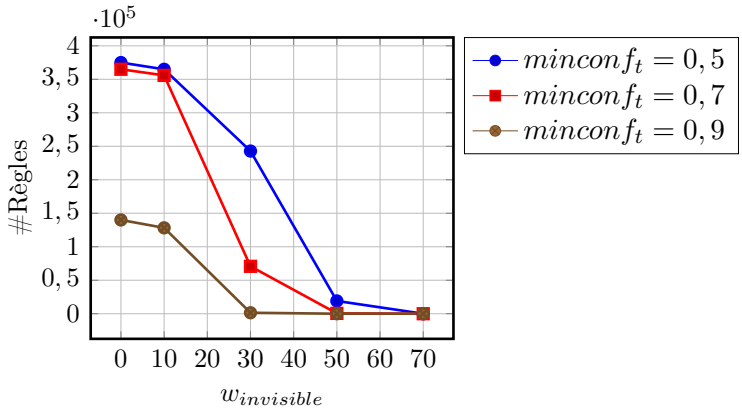


FIGURE 3.10 – Nombre de règles en fonction de taille de *gap* entre l'antécédent et la conséquence ($w_{invisible}$) et de seuil de confiance temporelle ($minconf_t$).

Comme nous l'avons mentionné, certaines applications s'intéressent à l'extraction de règles avec une conséquence qui apparaît la plupart de temps dans la sous-fenêtre Win_{consq}. Pour cette raison, un seuil minimum $minconf_t$ est fixé afin de ne maintenir que les règles ayant une confiance temporelle qui dépasse ce seuil.
Dans la figure 3.10, nous faisons varier $w_{invisible}$ des mêmes valeurs appliquées dans la figure 3.9, et étudions le nombre de règles en fonction de $minconf_t$. Nous remarquons que plus $w_{invisible}$ est grand, plus petit est le nombre de règles temporellement confiantes. Cette diminution est évidente puisque plus la conséquence recherchée est éloignée (pour une grande sous-fenêtre $Win_{invisible}$), moins la dépendance entre les événements est forte.
Malgré le fait que la confiance temporelle représente un filtre strict des règles, un nombre considérable de règles temporellement confiantes est obtenu. Prenons par exemple lorsque $w_{invisible} = 30$, environ 1 400 règles sont extraites avec $minconf_t = 0,9$, environ 70 800 règles sont extraites avec $minconf_t = 0,7$ et environ $242K$ règles pour $minconf_t = 0,5$ ce qui est particulièrement élevé. Encore une fois, cela confirme l'existence d'une forte dépendance temporelle entre les événements dans cette expérimentations. Nous pouvons conclure que $DEER$ réussit à extraire des règles essentielles en montrant le degré de dépendance temporelle, représentée par la mesure de confiance temporelle, entre l'antécédent et la conséquence.

TABLE 3.6 – Confiance temporelle ($conf_t$) en fonction de taille de *gap* entre l'antécédent et la conséquence ($w_{invisible}$).

$w_{invisible}$	min-$conf_t$	max-$conf_t$	moyenne-$conf_t$	médiane-$conf_t$
70	0	0,5	0,2	0,2
50	0,2	0,9	0,4	0,4
30	0,3	1	0,6	0,6
10	0,2	1	0,9	0,9

3.5.3 Performance en prédiction

Comme nous l'avons mentionné, les règles d'épisode obtenues sont dédiées à la tâche de prédiction d'événements. Dans cette section, nous nous focalisons sur la performance de $DEER$ en prédiction. Nous évaluons la performance au travers des mesures traditionnelles de précision et de rappel. En raison des caractéristiques temporelles du corpus utilisé, les règles d'épisode sont apprises à partir des premiers 75% de messages (les plus anciens) et sont testés sur les 25% de messages restants (les plus récents).
Dans la figure 3.11, nous présentons les valeurs de précision et de rappel sur une liste de prédiction de taille 20 (@20). La précision représente, sur une liste de 20 prédictions, le pourcentage de cas dans lesquels les événements prédits apparaissent effectivement. La rappel, quant à lui, représente le pourcentage de cas , sur une liste de 20 prédictions, dans lesquels les événements qui apparaissent effectivement, sont bien prédits par l'algorithme.
Nous prenons comme configuration : $minsupp = 20$, $minconf = 0,4$, $w = 100$ et $w_{ant} = 20$. Comme dans les expérimentations précédentes, nous faisons varier $w_{invisible}$ de 0 à 70. Avant tout, il est important de noter que, pour deux valeurs différentes de $w_{invisible}$, deux valeurs de précision et de rappel ne sont pas directement comparables car elles ne sont pas calculées sur les mêmes données, en raison de la variation de la taille de la sous-fenêtre Win_{consq} sur laquelle ces mesures sont calculées. Cela peut impacter la performance en prédiction. Les deux courbes de précision et de rappel diminuent avec l'augmentation de $w_{invisible}$. Cette diminution est prévue comme le nombre de règles diminue avec l'augmentation de $w_{invisible}$. Pour $w_{invisible} = 70$ (et $w_{consq} \geq 10$), les deux valeurs de précision et de rappel sont relativement faibles. Cela est prévu étant donné que les règles visent à prédire l'apparition des événements à une distance au moins égale à 70, dans une sous-fenêtre de petite taille $w_{consq} \geq 10$. La probabilité d'effectuer une prédiction précise des événements est donc faible. Au contraire, lorsque $w_{invisible} = 30$ (et donc $w_{consq} \geq 50$) comme dans la section précédente, nous pouvons remarquer que les deux valeurs de précision et de rappel sont acceptables. Par conséquent, lorsqu'un événement est prédit, dans 37% des cas il apparaît effectivement : dans environ 7 cas sur une liste de prédiction de taille 20, $DEER$ réussit à prédire l'apparition de l'événement. De plus, 70% des événements qui apparaissent dans la séquence sont bien prédits au travers des règles : 14 événement qui apparaissent sur 20 sont prédits par $DEER$.
Afin de pouvoir détecter la raison de la performance relativement faible de $DEER$, nous avons choisi de comparer la performance de $DEER$ à celle d'un algorithme traditionnel. À l'opposé des algorithmes traditionnels, $DEER$ applique un gap minimal de taille $w_{invisible}$ entre l'antécédent et la conséquence d'une règle d'épisode. Lorsque $w_{invisible} = 0$, les configurations de $DEER$ sont presque comparables à celles de l'état de l'art. La seule différence réside dans le fait que

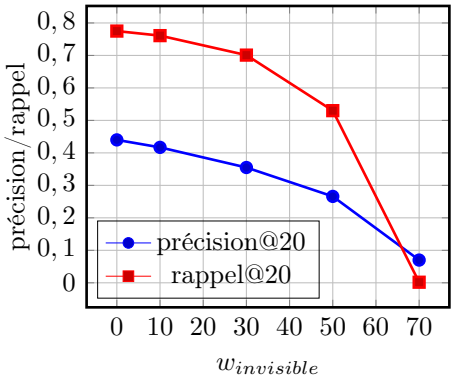


FIGURE 3.11 – Précision, rappel en fonction de taille de *gap* entre l'antécédent et la conséquence ($w_{invisible}$).

l'antécédent d'une règle dans $DEER$ ne peut pas dépasser la sous-fenêtre Win_{ant}, contrairement à un algorithme traditionnel dans lequel l'antécédent peut être construit tout au long de la fenêtre W (le span qui borne la règle). Rappelons que nous avons pris le choix de comparer $DEER$ avec l'algorithme $MINEPI$ (pour plus de détail sur ce choix voir section 3.4.1).
Pour $w = 100$, $MINEPI$ obtient 35% en précision (44% pour $DEER$) et 68% en rappel (77% pour $DEER$). Deux conclusions peuvent être tirées de ces derniers résultats : (i) le grand nombre de règles extraites par $MINEPI$ impacte sa performance en prédiction (cette prédiction qui est limitée à une liste de taille 20 (@20)), par rapport à $DEER$ qui extrait beaucoup moins de règles ; (ii) les deux algorithmes $DEER$ et $MINEPI$, dans des configurations presque comparables, ont une performance relativement faible, ce qui peut être liée à la nature du corpus utilisé.

3.5.4 Comparaison des règles formées par $DEER$ à celles de l'état de l'art

Nous nous nous intéressons maintenant à comparer les caractéristiques de règles formées par $DEER$ (lorsque $w_{invisible} = 0$) à celles de l'algorithme traditionnel $MINEPI$.
Dans le tableau 3.7, nous comparons le nombre de règles formées par $DEER$ à celles formées par $MINEPI$, pour $minsupp = 20$, $minconf = 0,4$, $w = 40$, $w_{ant} = 20$, et $w_{invisible} = 0$ (le seul paramètre utilisé dans $MINEPI$ est $w = 40$), qui correspond au w utilisé par $DEER$.
$MINEPI$ forme plus de 136 000 règles d'épisode, tandis que $DEER$ forme environ 40 000 règles, ce qui représente une diminution de 70%. Deux raisons expliquent cette diminution : (i) notre algorithme a pour objectif la construction de règles essentielles (avec un antécédent minimal en nombre d'événements), ce qui réduit le nombre de règles construites par rapport à $MINEPI$. Nous pouvons remarquer que 25% de règles extraites par $MINEPI$ possèdent un antécédent de taille au moins 3, alors que ce taux est seulement égal à $1,8\%$ pour $DEER$. La différence de ces taux $25\% - 1,8\% = 7\%$ représente le taux de règles (de taille au moins 3) de $MINEPI$ qui ne sont pas essentielles et qui seront filtrées par la suite ; (ii) la contrainte imposée par $DEER$ sur la position de la conséquence d'une règle d'épisode (dans cette configuration, la distance entre le préfixe de l'antécédent et la conséquence est au moins 20 dû à la fenêtre w_{ant}, même si $w_{invisible} = 0$), rend le nombre de règles extraites par $DEER$ réduit.
Présentons maintenant quelques exemples de règles d'épisode extraites à la fois par $DEER$ (lorsque $w_{invisible} = 0$) et par $MINEPI$.

— (crédit, positif), (conseiller, positif) → (souscription pour un prêt, positif).
— (taux d'intérêt, neutre), (crédit, négatif), (attente, neutre) → (concurrence, négatif).

TABLE 3.7 – $DEER$ vs. $MINEPI$.

Algorithme	#Règles	#R : antécédent de taille 1	#R : antécédent de taille 2	#R : antécédent de taille 3	#R : antécédent de taille 4
$DEER$	40 061	578	38 742	741	0
$MINEPI$	136 225	9 300	93 389	33 505	31

Dans les règles ci-dessus, l'antécédent et la conséquence respectent les exigences de distance et de sous-fenêtres imposées par $DEER$: l'antécédent apparaît durant les 20 premiers instants, et la conséquence apparaît à l'instant 25 pour la première règle, et à l'instant 27 pour la deuxième. Voici quelques règles qui n'ont pas été extraites par $DEER$, car elles ne satisfont pas la caractéristiques exigées : l'antécédent dépasse la sous-fenêtre Win_{ant} (pour la première règle), et la conséquence n'est pas dans la sous-fenêtre Win_{ant} (pour la deuxième règle).

— (crédit, neutre), (immobilier, neutre) → (prêt immobilier, neutre), où l'antécédent apparaît durant les 30 premiers instants et la conséquence apparaît à l'instant 31.
— (conseiller, neutre), (taux d'intérêt, positif) → (demande taux d'intérêt 0, positif), où l'antécédent apparaît durant les 5 premiers instants et la conséquence apparaît à l'instant 7.

Ces règles sont filtrées dans une étape de post-traitement afin d'avoir des règles essentielles. L'algorithme $MINEPI$ consomme donc plus de temps pour extraire des règles qui seront considérées comme étant inutiles à la fin de processus de fouille.
Nous allons maintenant montrer, via des exemples, que $DEER$ extrait uniquement les règles essentielles contrairement à $MINEPI$. Prenons donc un exemple d'une règle non essentielle extraite par $MINEPI$, et de la règle essentielle correspondante extraire par $DEER$:

— (carte bancaire, négatif), (offre concurrente, neutre), (conseiller, négatif) → (fermeture compte, négatif), est une règle confiante extraire par $MINEPI$.
— (carte bancaire, négatif), (offre concurrente, neutre) → (fermeture compte, négatif), est une règle extraite par $DEER$, elle représente la règle essentielle correspondante à la règle ci-dessus : sans l'événement (conseiller, négatif) dans l'antécédent. Cette même règle est également extraite par $MINEPI$.

Ces règles extraites par $MINEPI$ peuvent être utiles dans un cas traditionnel de prédiction : la prédiction des événements proches. Cependant, ces règles ne rentrent pas dans notre objectif pour la prédiction au plus tôt des événements distants, étant donné leur antécédent trop long en terme de temps d'apparition et nombre d'événements, de plus la conséquence est très proche de l'antécédent. Par conséquent, $DEER$ s'avère plus adéquat que les algorithmes traditionnels pour la prédiction au plus tôt des événements distants.

3.5.5 Temps d'exécution

Dans cette section, nous nous intéressons au temps d'exécution de notre algorithme en fonction du gap minimal entre l'antécédent et la conséquence ($w_{invisible}$), et par rapport au temps d'exécution de $MINEPI$. Nous considérons les deux valeurs de w précédemment étudiées de $w = 40$ et $w = 100$. Le temps d'exécution est présenté dans la figure 3.12, en terme de temps d'exécution relatif à celui de $MINEPI$ (un $temps\ relatif = 1$ représente le temps d'exécution de $MINEPI$). D'abord, nous pouvons remarquer que $DEER$ s'exécute beaucoup plus rapidement que $MINEPI$ quelles que soit les valeurs de $w_{invisible}$. Les points les plus comparables entre les

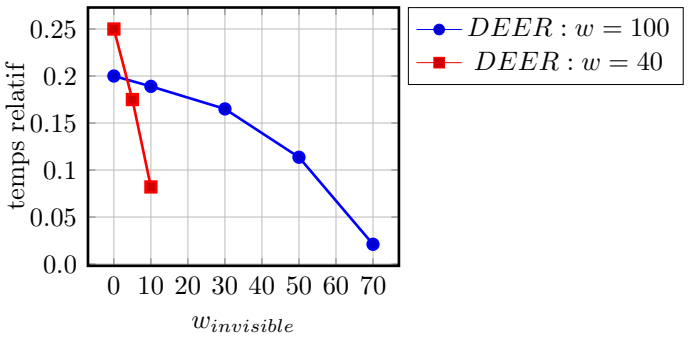


FIGURE 3.12 – Temps d'exécution relatif en fonction de taille de *gap* entre l'antécédent et la conséquence ($w_{invisible}$) ($temps\ relatif = 1$ représente le temps d'exécution de $MINEPI$).

deux algorithmes sont ceux qui correspondent à $w_{invisible} = 0$: pas de gap minimal entre l'antécédent et la conséquence, où $DEER$ tourne 5 fois plus rapidement que $MINEPI$ pour $w = 100$, et 4 fois plus rapidement pour $w = 40$. Cette diminution est due aux mêmes deux raisons présentées précédemment : (i) la conséquence est identifiée au début du processus de la fouille ce qui permet d'éliminer les règles non fréquentes dès le début du processus (voir section 3.4), (ii) $DEER$ fouille des règles essentielles, ce qui permet d'éviter des itérations inutiles une fois qu'une règle confiante est trouvée.
Nous pouvons remarquer également que plus $w_{invisible}$ est grande, plus notre algorithme est rapide. Cette tendance est évidente, comme les valeurs élevées de $w_{invisible}$ impliquent des valeurs petites de w_{consq} (w reste fixe), et donc un petit nombre de conséquences est considéré et sera étudié dans Win_{consq} pour chaque règle potentielle. Prenons par exemple la cas où $w = 100$. Le temps d'exécution dans ce cas est presque divisé par 2 entre $w_{invisible} = 10$ et $w_{invisible} = 50$.
Enfin, lorsque l'on compare les deux temps d'exécution de $DEER$ pour $w = 40$ ($w_{invisible} = 10$) et $w = 100$ ($w_{invisible} = 70$), nous remarquons qu'ils ont presque la même valeur (cela n'est pas présenté dans la figure 3.12 dans laquelle la comparaison est uniquement relative). Cela confirme que le temps d'exécution de $DEER$ est indépendant de la taille de la fenêtre $Win_{invisible}$. Il est dépendant uniquement de Win_{ant} et Win_{consq} dans lesquelles l'antécédent et la conséquence sont recherchés.

3.5.6 Discussion

Nous avons montré dans les expérimentations ci-dessus que la performance de l'algorithme $DEER$ est impactée par plusieurs facteurs : (i) en augmentant la taille de gap $w_{invisible}$ entre l'antécédent et la conséquence, l'espace de la recherche de la conséquence w_{consq} diminue, ce qui impacte la précision et le rappel de la prédiction. Nous proposons comme perspective de lancer les expérimentions de manière à ce que w_{consq} reste relativement fixe avec l'augmentation de $w_{invisible}$. Pour cela, la taille de W doit augmenter pour chaque augmentation de $w_{invisible}$; (ii) nous avons montré que les deux algorithmes $DEER$ et $MINEPI$ ont une performance en prédiction relativement faible, cela peut être dû au corpus utilisé. Nous envisageons de lancer ces expérimentations sur d'autres corpus afin de bien distinguer les facteurs qui impactent la performance de $DEER$.

3.6 Synthèse

Dans ce chapitre, nous nous sommes intéressés à la prédiction au plus tôt des événements par règles d'épisode. Nous avons montré que les algorithmes traditionnels ne sont pas adaptés à cette tâche principalement en raison de l'étape de l'extraction d'épisodes (qui précède la construction de règles) et de la construction des épisodes dans lesquels les événements sont proches, ce qui permet d'effectuer une prédiction proche et non pas une prédiction loin dans le temps.
Deux défis se présentent pour atteindre l'objectif de la prédiction au plus tôt : (i) la fouille de règles d'épisode avec une conséquence distante, (ii) la fouille de règles avec un antécédent le plus petit possible. Nous avons montré que les algorithmes traditionnels ne sont pas adaptés à relever ces défis au cours du processus de fouille, mais uniquement durant une étape de post-traitement.
Nous avons proposé l'algorithme $DEER$ qui comporte trois points originaux : (i) l'absence d'une étape d'extraction des épisodes fréquents, ce qui permet d'extraire les règles directement, (ii) l'identification de la conséquence très tôt dans le processus de la fouille, ce qui permet d'imposer une distance entre l'antécédent et la conséquence, (iii) la construction de règles dites essentielles (avec un antécédent minimal), ce qui permet d'effectuer une prédiction au plus tôt en utilisant ces règles.
Un étude expérimentale a été menée afin d'évaluer la performance de $DEER$ sur un corpus de données réelles. Après avoir montré l'effet individuel de chaque paramètre, nous nous sommes intéressés à montrer l'efficacité de $DEER$ en prédiction et en temps d'exécution, comparé à un algorithme traditionnel.
Dans l'algorithme $DEER$, nous avons proposé que la conséquence des règles soit composée d'un seul événement (comme c'est largement le cas dans la littérature). Nous avons dit que $DEER$ peut facilement être adapté pour construire des règles avec une conséquence plus longue. En effet, dans la version présentée de $DEER$ et plus précisément pendant la phase de l'identification de la conséquence, un seul épisode de taille 1 est considéré comme la conséquence de la règle en cours de construction. Pour pouvoir construire une règle avec une conséquence plus longue, nous pouvons proposer d'ajouter itérativement des épisodes de tailles 1 à la conséquence dans la limite de la sous-fenêtre Win_{consq}. L'opération de jointure temporelle, qui sert à construire la liste des occurrences de la règle, doit être adaptée dans ce cas. D'autre part, il est important de préciser que l'approche proposée n'est pas exclusive pour l'extraction de règles d'épisode à partir d'une séquence d'événements. Elle peut être généralisée à l'extraction de règles d'association à partir d'une base de transactions.
Nous avons montré également que, contrairement à la règle $R : a, b, c \rightarrow z$, par exemple, une règle essentielle, à savoir avec un antécédent plus petit, comme par exemple la règle $R' : a, b \rightarrow z$, permet de trouver des occurrences de la règle avec une distance plus grande entre l'antécédent et la conséquence, et donc de prédire la conséquence au plus tôt. Il est important de noter que la règle $R'' : b, c \rightarrow z$ dans laquelle l'antécédent est composé de deux derniers événements de l'antécédent de la règle R, n'est, en principe, pas une règle essentielle de R et ne devrait pas être extraite par l'algorithme $DEER$. Cependant, cette règle peut pratiquement être extraite par l'algorithme car elle est une règle essentielle d'une autre règle $R''' : b, c, e \rightarrow z$ et donc elle permet de prédire la même conséquence z au plus tôt. Nous considérons que, lorsque plusieurs règles essentielles prédisent au plus tôt l'apparition d'une même conséquence, cela devrait influencer, voire augmenter, la probabilité d'apparition de la conséquence. Nous envisageons d'étudier ces cas dans un travail futur.
Nous souhaitons maintenant tenir compte, dans un modèle de prédiction par règles d'épisodes, de l'une des caractéristiques de séquences complexes : la vélocité. En effet, dans une séquence véloce, appelée un flux d'événements, les règles d'épisode apparaissent au fil du temps. Générale-

ment, pour pouvoir les intégrer dans le modèle de prédiction, il faut attendre qu'elles deviennent fréquentes. Il est donc important de détecter l'émergence de l'apparition de règles afin de pouvoir les intégrer au plus tôt dans le modèle. Nous traitons ce problème dans le chapitre suivant.

Chapitre 4

EER : Un algorithme de détection de l'émergence de règles d'épisode

Sommaire

4.1 Introduction **79**
4.2 Principe de la plate-forme de traitement distribué *STORM* **81**
4.3 Principe de l'algorithme *EER* **81**
4.3.1 Phase d'initialisation 84
4.3.2 Parallélisation : Topologie proposée pour *STORM* 84
4.3.2.1 Le traitement effectué dans les unités de calcul *Spouts* 84
4.3.2.2 Le traitement effectué dans les unités de calcul *Bolts* : premier niveau 84
4.3.2.3 Le traitement effectué dans les unités de calcul *Bolts* : deuxième niveau 85
4.3.3 Phase de détection de l'émergence 86
4.3.3.1 La détection de l'émergence préliminaire 86
4.3.3.2 La détection de l'émergence finale 86
4.4 Expérimentations **88**
4.4.1 Corpus : flux d'événements 88
4.4.2 Phase d'initialisation 90
4.4.3 Caractéristiques de règles extraites 91
4.4.4 Évaluation de la performance 93
4.4.5 Impact du choix de se baser sur uniquement les règles similaires aux règles de référence 96
4.4.6 Temps d'exécution 97
4.4.7 Discussion 98
4.5 Synthèse **98**

4.1 Introduction

Dans ce chapitre, nous nous sommes confrontés au défi de la **détection au plus tôt de l'émergence de nouvelles associations de type règles d'épisodes dans un flux de d'événements**. Comme nous l'avons conclu dans la section 2.7 du chapitre "État de l'art", les approches proposées dans la littérature pour la détection de l'émergence qui se focalisent sur la surveillance

des règles, ne permettent pas d'apprendre de nouvelles règles non vues auparavant. De plus, les approches s'appuyant sur un résumé du flux peuvent manquer des occurrences de règles et souffrent donc de l'incomplétude du résultat final.
De plus, pour la prédiction au plus tôt dans un flux de données, qui est le focus de ce manuscrit, la vélocité et la variété du flux rendent cette tâche très difficile : lorsqu'une association est apparue relativement fréquemment, elle est intégrée dans le modèle prédictif. Cependant, n'intégrer une nouvelle association qu'à partir du moment où elle est fréquente, limite sa possibilité d'utilisation en prédiction. Par conséquent, il est souhaitable de pouvoir estimer qu'une nouvelle règle sera fréquente avant qu'elle le soit effectivement afin d'intégrer cette nouvelle règle dans le modèle et ainsi bénéficier de son pouvoir prédictif.
Voici deux exemples applicatifs qui montrent l'importance de cette tâche de détection :

— Dans le cadre de réseaux sociaux, les entreprises comme les banques, ont pour but la surveillance de tendances afin d'améliorer leur relation avec les clients et répondre rapidement à leurs besoins. Considérons, parmi les tendances apprises, la règle d'épisode suivante : $R : (jeune\ actif), (un\ an\ de\ salaire) \rightarrow (gros\ achat)$. R veut dire qu'après un an de salaire, un jeune actif va effectuer un gros achat comme par exemple l'achat d'une voiture ou d'un logement. Cette règle est une règle apprise auparavant et intégrée dans le modèle de prédiction, qui permet à la banque de proposer au plus tôt des prêts intéressants à son client avant que ce dernier ne se dirige vers la concurrence. Cependant, le comportement des clients change au fil du temps et de nouvelles tendances émergent dans le flux des réseaux sociaux : la règle R n'est probablement plus fiable, car une nouvelle règle émerge et commence à la remplacer, comme par exemple la règle $R' : (jeune\ actif), (premier\ salaire) \rightarrow (gros\ achat)$ (qui diffère de R par le deuxième événement de son antécédent). Il est important pour la banque de détecter au plus tôt l'émergence de la règle R', afin de pouvoir proposer des prêts à son client jeune actif dès son premier salaire.
— Dans le cadre du contrôle des épidémies, les informations sur les patients (consignées dans les fiches patients : date d'admission, âge, symptômes, traitements proposés, code postal, ...) dont disposent les hôpitaux, sont en règle générale exploitées. Un exemple de règle fouillée peut être R : ($symptôme_A$, $ville_1$ humide), ($symptôme_B$, $ville_1$ humide) $\rightarrow$ ($symptôme_A$, $ville_2$)). Cette règle représente la propagation de $symptôme_A$ vers une ville voisine $ville_2$ lorsque les événements $symptôme_A$ et $symptôme_B$ sont présents dans une même ville humide. Avec cette règle, les hôpitaux de la ville voisine $ville_2$ s'apprêtent à recevoir des patients avec le symptôme $symptôme_A$. Cependant, des mutations génétiques du virus qui cause la maladie peuvent se produire et donc des nouveaux symptômes peuvent apparaître dans la ville voisine $ville_2$, ce qui est représenté par la règle suivante R' : ($symptôme_A$, $ville_1$ humide), ($symptôme_B$, $ville_1$ humide) $\rightarrow$ ($symptôme_C$, $ville_2$). Il est très important de pouvoir détecter cette règle très tôt, car cela permet de prendre les mesures nécessaires avant que $symptôme_C$ ne se propage dans $ville_2$.

Dans ce chapitre, nous répondons à ces besoins en proposant l'algorithme EER (Emergent Episode Rules) : un algorithme pour la détection au plus tôt de l'émergence de nouvelles associations de type règles d'épisode dans un flux d'événements. Cet algorithme nous permet de détecter les premières occurrences d'une nouvelle règle d'épisode et de décider si elle tend à émerger ou non. Afin de pouvoir traiter un flux de données en temps réel, nous avons choisi de concevoir EER de manière à ce qu'il soit parallélisable. Nous avons choisi de nous baser sur la plate-forme de traitement distribué $STORM$ (voir section 2.9) pour exécuter notre algorithme.
Le reste de ce chapitre est organisé comme suit : dans la section 4.2, nous présentons le principe

de fonctionnement technique de la plate-forme *STORM* sur laquelle nous exécutons notre algorithme. Dans la section 4.3, nous présentons l'algorithme *EER*. Nous procédons à l'évaluation expérimentale de l'algorithme dans la section 4.4. Enfin, nous concluons dans la section 4.5.

4.2 Principe de la plate-forme de traitement distribué *STORM*

Comme nous l'avons présenté dans la section 2.9 du chapitre "État de l'art", la plate-forme *STORM* est inspirée de l'architecture *MapReduce*.
STORM repose sur une architecture particulière appelée *topologie*. Une *topologie* est représentée par un ensemble d'unités de calcul qui forme un graphe acyclique (voir figure 4.1), elle est constitué de deux composants principaux : les *spouts* et les *bolts* :

— Les *spouts* : unités de calcul représentant une sorte de "robinets" qui se connectent à la source du flux d'événements et qui sont donc responsables de gérer le flux et de récupérer les événements. De plus, chaque *spout* peut effectuer un premier traitement sur un ou plusieurs événements, puis envoyer le résultat vers un bolt.

— Les *bolts* : unités de calcul représentant une sorte de "verrous de robinet", ils sont chargés des traitements principaux des événements, c'est-à-dire, chaque *bolt* reçoit un événement, un paquet d'événements ou le résultat d'un premier traitement effectué par un *spout*, puis ce *bolt* effectue le traitement nécessaire.

L'utilisateur de *STORM* implémente le traitement nécessaire dans les composants. En fonction de la complexité et de la difficulté de ce traitement, l'utilisateur peut concevoir plusieurs niveaux de parallélisation. Par exemple, il pourrait établir des *spouts*, puis rajouter un premier niveau de bolts qui effectuent une partie du traitement principal, pour finaliser avec un deuxième niveau de bolts qui effectuent une deuxième partie du traitement, etc.

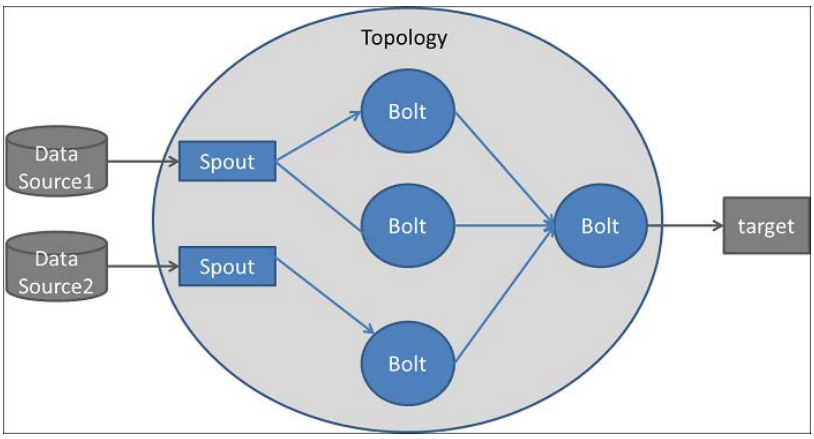


FIGURE 4.1 – La composition d'une topologie dans la plate-forme *STORM* (Smart Grid Big Data Management Team, 2014).

4.3 Principe de l'algorithme *EER*

Dans cette section nous allons présenter l'algorithme *EER* (Emergent Episode Rules) que nous proposons.
L'algorithme *EER* repose sur l'extraction de règles dans un flux de données et non pas sur la surveillance des règles fréquemment apparues auparavant.

Dans EER, nous nous focalisons sur la détection au plus tôt, c'est-à-dire sur le déclenchement de cette détection en prenant en compte uniquement un nombre très petit d'occurrences d'une règle. Cependant, se baser uniquement sur un très petit nombre d'occurrences n'est pas suffisant pour effectuer une détection précise et sûre. Par exemple, si on dispose uniquement de trois ou quatre occurrences d'une règle dans le flux d'événements, nous considérons que cela ne permet pas d'être sûr que la règle va émerger (devenir fréquente) dans le futur.
C'est pourquoi, afin de pouvoir réaliser cette tâche et augmenter la précision de cette détection, nous partons de l'hypothèse suivante :
Hypothèse : les nouvelles règles qui émergent ne sont pas totalement nouvelles, elles sont liées ou influencées par d'autres règles déjà connues (c'est-à-dire que les règles mutent au fil du temps). Nous nous intéressons aux nouvelles règles qui tendent à apparaître et sont *similaires* à des règles connues, considérant qu'elles vont émerger dans le futur.

Pour estimer la similarité entre règles, nous avons besoin d'une mesure de similarité adéquate pour les règles d'épisodes et pour notre objectif de détection de nouvelles tendances. Nous allons maintenant présenter le choix de la mesure de similarité que nous adoptons avec les éléments qui justifient ce choix.

Plusieurs types de mesures de similarité sont proposés dans la littérature. Nous nous intéressons à deux types qui peuvent être appliqués sur des épisodes (qui représentent des listes *ordonnées* d'événements) : (i) la similarité sémantique basée sur le contenu et le sens des événements (Zargayouna and Salotti, 2004) qui nécessite l'exploitation d'ontologies dédiées, et (ii) la similarité non sémantique basée sur le degré d'appariement entre deux épisodes (Veltkamp, 2001), qui prend en compte le nombre d'opérations nécessaires pour passer d'un épisode à l'autre.
Dans notre contexte, se baser sur une similarité sémantique permet de détecter les tendances qui restent dans le même cadre de ce qu'on a pu apprendre auparavant. Prenons un exemple dans le domaine bancaire. Supposons que le modèle appliqué par la banque a appris la règle suivante R : $(jeune\ actif), (gros\ achat) \rightarrow (client\ rentable)$ qui est une règle fréquente et confiante. La banque souhaite détecter l'émergence de nouvelles tendances dans le flux de ses réseaux sociaux et blogs. Lorsque l'algorithme de détection de règles émergentes est basé sur une similarité sémantique entre les nouvelles règles et les règles déjà apprises, la règle suivante peut donc être détectée comme émergente car elle est similaire à la règle R : *R' : (jeune actif), (énorme dépense)* $\rightarrow (client\ important)$. Les deux règles R et R' ont le même sens même si les derniers événements de leur antécédent et les conséquences ne sont pas identiques. Cependant, nous considérons que cette règle n'a pas beaucoup d'intérêt car elle n'apporte pas de nouveauté par rapport à la règle R : elle n'apporte pas de nouvelle information sur le comportement des clients et elle ne représente donc pas une nouvelle tendance.
Lorsque l'on se base sur une similarité non sémantique en fonction du degré d'appariement, la règle suivante peut être détectée comme émergente R'' : $(jeune\ actif), (gros\ achat) \rightarrow (client\ endetté)$ (La seule différence entre R'' et R est l'événement $(client\ endetté)$). Cette règle apporte de la nouveauté car elle permet de prédire un comportement de clients qui était inattendu auparavant. Nous choisissons de nous baser, dans l'algorithme EER, sur une mesure de similarité non sémantique car elle nous permet de détecter des nouvelles tendances.
Étant donné que la similarité sera calculée au fur et à mesure de l'apparition des règles, nous choisissons une mesure de similarité non sémantique simple et peu complexe afin de ne pas trop impacter le temps d'exécution de l'algorithme : la mesure de *distance d'édition* (Edit distance) proposée par Levenshtein (Levenshtein, 1966). La distance d'édition mesure à quel degré deux listes ordonnées d'événements, qui représentent des épisodes, sont non similaires, en calculant le

nombre minimal d'opérations nécessaires pour transformer un épisode dans un autre : le nombre d'événements qu'il faut supprimer, insérer ou remplacer pour avoir deux épisodes identiques. Cette distance est d'autant plus grande que le nombre de différences entre les deux épisodes grandit.
En utilisant la programmation dynamique (Bellman et al., 1959), la distance d'édition, notée $sim_{édit}$, entre le i^{eme} événement de l'épisode P (noté P_i) et le j^{eme} événement de l'épisode Q (noté Q_j), telle qu'elle est proposée dans (Wagner and Fischer, 1974), est calculée comme suit (équation 4.1) :

$$sim_{édit}(P_i, Q_j) = \begin{cases} max(i,j) & : \text{si } min(i,j) = 0 \\ min \begin{cases} sim_{édit}(P_{i-1}, Q_j) + 1 \\ sim_{édit}(P_i, Q_{j-1}) + 1 \\ sim_{édit}(P_{i-1}, Q_{j-1}) + 1_{P_i \neq Q_j} \end{cases} & : \text{sinon} \end{cases} \tag{4.1}$$

À noter que $1_{P_i \neq Q_j}$ est une fonction qui a une valeur égale à 0 lorsque $P_i = Q_j$ et a une valeur égale à 1 dans le case contraire. Par conséquence, la distance d'édition entre deux épisodes P et Q est calculée comme suit (équation 4.2) :

$$sim_{édit}(P, Q) = sim_{édit}(P_{|P|}, Q_{|Q|}) \tag{4.2}$$

Dans cette équation, $|P|$ et $|Q|$ représentent respectivement la taille de P et celle de Q. Lorsque les deux épisodes sont identiques, $sim_{édit}(P, Q) = 0$.

Revenons au déroulement de l'algorithme EER. Nous proposons une topologie composée d'un niveau de *spouts* et de deux niveaux de *bolts* pour la plate-forme $STORM$ sur laquelle s'exécute EER. Le principe de EER est le suivant (un schéma de l'algorithme EER est présenté dans la figure 4.3, et un pseudo-code est présenté dans Algorithme 5) :

- Dans une phase d'initialisation, un ensemble de règles, que nous appellerons "règles de référence", est disponible (obtenu par exemple avec l'algorithme du chapitre 3) : nous calculerons la similarité de règles extraites dans le flux avec les règles de référence.
- Les règles d'épisode sont extraites (par un algorithme de fouille de règles d'épisode) au fur et à mesure de l'arrivée des événements dans le flux. Cette extraction est effectuée dans le niveau de *spouts* dans $STORM$.
- Seules les règles similaires aux règles de référence sont prises en compte pour la détection de l'émergence et seront notées "règles candidates". Les règles candidates sont surveillées dans le flux : leur support est mis à jour pour chaque nouvelle apparition. Cette étape est effectuée dans le premier niveau de *bolts* dans $STORM$.
- La détection de l'émergence est réalisée en deux étapes : (i) une émergence préliminaire est détectée lorsque le support d'une règle candidate arrive à un seuil prédéfini, (ii) une mesure de taux de croissance d'apparition de règles est proposée qui prend en compte plusieurs points de contrôle (que nous appellerons "checkpoints") afin de capter la tendance de l'augmentation du support pour enfin prendre la décision de l'émergence ou non de la règle. Cette étape est effectuée dans le deuxième niveau de *bolts* dans $STORM$.

Dans les sections suivantes, nous allons détailler la conception de l'algorithme EER, les choix que nous effectuons et la manière que nous proposons pour intégrer l'algorithme dans la plateforme $STORM$.

4.3.1 Phase d'initialisation

Les règles de référence : Un ensemble de règles d'épisode appelé *règles de référence* doit être disponible avant l'exécution de l'algorithme EER. Ces règles sont fréquentes et confiantes et, elles serviront pour calculer la similarité avec les nouvelles règles extraites dans le flux d'événements. Nous formalisons la définition d'une règle de référence comme suit :

Définition 4.3.1. *Une* **règle de référence**, *notée R_{ref}, est une règle fréquente et confiante apprise auparavant (extraite auparavant par un algorithme de fouille de règles d'épisode sur un corpus dédié). L'ensemble de règles d'épisode de référence est noté ER_{ref}.*

Le modèle de fenêtres pour le traitement de flux : Comme nous l'avons montré dans la section 2.7 du chapitre de l'état de l'art, pour traiter un flux d'événements, trois modèles de fenêtres sont proposés dans la littérature : la fenêtre glissante, la fenêtre point de repère et la fenêtre amortie. Nous choisissons de nous baser sur le modèle de fenêtres glissantes (voir figure 4.3) : il est le seul modèle qui garantit de trouver toutes les occurrences d'une règle, sans manquer d'occurrences qui peuvent apparaître entre les fenêtres. En effet, comme notre objectif est la détection au plus tôt de l'émergence, chaque occurrence d'une règle est importante et il est inacceptable que l'algorithme manque une occurrence d'une règle.

4.3.2 Parallélisation : Topologie proposée pour *STORM*

Les différentes étapes de l'algorithme EER sont parallélisées à travers la topologie que nous proposons pour $STORM$. Voici les différents composants de cette topologie et les étapes de l'algorithme qui s'exécutent sur chaque composant.

4.3.2.1 Le traitement effectué dans les unités de calcul *Spouts*

Nous choisissons de concevoir les *spouts* comme suit : chaque *spout* est chargé de surveiller le flux d'événements et reçoit donc un paquet d'événements. Chaque paquet représente une fenêtre glissante W nécessaire pour l'extraction de règles d'épisodes (voir figure 4.3 et Algorithme 5 ligne 4). Chaque *spout* possédant maintenant une fenêtre d'événements W applique un algorithme d'extraction de règles d'épisode sur cette fenêtre (voir les composants bleues dans la figure 4.3 et Algorithme 5 ligne 6). Le choix de l'algorithme pour l'extraction de règles d'épisode dans chaque fenêtre se fait en fonction du besoin de l'application. Étant donné que dans ce manuscrit nous nous intéressons à la prédiction au plus tôt, nous choisissons d'appliquer un algorithme avec une mesure de fréquence basée sur l'occurrence minimale, afin de trouver les occurrences les plus proches de la réalité. Dans ce cas, une unique occurrence peut être trouvée pour une règle dans une fenêtre.
À noter qu'une fois qu'un *spout* finit cette tâche, il distribue l'ensemble de règles extraites ER à plusieurs bolts disponibles. Le *spout* se libère de cette tâche, il devient maintenant disponible et prêt à traiter une nouvelle fenêtre.

4.3.2.2 Le traitement effectué dans les unités de calcul *Bolts* : premier niveau

Dans la topologie de $STORM$, nous proposons un premier niveau de bolts qui effectuent une partie du traitement de l'algorithme (voir les composants jaunes dans la figure 4.3). Dans ce niveau, plusieurs bolts sont disponibles pour effectuer les tâches dédiées. Un bolt de premier niveau est noté $Bolt_i$.

Chaque $Bolt_i$ reçoit (de la part d'un spout) une ou plusieurs règles d'épisodes : des occurrences de règles qui apparaîssent pour la première fois dans le flux ou de règles qui sont déjà apparues dans le flux. Ces règles représentent des règles candidates $R_{candidate}$ à être émergentes dans le flux. L'ensemble de règles candidates est noté $ER_{candidates}$. Un *bolt* est chargé de fournir les informations suivantes sur chaque règle candidate :

— Les événements qui composent la règle candidate.
— Une valeur booléenne indiquant si la règle candidate est similaire ou non à, au moins, une règle de référence.
— Le support, mis à jour pour chaque nouvelle occurrence, de la règle candidate.
— L'ensemble des instants de début de chaque occurrence de la règle candidate qui est représenté par les instants de son préfixe. Cette dernière information peut être fournie par le *spout* qui a extrait la règle.

Une règle $R_{candidate}$ est donc représentée comme suit : *(R, estSimilaire, support, instantsPréfixe)*.

Exploitation de la mesure de similarité dans EER : Au cours de l'algorithme EER, la similarité entre une règle candidate et les règles de référence est calculée (voir Algorithme 5 ligne 25) en appliquant la mesure de distance d'édition (voir équation 4.2). Étant donné que la mesure de distance d'édition prend une valeur proche de 0 en cas de similarité élevée entre les règles, et une valeur proche de 1 en cas de similarité très faible, nous proposons $max_{similarité}$: un seuil maximum de la mesure de distance d'édition. Ce seuil est utilisé comme suit (voir Algorithme 5 lignes 12, 25, 26 et 27) :

— Lorsque la similarité entre une règle candidate et une règle de référence ne dépasse pas ce seuil, les deux règles sont considérées comme similaires (Algorithme 5 ligne 27).
— Si la règle candidate est similaire à au moins une règle de référence, l'information associée à cette règle concernant la similarité : $estSimilaire$ prend la valeur $vrai$, sinon elle prend la valeur $faux$.
— Cependant, si la règle candidate est identique à au moins une règle de référence, l'information associée à cette règle concernant la similarité : $estSimilaire$ prend la valeur $faux$. En effet, nous souhaitons détecter les nouvelles tendances représentées par les nouvelles règles jamais apprises auparavant, c'est-à-dire qu'elles ne sont pas identiques aux règles de référence.

Si le $Bolt_i$ dans ce niveau reçoit une règle qui apparaît pour la première fois dans le flux : $R \notin ER_{candidates}$ (voir Algorithme 5 ligne 11), il doit vérifier si la règle est similaire ou non à au moins une règle de référence et fournir toutes les informations la concernant. Si la règle est déjà apparue dans le flux : $R \in ER_{candidates}$ (voir Algorithme 5 ligne 17), le *bolt* doit juste mettre à jour les informations de la règle (à noter que le support est mis à jour par une augmentation de 1).
Chaque $Bolt_i$ envoie les informations mises à jour de $ER_{candidates}$ au *bolt* du niveau supérieur que nous allons maintenant présenter.

4.3.2.3 Le traitement effectué dans les unités de calcul $Bolts$: deuxième niveau

Nous proposons un deuxième niveau de bolts composé d'un seul *bolt* noté $Bolt'$ (voir le composant rouge dans la figure 4.3). Ce *bolt* est chargé de surveiller la mise à jour des règles candidates $R_{candidate}$ et d'effectuer les étapes nécessaires pour la détection de l'émergence de règles (voir Algorithme 5 ligne 21).
Le traitement effectué par $Bolt'$ consiste à la détection de l'émergence d'une règle et il est réalisé en deux étapes, comme nous allons le présenter dans la sous-section suivante.

4.3.3 Phase de détection de l'émergence

La détection de l'émergence se réalise dans le deuxième niveau de bolts au sein de l'algorithme EER. Nous proposons deux étapes qui nous permettent de décider de l'émergence d'une règle ou non : (i) la détection de l'émergence préliminaire en fonction du support, et (ii) la détection de l'émergence finale en fonction du taux de croissance du support à travers plusieurs points de contrôle "checkpoints". Présentons maintenant ces étapes en détail.

4.3.3.1 La détection de l'émergence préliminaire

La mise à jour de règles extraites dans le flux est surveillée (par le *bolt* du deuxième niveau). Nous nous focalisons sur les règles $R_{candidate}$ similaires aux règles de référence : $estSimilaire = vrai$. Nous proposons d'effectuer une étape préliminaire afin de sélectionner les règles candidates pour lesquelles le support dépasse un seuil prédéfini (voir Algorithme 5 ligne 22). Nous considérons ces règles comme ayant un premier signe d'émergence et nous les appelons "règles avec une émergence préliminaire", définies comme suit :

Définition 4.3.2. *Une* **règle avec une émergence préliminaire**, *notée* $R_{émergPré}$, *est une règle candidate pour laquelle le support dépasse un seuil de support prédéfini* $minsupp_{émergPré}$. *L'instant auquel la règle devient une règle avec une émergence préliminaire, diffère d'une règle à l'autre, et sera noté* $t_{émergencePré}$. *Le support de la règle à cet instant est noté* $supp_{émergPré}(R)$ *(pour le distinguer de* $supp(R)$ *utilisé dans le cas général).*

Nous considérons que, à ce stade, il n'est pas fiable ou suffisant de décider l'émergence de règles issues de cette étape, c'est-à-dire celles de l'émergence préliminaire. En effet, il est possible que la règle n'apparaisse plus dans le flux ou qu'elle apparaisse sur des périodes très espacées. Pour éviter ce problème, nous proposons une étape supplémentaire d'analyse : la détection de l'émergence finale.

4.3.3.2 La détection de l'émergence finale

Pour une règle avec une émergence préliminaire $R_{émergPré}$, nous commençons la phase de surveillance de cette règle durant une période relativement courte. En effet, cette période, qui est fixée en fonction de l'application et de la rapidité des événements dans le flux, doit être courte car comme nous souhaitons effectuer une détection au plus tôt, il ne faut pas tarder pour prendre une décision. Nous choisissons de surveiller le support de chaque règle à des points précis appelés points de contrôle ou "checkpoints" :

Définition 4.3.3. *Un* **checkpoint** *est un instant auquel l'algorithme EER effectue un contrôle de support d'une règle candidate. Le premier checkpoint pour une règle représente l'instant de la détection de son émergence préliminaire et est noté* $checkpoint_0$.

Cette étape a pour objectif de savoir à quel point l'apparition de la règle accroît au fil du temps, ce qui nous permettra de prendre la décision finale : la règle est émergente ou non. Nous avons donc besoin d'une mesure pour évaluer la croissance du support de règles $R_{émergPré}$. Pour cela, nous nous inspirons de la mesure de taux de croissance proposée dans la littérature (voir définition 2.7.2 dans le chapitre de l'état de l'art).
Le taux de croissance, tel qu'il est présenté dans l'état de l'art, doit être calculé entre deux instants, à savoir *checkpoints*. Supposons que l'algorithme EER calcule le taux de croissance entre deux instants successifs (décalés d'un seul instant). Dans ce cas, le support de la règle peut augmenter ou rester stable. S'il augmente, il ne peut être plus grand que d'une valeur de 1 par

rapport au support au premier instant, car entre deux instants successifs, pas plus qu'une seule occurrence peut être trouvée. Cependant, le support d'une règle peut rester stable pendant une longue période. Nous considérons que calculer le taux de croissance pour chaque couple de deux *checkpoints* successifs représente une perte potentielle de ressources et une charge supplémentaire sur l'unité de calcul $Bolt'$ chargée d'effectuer cette étape.
Par conséquent, nous proposons une nouvelle mesure de taux de croissance adaptée à nos besoins. Nous allons présenter cette mesure de manière "incrémentale" ce qui permettra de comprendre le choix final de cette mesure :

Taux de croissance partielle : Nous proposons de calculer le taux de croissance, que nous appelons "taux de croissance partielle", entre deux *checkpoints* qui ne sont pas successifs mais qui sont décalés l'un de l'autre d'une période de k instants. k doit être fixé en fonction de la vélocité des événements et de la nature du flux, préférablement par un expert de domaine. La mesure de taux de croissance entre deux *checkpoints* : $checkpoint_0$ et $checkpoint_1$ (qui la suit avec un décalage de k) représente le taux d'évolution du support de la règle au $checkpoint_1$ (noté $supp_{émergPré+k}(R)$) par rapport à celui au $checkpoint_0$ (noté $supp_{émergPré}(R)$). Ce taux de croissance représente le premier taux calculé pour la règle, il est donc noté $tauxCroissance_1(R)$ (équation 4.3) :

$$tauxCroissance_1(R) = \frac{supp_{émergPré+k}(R)}{supp_{émergPré}(R)} \tag{4.3}$$

Notons que ce taux a toujours une valeur supérieure à 1, du fait que le support à un instant donné est supérieur ou égal à celui à l'instant précédent. Dans la figure 4.2, nous montrons un exemple de *checkpoints* auxquels les taux de croissance doivent être calculés. Pour $k = 5$, les *checkpoints* sont $checkpoint_1 = t_{10}$, $checkpoint_2 = t_{15}$ et $checkpoint_3 = t_{20}$, etc. Le nombre de ces *checkpoint* est noté n (dans la figure 4.2, $n = 3$), il est fixé en fonction du degré d'importance, pour l'application, d'effectuer une détection au plus tôt, et donc préférablement par un expert de domaine. Par conséquent, les instants auxquels il faut calculer le taux de croissance sont : $checkpoint_1$, $checkpoint_2$, ... , $checkpoint_n$.

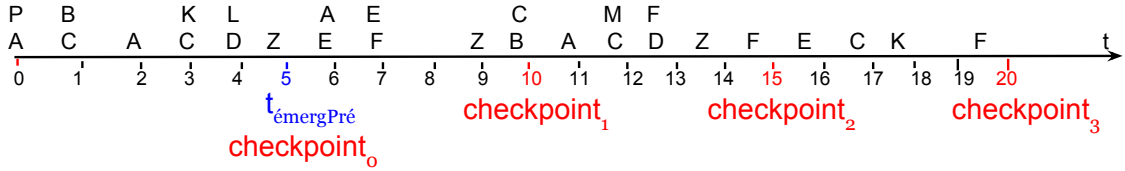


FIGURE 4.2 – Un exemple de répartition des instants pour calculer le taux de croissance, $k = 5$.

Taux de croissance finale : Nous avons besoin maintenant de décider de l'émergence de la règle au $checkpoint_n$ en prenant en compte tous les taux de croissance partielle calculés aux *checkpoints* précédents. En effet, les mesures de calcul de croissance traditionnelles ne permettent de tracer l'évolution du support qu'entre deux *checkpoints*, ou entre trois *checkpoints* en cas d'une mesure basée sur la notion de *dérivée seconde* (Ramsay, 2006). C'est-à-dire que ces mesures vérifient s'il existe une croissance entre uniquement deux ou trois instants et donc elles permettent d'avoir une vue de la croissance du support de la règle sur deux ou trois instants uniquement, ce qui est représenté par une seule valeur de taux de croissance partielle. Par conséquent, les mesures traditionnelles ne permettent pas de vérifier la croissance qui commence dès l'émergence

préliminaire de la règle ($checkpoint_1$) jusqu'à $checkpoint_n$ (où n peut être supérieur à 3), ce qui, de notre point de vue, peut être calculé par une mesure qui regroupe tous les taux de croissance partielle.
Afin de regrouper les taux de croissance partielle dans une seule mesure, nous proposons de nous baser sur la moyenne de ces taux, ce qui nous permet d'avoir une vue globale de la croissance du support de la règle dès le premier *checkpoint*. Nous proposons donc une mesure du taux de croissance finale qui représente la moyenne des taux de croissance calculés entre les n checkpoints, comme suit (équation 4.4) :

$$\begin{aligned} tauxCroissance_{fin}(R) &= moyenne(\frac{supp_{émergPré+k}(R)}{supp_{émergPré}(R)}, ..., \frac{supp_{émergPré+nk}(R)}{supp_{émergPré+(n-1)k}(R)}) \\ &= moyenne(tauxCroissance_1(R), ..., tauxCroissance_n(R)) \end{aligned} \tag{4.4}$$

Soit $min_{croissance}$ un seuil de taux de croissance finale. Lorsque $tauxCroissance_{fin}(R)$ dépasse ce seuil, nous prenons la décision que la règle R est une règle émergente (voir Algorithme 5 ligne 23). La mesure de taux de croissance finale que nous proposons est sensible et pénalise les cas lorsque le support reste stable (ou presque), ce qui est important pour la fiabilité de la détection effectuée. Nous montrons cela à travers l'exemple suivant :
Soit R et R' deux règles qui sont arrivées à ce stade de l'algorithme (avec une émergence préliminaire). Pour les deux règles, $supp_{émergPré} = 5$. Pour R, l'instant de son émergence préliminaire est $t_{émergPré} = 5$. Nous surveillons le support de R durant $n = 4$ *checkpoints* pour $k = 5$ (le décalage entre les *checkpoints*). Les supports obtenus sont : $supp_{10} = 7, supp_{15} = 9, supp_{20} = 11, supp_{25} = 13$. Le taux de croissance finale pour R est $tauxCroissance_{fin}(R) = moyenne(7/5, 9/7, 11/9, 13/11) = 1,27$. Pour $min_{croissance} = 1,2$, R est une règle émergente. Étudions maintenant le cas de la règle R'. L'instant de l'émergence préliminaire de R' est $t_{émergPré} = 7$. Les supports obtenus sont : $supp_{12} = 5, supp_{17} = 5, supp_{22} = 5, supp_{27} = 6$ (pour $k = 5$). Le taux de croissance finale ($n = 4$, $k = 5$) pour R' est $tauxCroissance_{fin}(R') = moyenne(5/5, 5/5, 5/5, 6/5) = 1,05$. Pour $min_{croissance} = 1,2$, R' n'est pas une règle émergente car son support reste presque stable au fil du temps, bien que son support dépasse le seuil d'émergence préliminaire.

4.4 Expérimentations

Dans cette section, nous présentons les études expérimentales que nous avons réalisées afin de valider l'algorithme *EER*. Pour cela, nous allons, dans un premier temps, présenter la manière dont le flux d'événements, sur lequel s'exécute l'algorithme, est généré et la phase d'initialisation de l'algorithme. Dans un deuxième temps nous réalisons une analyse de la performance de l'algorithme en fonction de différents paramètres fixés.
Il est important de mentionner que, à notre connaissance, aucun travail dans la littérature n'a été dédié à la détection de nouvelles règles émergentes dans un flux d'événements. Pour cela, l'algorithme *EER* n'est pas directement comparable avec aucun autre algorithme de l'état de l'art.

4.4.1 Corpus : flux d'événements

Dans ces expérimentations, nous utilisons le corpus utilisé pour valider les autres contributions dans ce manuscrit, présenté dans la section 1.4.

Algorithm 5: EER : Emergent Episode Rules

Données: F : le flux d'événements, ER_{ref} : l'ensemble des règles d'épisode de référence, $minsupp_{émergPré}$, $min_{croissance}$, w : taille des fenêtres, k : nombre d'instants de décalage entre les fenêtres

Résultat: $ER_{émerg}$: l'ensemble des règles d'épisode émergentes

1 **Procédure** *Détection de règles d'épisode émergentes*
2 $ER_{émerg} = \emptyset$;
3 $ER_{candidates} = \emptyset$;
4 **pour chaque** W *: une fenêtre glissante dans* F **faire**
5 Émettre W vers un **Spout** disponible

6 **Procédure** *Spout(W)*
7 $ER \leftarrow$ ExtraireRèglesÉpisode(W) `/* appliquer un algorithme d'extraction de règles */` ;
8 **pour chaque** $R \in ER$ **faire**
9 Émettre R vers un $\mathbf{Bolt_i}$ disponible `/* le nombre de Bolts` i `est initialisé par` $STORM$ `*/` ;

10 **Procédure** $Bolt_i(R, supp(R))$
11 **si** $(R \notin ER_{candidates})$ **alors**
12 estSimilaire = CalculerSimilarité(R) ;
13 **si** $(estSimilaire == vrai)$ **alors**
14 supp(R) = 1 ;
15 instantsPréfix += Bolt.demanderInstantPréfix() `/* via le` *spout* `qui a émis la règle */` ;
16 $ER_{candidates} = ER_{candidates} \cap \{R\}$

17 **sinon**
18 $supp(R) = supp(R) + 1$;
19 instantsPréfix += Bolt.demanderInstantPréfix() ;
20 Émettre $(R, estSimilaire, supp(R), instantsPréfix)$ vers $\mathbf{Bolt'}$;

21 **Procédure** $Bolt'((R, estSimilaire, supp(R), instantsPréfix))$ `/*` $\mathbf{Bolt'}$ `est l'unique` *bolt* `dans le deuxième niveau de` $STORM$ `*/`
22 **si** $(supp(R) \geq minsupp_{émergPré})$ **alors**
23 **si** $(tauxCroissance_{fin}(R) \geq min_{croissance})$ **alors**
24 $ER_{émerg} = ER_{émerg} \cap \{R\}$ `/* la règle est émergente */`

25 **Fonction** *booléen CalculerSimilarité (R)*
26 **pour chaque** $R_{ref} \in ER_{ref}$ **faire**
27 **si** $sim_{édit}(R, R_{ref}) \leq max_{similarité}$ & $sim_{édit}(R, R_{ref}) \neq 0$ **alors**
28 **retourner** $vrai$

29 **retourner** $faux$

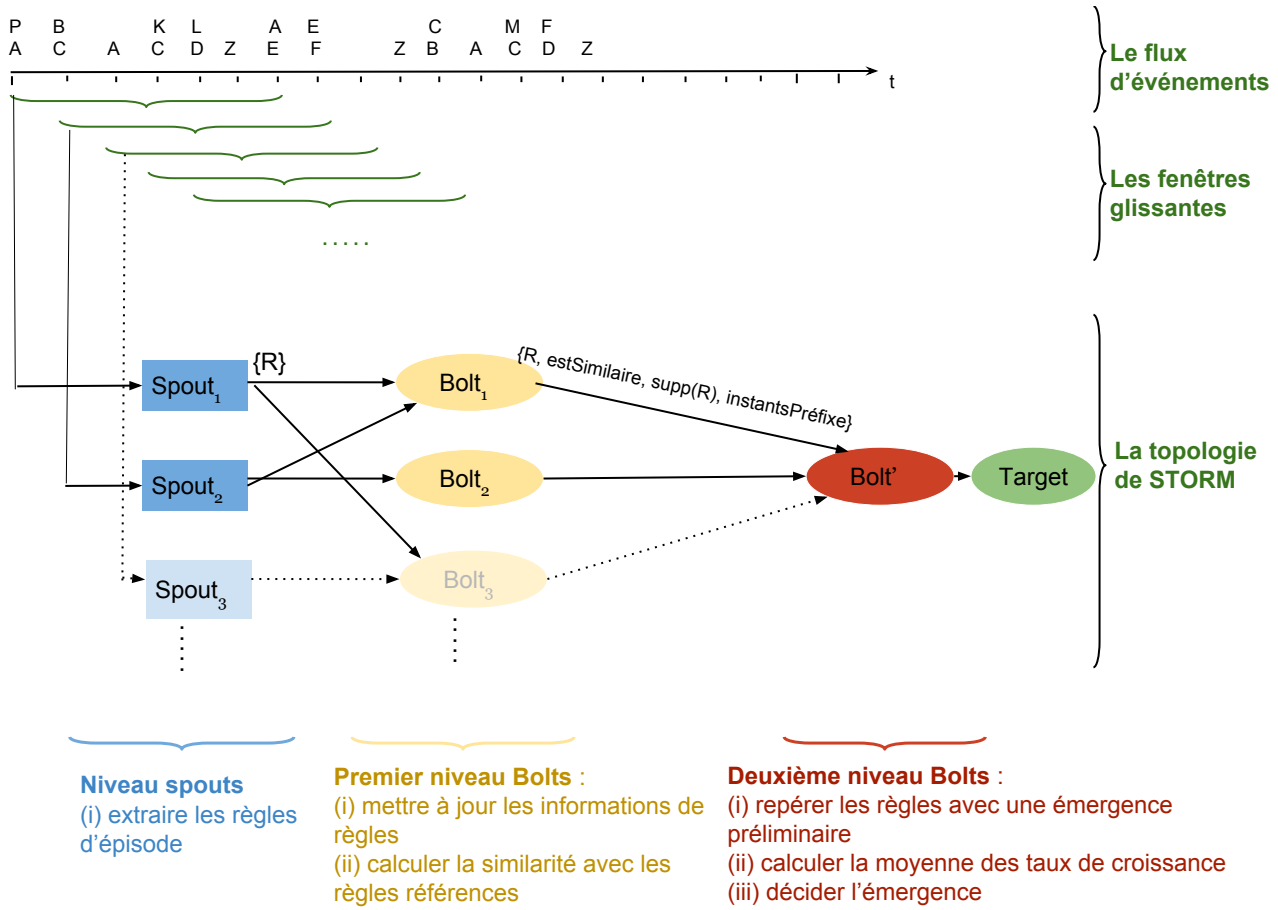


FIGURE 4.3 – Schéma de l'algorithme EER.

Rappelons que l'algorithme EER présenté dans ce chapitre exige un ensemble de règles de référence (à partir desquelles nous calculerons la similarité des nouvelles règles, voir section 4.3.1). Évidemment, cet ensemble doit être extrait à partir d'un corpus de la même nature et thématique générale que celui utilisé en tant que flux d'événements. Pour cette raison, nous choisissons de diviser le corpus, que nous venons de présenter, en deux sous-corpus : les 10000 premiers messages seront utilisés pour extraire les règles de référence, le reste, qui représente 17612 messages, sera utilisé pour générer le flux d'événements et donc pour exécuter l'algorithme EER.
Lorsque le corpus de 17612 messages est transmis à la plateforme $STORM$, sur laquelle nous avons implanté l'algorithme EER, $STORM$ s'occupe de le lire et de diffuser les événements (les messages étiquetés) en temps réel, ce qui représente un flux d'événements.
Il est important de noter que le flux utilisé pour ces expérimentations n'est pas infini. Le fait que le flux soit fini ou non n'a pas un impact sur la performance de l'algorithme.

4.4.2 Phase d'initialisation

Nous présentons maintenant le déroulement de la phase d'initialisation : comment les valeurs des différents paramètres sont fixés.

Règles de référence : Afin d'extraire les règles de référence, nous choisissons d'appliquer notre algorithme $DEER$ (voir chapitre 3) avec les valeurs suivantes des paramètres : $minsupp = 20$,

$minconf = 0,8$, $w = 100$. En effet, avec ces valeurs, le pourcentage de règles confiantes extraites est plus élevé ($0,88$ de confiance moyenne) et les règles sont plus variées (elles comportent plus de 40% des événements apparus dans le corpus), ce qui nous garantit un ensemble de règles de référence de bonne qualité. Nous choisissons d'appliquer $DEER$ sans *gap* entre l'antécédent et la conséquence : $w_{invisible} = 0$. En effet, fixer un *gap* plus grand permet d'extraire de règles adaptées pour la prédiction au plus tôt, ce qui n'est pas le focus dans le cadre de l'algorithme EER.
À partir des 10000 messages dédiés à l'extraction de règles de référence, nous obtenons 15982 règles fréquentes et confiantes, avec un support moyen de 21 (le support maximum obtenu est égal à 27), et une confiance moyenne de $0,88$. L'antécédent des règles contient en moyenne 3 événements.

Fenêtres glissantes : Comme nous l'avons mentionné dans la phase d'initialisation de EER (voir section 4.3.1), nous proposons d'utiliser des fenêtres glissantes pour traiter le flux d'événements. Nous fixons la taille de ces fenêtres à $w = 100$. En effet, nous ne voulons pas que la taille de fenêtres restreigne le nombre de règles extraites dans le flux et leur variabilité, pour cela nous avons choisi de fixer une taille grande de fenêtres, identique à la taille appliquée pour extraire les règles de référence.
Comme dans le cas d'extraction de règles de référence, nous choisissons d'appliquer le même algorithme $DEER$ (sans *gap* entre l'antécédent et la conséquence). Il est important de mentionner que avec $w = 100$, les règles extraites peuvent avoir un antécédent de taille 99, ce qui est excessivement grand et ingérable. Pour cela, nous choisissons de limiter la taille de l'antécédent de règles extraites à 3 événements. En effet, nous avons choisi cette valeur de manière à ce qu'elle soit équivalente à celle des règles de référence (3 en moyenne), ce qui permet d'augmenter la probabilité qu'une règle extraite soit similaire à une règle de référence : le nombre d'opérations nécessaires pour le calcul de similarité sera forcément plus faible, par rapport au cas où l'une des deux règles possède un antécédent très large.
À noter que, à ce stade, l'algorithme $DEER$ extrait toutes les règles sans aucune contrainte de support ou de confiance, ce qui est le principe pour la détection de l'émergence de règles dans EER : le support est cumulé (par les bolts du premier niveau) au fil du temps. Et en tout cas, la confiance ne peut pas être calculée tant que la règle n'est pas fréquente.

4.4.3 Caractéristiques de règles extraites

Nous allons maintenant étudier les caractéristiques de règles extraites en fonction de deux informations recueillies par les bolts du premier niveau (voir section 4.3.2.2) : (i) le support final des règles extraites, et (ii) la similarité de ces règles avec les règles de référence.

Analyse du support des règles extraites : En arrivant à la fin du flux (ou à un point de repère en cas d'un flux infini), nous pouvons faire le point sur le support des règles extraites. Dans la figure 4.4, nous présentons le nombre de règles extraites en fonction de leur support. Environ $496 \cdot 10^6$ règles distinctes sont extraites. La majorité de ces règles apparaissent une ou deux fois (leur support est donc très faible) : $87,5\%$ ($434 \cdot 10^6$) des règles ont un support égal à 1, et 8% ($52 \cdot 10^6$) des règles ont un support égal à 2. Ce nombre énorme de règles avec un support très faible est attendu. En effet, étant donné que l'algorithme a pour but la détection de l'émergence, il doit surveiller la croissance des supports de chaque règle dès leur première apparition. L'algorithme ne peut donc appliquer aucun filtre sur les occurrences de règles trouvées (une unique occurrence trouvée à la fois). C'est pourquoi un nombre énorme de

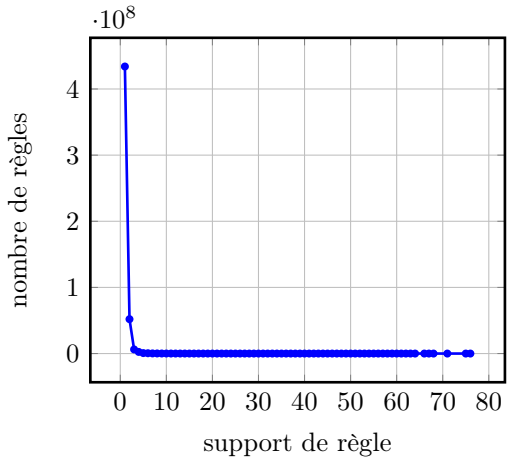


FIGURE 4.4 – Nombre de règles en fonction de leur support.

règles pour lesquelles le support n'augmente pas est obtenu.
Nous remarquons que 22398 règles ont un support supérieur à 20. Ces règles sont devenues fréquentes ($minsupp = 20$) à un moment donné dans le flux ou en arrivant à la fin du flux.
L'objectif de EER est de détecter l'émergence de ces dernières règles, à partir de n^{ème} *checkpoint* (voir définition 4.3.3 dans section 4.3.3.2) : très tôt avant qu'elles soient fréquentes dans le flux.

Analyse de similarité de règles extraites avec les règles de référence : Rappelons que la similarité de règles extraites avec les règles de référence est calculée dans le premier niveau de bolts en appliquant un seuil maximum de mesure de distance d'édition $max_{similarité}$ (qui prend une valeur entre 0 et 1).
Dans la figure 4.5, nous présentons la distribution des $434 \cdot 10^6$ règles extraites en fonction de leur similarité avec les règles de référence : le pourcentage de règles similaires, le pourcentage de règles non similaires et le pourcentage de règles identiques en fonction de la valeur du seuil $max_{similarité}$.
Notons que plus ce seuil est grand, plus grand est le nombre de règles similaires aux règles de référence. Rappelons que, étant donné que notre objectif est la détection de nouvelles tendances représentées par de nouvelles règles (qui émergent) non apprises auparavant (qui ne font pas partie de règles de référence), cela implique que les règles identiques aux règles de référence ne doivent pas être pas considérées. Le nombre de règles identiques aux règles de référence reste stable, il représente 0.002% (environ $12 \cdot 10^3$ règles) du nombre total de règles étudiées. Pour $max_{similarité} = 0,6$, environ la moitié des règles : $243 \cdot 10^6$ (49.17%) des règles sont similaires aux règles de référence, ce qui est assez élevé. Ce résultat est dû a la nature de la mesure de similarité utilisée (la mesure de distance d'édition) et étant donné que la taille de règles de référence (plus précisément de leur antécédent) est relativement petite (3 événements en moyenne), deux règles ont une forte probabilité d'être similaires lorsque l'une des deux (ou toutes les deux) a un petit nombre d'événements, ce qui diminue le nombre d'opérations nécessaires pour passer d'une règle à l'autre. Par conséquent, cette valeur de seuil de similarité ($max_{similarité} = 0,6$), représente le juste milieu pour lequel les règles obtenues ne sont pas extrêmement similaires avec les règles de référence (similarité à 30%). Cette valeur sera donc utilisée dans la suite des expérimentations.
Pour chaque règle extraite dans le flux, nous avons mené une analyse en profondeur afin d'étudier le nombre de règles de référence pour lesquelles la dite règle est similaire (figure 4.6). La figure 4.6 présente le nombre moyen de règles de référence avec lesquelles les règles extraites présentent des

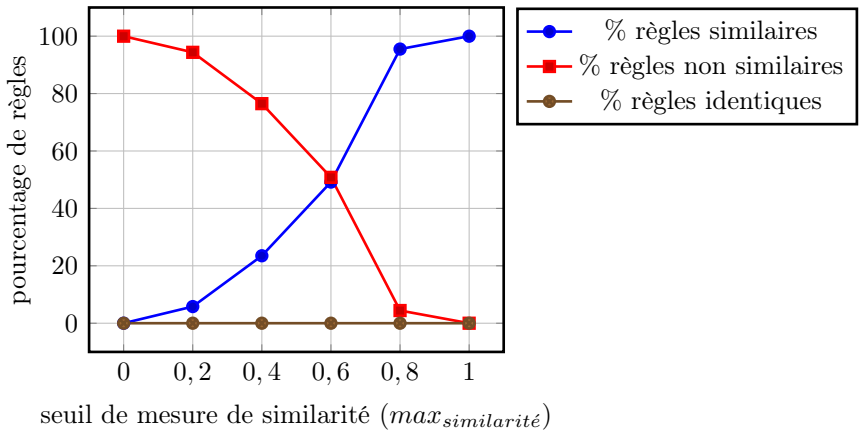


FIGURE 4.5 – Pourcentage de règles similaires, identiques et non similaires aux règles de référence en fonction de seuil de mesure de similarité ($max_{similarité}$).

similarités en fonction de la valeur de seuil de mesure de similarité $max_{similarité}$. Remarquons que le nombre moyen de règles de référence est relativement petit pour $max_{similarité} \leq 0,6$ (il est égal à 17 règles pour $max_{similarité} = 0,6$). Cependant, ce nombre augmente rapidement pour $max_{similarité} > 0,6$ (jusqu'à 650 règles pour $max_{similarité} = 0,8$). Encore une fois, cette augmentation peut être justifiée par la petite taille de l'antécédent de règles de référence qui impacte la similarité en l'augmentant.
Suite à cette analyse, nous confirmons l'utilisation de la valeur $max_{similarité} = 0,6$ pour la suite de ces expérimentations. En effet, avec cette valeur, nous obtenons le résultat le plus raisonnable : les règles obtenues ne sont pas extrêmement similaires aux règles de référence, elle sont plutôt similaires à juste un petit nombre de règles de référence (17 en moyenne), ce qui permet de détecter l'emergence de règles qui représentent des tendances suffisamment nouvelles.

4.4.4 Évaluation de la performance

Dans cette section, nous allons étudier la performance de l'algorithme EER pour la phase de détection de l'émergence réalisée par l'unique *bolt* du deuxième niveau de la topologie de $STORM$ (voir section 4.3.3).
Rappelons que dans cette phase, uniquement les règles dites candidates qui ont passé le filtre de similarité pour $max_{similarité} = 0,6$, seront prises en compte. : $243 \cdot 10^6$ règles.
À partir du filtrage antérieur commence l'étape de la détection de l'émergence préliminaire (voir section 4.3.3). D'abord, un seuil de support d'émergence préliminaire $minsupp_{émergPré}(R)$ est appliqué. Puis le taux de croissance finale est calculé en fixant les valeurs de paramètres suivantes : (i) k : le décalage entre les *checkpoints* auxquels les taux de croissance partielle sont calculés. Nous choisissons de fixer $k = 20$, une valeur peu élevée qui représente 20 instants de décalage ; (ii) n : le nombre de ces *checkpoints*, à la fin desquels le taux de croissance finale est calculé et donc l'émergence ou non de la règle est décidée en appliquant le seuil $min_{croissance}$. Nous choisissons de fixer $n = 5$, ce qui permet de calculer 5 taux de croissance partielle aux instants décalés de $k = 20$. Ces deux paramètres, n et k, doivent être fixés avant le début du flux et en fonction de la nature du corpus, plus précisément en fonction de l'espacement des occurrences de règles. En effet, la nature du corpus peut être jugée uniquement à travers le sous-corpus utilisé pour extraire les règles de référence (les premiers 10000 messages du corpus), car une fois que le flux débute, on ne peut vérifier l'espacement des occurrences de règles qu'à la fin du flux.

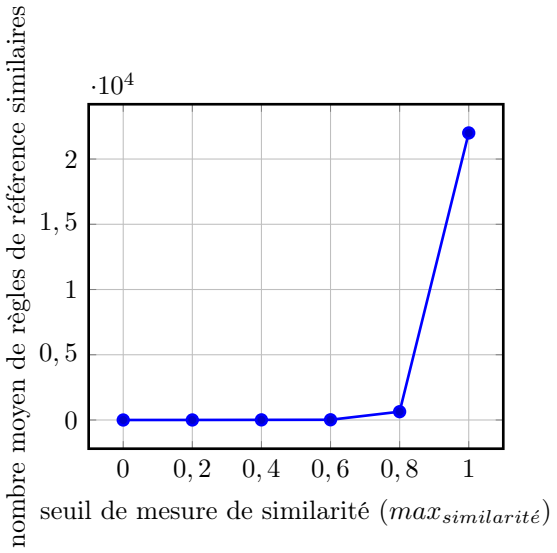


FIGURE 4.6 – Nombre moyenne de règles similaires aux règles de référence en fonction du seuil de mesure de similarité ($max_{similarité}$).

Par conséquent, nous avons constaté que, dans le sous-corpus utilisé pour extraire les règles de référence, l'espacement entre les occurrences d'une règle est de 20 instants en moyenne et que à partir de 5ème *checkpoint* en moyenne, une règle possède 10 occurrences ce que nous considérons comme étant suffisant pour décider de l'émergence d'une règle. Il est important de préciser que nous n'avons pas d'autre moyen pour fixer ces deux paramètres et qu'il est possible que la nature du corpus ne soit pas homogène : l'espacement des occurrences de règles dans une partie du corpus ne soit pas généralisable à tout le corpus.
Par conséquent, avec ces valeurs de n et k, une fois que l'émergence préliminaire est détectée, l'algorithme prend une décision d'émergence finale après $n \times k = 5 \times 20 = 100$ instants. En effet, ce nombre n'est pas du tout élevé car l'apparition de nombreuses règles est très espacée et s'étale tout au long du flux, et après 100 instants, il est possible qu'une nouvelle occurrence de la règle ne soit pas retrouvée entre temps.

Impact du seuil d'émergence préliminaire : Dans cette section, nous allons évaluer, dans un premier temps, l'impact du seuil d'émergence préliminaire $minsupp_{émergPré}(R)$ sur la performance de l'algorithme au travers des mesures traditionnelles de précision et de rappel pour la tâche de détection effectuée.
La précision et le rappel sont calculés comme suit : lorsque l'algorithme détecte l'émergence finale d'une règle à un instant donné, nous continuons à surveiller l'apparition de la règle dans le flux afin de découvrir si elle émerge effectivement, c'est-à-dire, si elle devient fréquente et si son support dépasse le seuil de support minimum $minsupp = 20$. La précision, dans ce cas, signifie : combien de règles détectées par l'algorithme comme émergentes, émergent elles effectivement ? Le rappel signifie : combien de règles qui émergent effectivement sont elles bien détectées comme émergentes par l'algorithme EER ?
Dans la figure 4.7, nous présentons les valeurs de précision et de rappel. La figure 4.7 (A) montre ces valeurs en fonction du seuil d'émergence préliminaire $minsupp_{émergPré}$, où le seuil de taux de croissance finale est fixé à $min_{croissance} = 1,2$. Nous faisons varier $minsupp_{émergPré}$ de 4 à 10. Les deux courbes de précision et de rappel diminuent avec la diminution de $minsupp_{émergPré}$.

Les valeurs de précision et de rappel ne dépassent pas 0.25. Pour $minsupp_{émergPré}(R) = 5$, c'est-à-dire lorsque les règles atteignent 5 occurrences, le taux de croissance finale est calculé et l'émergence ou non de ces règles est décidée. Dans ce cas, 20% de règles détectées comme émergentes, émergent effectivement, et 21% de règles qui émergent effectivement sont détectées par l'algorithme.
Dans un cadre simple de prédiction ou de détection de l'émergence, ce résultat est relativement mauvais : nous manquons 79% de règles qui émergent effectivement. Cependant, nous considérons ce résultat assez satisfaisant vue la difficulté de notre cadre de détection de l'émergence : nous n'effectuons pas uniquement une détection de l'émergence, mais également une détection au plus tôt de cette émergence. En outre cette détection concerne uniquement des nouvelles règles : nous ne détectons que celles qui vérifient l'hypothèse (présentée dans la section 4.3), ce qui complexifie beaucoup cette tâche. Par conséquent, lorsque l'algorithme exploite très peu d'informations (le nombre très faible d'occurrences et la similarité) afin d'effectuer cette détection, et qu'il réussit à détecter 21% de règles qui émergent effectivement, cela représente un résultat satisfaisant pour l'algorithme.
Une autre question se pose : pourquoi la performance de l'algorithme diminue-t-elle avec l'augmentation de $minsupp_{émergPré}(R)$? En effet, nous avons constaté qu'avec l'augmentation de la valeur de ce seuil, le taux de croissance finale diminue à cause de l'espacement des occurrences de règles dans le flux, c'est-à-dire le support de règles reste stable pendant la période de calcul de ce taux ($n \times k$ instants). Cela est donc dû au corpus utilisé pour générer le flux dans lequel les occurrences des règles sont relativement espacées, ce qui est au contraire de l'espacement des occurrences de règles de référence : l'espacement des occurrences de règles dans le flux atteint 500 instants dans 42% des règles. Nous pouvons conclure que le corpus utilisé n'est pas homogène en terme de répartition des occurrences de règles et que le résultat peut être totalement différent pour un autre corpus de données.

Impact du seuil de taux de croissance finale : Nous allons maintenant évaluer la précision et le rappel en fonction du seuil de taux de croissance finale $min_{croissance}$ en fixant $minsupp_{émergPré}(R) = 5$, ce qui est présenté dans la figure 4.7 (B).
Dans la figure 4.7 (B), nous faisons varier $min_{croissance}$ de 1 à $1,5$ (rappelons que $min_{croissance}$ est toujours supérieur ou égal à 1). Les deux courbes de précision et de rappel diminuent avec la diminution de $min_{croissance}$. Les raisons de cette diminution sont similaires à celles en cas de variation de $minsupp_{émergPré}$.
Remarquons que lorsque $min_{croissance} = 1$, cela représente un cas non significatif, dans lequel il est évident que la valeur de rappel est égale à 1. En effet, comme la valeur minimum de $min_{croissance}$ est 1, toutes les règles sont détectées comme émergentes par l'algorithme, il est donc impossible de se tromper dans la décision pour une règle qui émerge effectivement. C'est pour cette raison que la valeur de rappel est égale à 1 lorsque $min_{croissance} = 1$. En revanche, l'algorithme se trompe dans la précision de l'émergence effective de règles : la valeur de précision est égale à $0,37$ dans ce cas.
Pour $min_{croissance} = 1,2$, lorsque les règles (qui possèdent 5 occurrences à un moment donné) atteignent un taux de croissance finale de $1,2$, 20% de règles détectées comme émergentes, émergent effectivement, et 21% de règles qui émergent effectivement sont détectées par l'algorithme. Comme nous l'avons mentionné, nous considérons que ces résultats sont plutôt satisfaisants dans le cadre de notre objectif.
Remarquons que dans la figure 4.7 (A) et (B), la valeur de rappel est toujours supérieure à la valeur de précision, ce que nous considérons comme un point fort pour la performance de l'algorithme. En effet, dans le cadre de la détection de l'émergence, nous considérons qu'il est

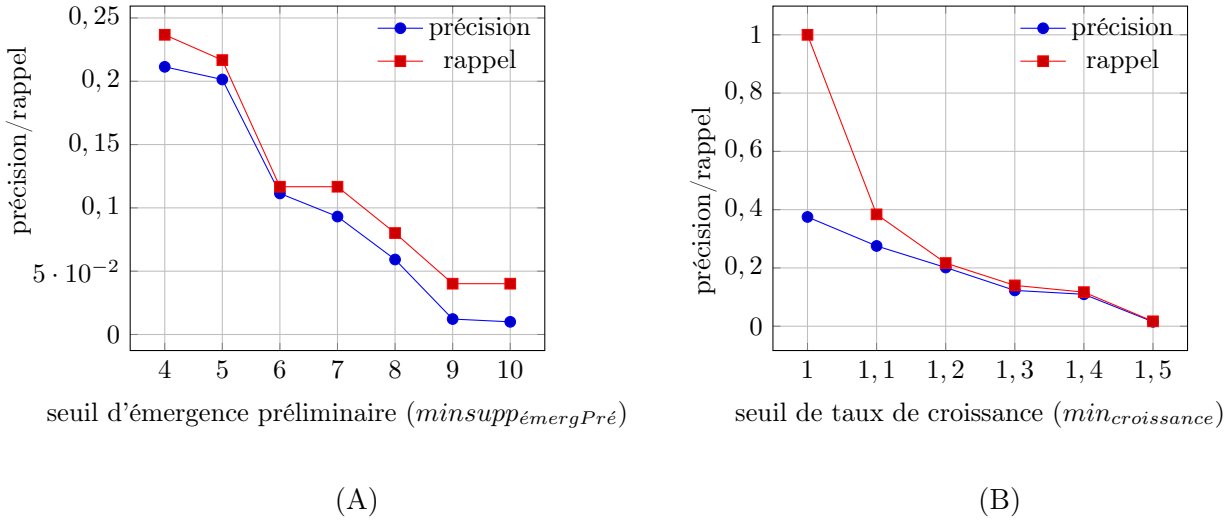


FIGURE 4.7 – Précision et rappel : (A) en fonction de seuil d'émergence préliminaire ($minsupp_{émergPré}$), (B) en fonction de seuil de taux de croissance ($min_{croissance}$).

préférable d'être plus performant dans la détection de règles qui emergent effectivement (le rappel), que d'être plus performant dans le sens des règles que l'algorithme détecte comme émergentes qui le sont effectivement (la précision).
Prenons maintenant quelques exemples de règles détectées comme émergentes dans le flux.

— La règle R : *(souscription), (bonne relation conseiller)* $\rightarrow$ *(acheter un produit)* est détectée comme émergente : une fois que l'algorithme a détecté que $supp(R) = 5$, le taux de croissance finale a été calculé après $n \times k = 100$ instants, ce qui résulte $tauxCroissance_{fin}(R) = 1,5$. Le support maximum obtenu à un moment donné pour R est égal à 30.

— La règle R : *(problème épargne), (prêt à taux zéro)*$\rightarrow$ *(contacter un concurrent)* est détectée comme émergente avec $tauxCroissance_{fin}(R) = 1,3$. Le support maximum obtenu pour R est égal à 23.

4.4.5 Impact du choix de se baser sur uniquement les règles similaires aux règles de référence

Rappelons que l'originalité principale dans EER réside dans l'hypothèse que les nouvelles règles sont forcément liées aux règles de référence et donc similaires à ces dernières. La question qui se pose est : tenant compte uniquement de règles similaires, est-ce que l'algorithme est plus performant dans la détection de l'émergence de nouvelles règles ?
Pour répondre à cette question, nous nous intéressons maintenant à reproduire l'évaluation de la performance (représentée par la précision et le rappel) en fonction de la valeur du seuil de taux de croissance finale $min_{croissance}$, mais cette fois nous conservons également les règles non similaires, c'est-à-dire que nous ne prenons pas en compte la similarité entre les nouvelles règles et les règles de référence. À préciser que les règles identiques aux règles de référence doivent être éliminées, car notre objectif reste de détecter l'émergence de nouvelles règles uniquement.
Dans la figure 4.8, nous présentons la précision et le rappel en fonction de $min_{croissance}$ dans deux cas : (i) avec la prise en compte de la similarité entre les nouvelles règles et les règles de référence (courbes bleue et rouge), ce qui était également présenté dans la figure 4.7 (B)), (ii) sans la prise

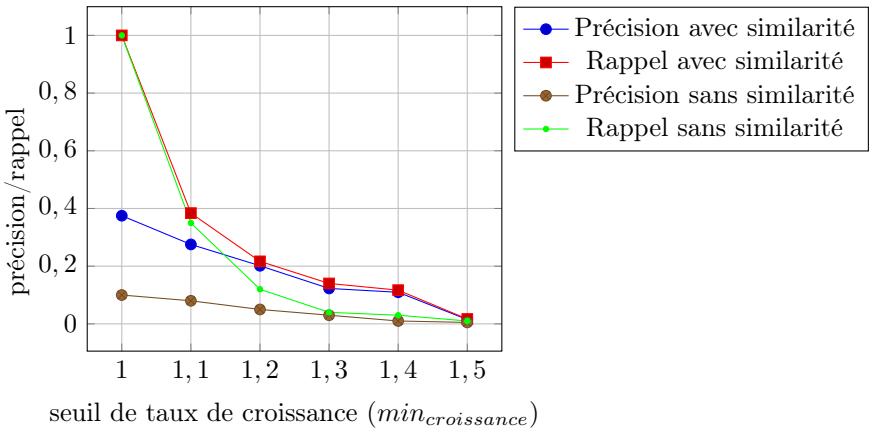


FIGURE 4.8 – Précision et rappel en fonction de seuil de taux de croissance ($min_{croissance}$) : (i) avec la prise en compte de similarité avec les règles de référence, (ii) sans la prise en compte de similarité.

en compte de la similarité (courbes marron et vert). Les formes de courbes de précision dans les deux cas sont similaires, également pour les courbes de rappel. Remarquons que la précision et le rappel sont meilleurs en cas de prise en compte de la similarité. Pour $min_{croissance} = 1,20$, la précision est 4 fois plus élevée en cas de prise en compte de similarité (précision = 0.2) qu'en cas de non prise en compte de cette mesure (précision = 0.05). Le rappel, quant à lui, est $1,75$ fois plus élevé en cas de prise en compte de similarité (précision = 0.21) qu'en cas de non prise en compte de cette mesure (précision = 0.12).
Cela prouve que notre hypothèse sur la similarité entre les nouvelles règles et les règles de référence est valable. Nous pouvons conclure également qu'il existe effectivement un lien entre les nouvelles règles et les règles de référence dans le corpus étudié.

4.4.6 Temps d'exécution

Afin d'évaluer le temps d'exécution de l'algorithme EER, il est important de mentionner que la plate-forme $STORM$, sur laquelle l'algorithme EER s'exécute, est conçue pour le traitement en temps réels de flux et se charge de la distribution des tâches vers ses composants (spouts et bolts) de façon optimale. C'est la raison pour laquelle le temps d'exécution de l'algorithme EER est très faible : 850 secondes pour traiter le corpus qui correspond au flux composé de 17612 instants. Nous avons aussi remarqué que la variation de différents seuils et paramètres nécessaires pour l'exécution de l'algorithme n'impacte pas significativement le temps d'exécution de l'algorithme. Encore une fois, cela est dû au fait que les composants de $STORM$ sont conçus de manière à prendre en compte, dans la mesure du possible, la charge de calcul attribuée à chaque composant.
Cependant, une analyse approfondie a montré que le calcul de similarité effectué dans l'algorithme représente 60% de son temps d'exécution. Il est connu dans la littérature que le calcul de similarité consomme beaucoup de ressources en temps. Malgré l'impact de cette mesure sur le temps de l'exécution, son grand intérêt pour la performance de l'algorithme en précision et en rappel reste le plus important.

4.4.7 Discussion

Suite à ces expérimentations, nous pouvons conclure que EER réussit à détecter au plus tôt l'émergence de nouvelles associations. Le résultat obtenu est relativement satisfaisant et dépend du corpus utilisé. Il est important de rappeler que l'algorithme détecte l'émergence de "nouvelles" règles, ce qui fait partie des grands défis dans le domaine du traitement de flux d'événements. La performance de l'algorithme est impactée également par les occurrences relativement espacées de chaque règle dans le deuxième sous-corpus utilisé pour généré le flux. En effet, lorsque les occurrences d'une règle sont espacées, l'algorithme considère que le support de la règle est stable et donc décide la non émergence de la règle. Pour améliorer ces résultats, nous pourrons proposer l'augmentation de décalage entre les instants k pour lesquels les taux de croissance partielle sont calculés. Cependant, cela retarderait la prise de décision pour l'émergence, ce qui est contradictoire avec le principe que nous défendons dans notre algorithme qui est d'effectuer une détection "au plus tôt". Une autre piste est envisageable, il s'agit de proposer par exemple une mesure de taux de croissance adaptée à l'espacement des occurrences de chaque règle.
D'autre part, l'ensemble de règles de référence disponible au démarrage de l'algorithme et la nature du corpus utilisé impactent également la performance de l'algorithme : un meilleur résultat pourrait être obtenu pour un corpus dans lequel l'apparition d'événements ne serait pas très espacée et l'alphabet d'événements serait fixe et plus petit.

4.5 Synthèse

Dans ce chapitre, nous nous sommes intéressés à la détection au plus tôt de l'émergence de nouvelles règles dans un flux d'événements. Un défi se présente pour atteindre cet objectif : il n'est pas suffisant de se baser sur un nombre d'occurrences très faible pour pouvoir détecter l'émergence d'une règle.
Nous avons proposé l'algorithme EER qui repose sur l'hypothèse que les nouvelles règles ne sont pas complètement indépendantes, elles sont forcément liées ou influencées par d'autres règles déjà connues. Pour cela, nous considérons que les nouvelles règles qui tendent à apparaître et qui sont *similaires* à des règles connues, vont émerger dans le futur. Ainsi, EER détecte l'émergence uniquement des règles similaires aux règles déjà extraites (que nous avons appelées : règles de référence). Nous avons choisi d'utiliser la distance d'édition comme mesure de similarité. L'algorithme EER s'exécute sur la plateforme de traitement distribué $STORM$. Nous avons proposé une *topologie* pour $STORM$ composé de *spouts* et de deux niveaux de *bolts*. Nous avons proposé une phase de détection d'émergence basée dans un premier temps sur la détection de l'émergence préliminaire et dans un deuxième temps sur la détection de l'émergence finale, à travers une nouvelle mesure de taux de croissance que nous l'avons appelée "mesure de taux de croissance finale".
Une étude expérimentale a été menée afin d'évaluer la performance de l'algorithme EER sur un flux d'événements composé de données réelles. Cette étude a montré l'effet individuel de chaque paramètre, la performance de l'algorithme et le temps d'exécution de l'algorithme. Les expérimentations ont prouvé l'importance de la prise en compte de la similarité, ce qui augmente la performance de l'algorithme. L'exécution de cet algorithme sur un autre corpus de données représente l'une des perspectives principales de ce travail.
Discutons maintenant la fiabilité dans une tâche de prédiction de règles extraites (prédire la conséquence de la règle). En effet, les règles détectées comme émergentes sont associées uniquement à une valeur de support (qui a tendance à augmenter). Cependant, pour qu'une règle soit fiable en prédiction, il faut qu'elle soit associée à une valeur de confiance (voir section défini-

tion 3.3.3 dans le chapitre de l'état de l'art), ce qui est impossible car la règle n'est pas encore fréquente à ce stade. Nous souhaitons traiter ce problème dans un travail futur.
D'autre part et toujours dans le cadre de prédiction par règles d'épisodes, une fois qu'un événement est prédit, l'application doit réagir dans certains cas face à cette prédiction afin d'impacter l'événement prédit. Nous nous intéressons dans ce manuscrit également à la manière dont on peut impacter les événements futurs, ce que nous allons présenter dans le chapitre suivant.

Chapitre 5

IE : un algorithme de détection d'événements influenceurs

Sommaire

5.1 Introduction **101**
5.2 Principe de l'algorithme IE **104**
5.3 Les trois types d'événements influenceurs **106**
5.3.1 Événements influenceurs du support : événements de disparition 107
5.3.2 Événements influenceurs de la confiance 107
5.3.3 Événements influenceurs de la distance 108
5.4 Intégration de l'algorithme IE dans un algorithme de fouille de règles d'épisode **109**
5.4.1 Se baser sur un algorithme traditionnel 110
5.4.2 Se baser sur un algorithme dédié 111
5.4.3 Synthèse 112
5.5 Expérimentations **112**
5.5.1 Temps d'exécution 113
5.5.2 Caractéristiques des événements influenceurs 114
5.5.2.1 Événements de disparition 115
5.5.2.2 Événements influenceurs de confiance 115
5.5.2.3 Événements influenceurs de distance 119
5.5.2.4 Discussion 122
5.6 Synthèse **122**

5.1 Introduction

Dans ce chapitre, nous nous intéressons au problème d'influence sur des futurs événements. Nous introduisons une notion nouvelle "les événements influenceurs" avec l'algorithme IE (Influencer Events) qui permet de les détecter, ce que nous proposons comme contribution.
Comme nous l'avons mentionné dans la section 2.8 du chapitre "État de l'art", très peu de travaux s'intéressent à influencer des événements futurs. Nous avons montré que le seul lien que nous pouvons trouver entre fouille de données et influence sur des événements réside dans l'extraction de motifs dits "d'une grande utilité". Ces motifs sont souvent extraits dans le but d'analyser les données par l'identification des combinaisons utiles et non pour des objectifs liés aux

événements futurs ou pour influencer des événements. La manière dont ces motifs sont extraits (en se focalisant sur le motif ou l'épisode dans son entier) ne permet pas d'étudier l'évolution de ces motifs ou épisodes, c'est-à-dire de leur utilité, au cours de leur construction.
De plus, pour la prédiction dans une séquence d'événements, qui est le focus dans ce manuscrit, une difficulté se présente en raison de la vélocité et de la variété des événements. En principe, pour influencer un événement futur, il faut étudier quel événement il faut injecter dans le contexte de l'événement futur et quelle est la nature de l'influence qu'il est censé avoir. En raison de la vélocité et de la variété des événements dans une séquence d'événements, un événement futur, qu'on a pour but d'influencer, peut rapidement être entouré (en raison de la vélocité) par de nombreux autres événements (en raison de la variété) sur lesquels l'influence de l'événement injecté n'a pas été étudiée. Pour cette raison, il est possible que l'événement injecté impacte de manière souhaitable ou non d'autres événements, ce qui doit être pris en compte dans ce contexte.
Afin d'effectuer une telle influence, nous proposons dans ce chapitre la détection d'un nouveau type d'événements dans le contexte de règles d'épisode et partons de l'hypothèse suivante :
Hypothèse : dans une séquence d'événements, un événement peut avoir un impact, souhaitable ou non, sur d'autres événements, à savoir certaines caractéristiques de ces autres événements. Cet impact est détectable : l'influence d'un événement sur une caractéristique précise d'un autre événement peut être détectée.
Rappelons que les règles d'épisode sont généralement utilisées pour la prédiction d'événements (Cho et al., 2007). L'exploitation de la règle $R : P \rightarrow Q$ permet la prédiction de l'occurrence de sa conséquence Q une fois que l'antécédent P apparaît dans la séquence d'événements. Cependant, l'apparition d'un événement, que nous appellerons e, après l'apparition de l'antécédent P, peut impacter l'occurrence de la conséquence de la règle Q, pour empêcher l'apparition de Q. Nous appellerons un tel événement un "événement influenceur".
Dans ce chapitre, nous nous intéressons à la détection d'**événement influenceur**. C'est une nouvelle notion, que nous proposons comme contribution principale de ce chapitre. À notre connaissance, la détection des événements influenceurs n'a jamais été proposée dans la littérature de l'extraction d'épisodes ou de règles d'épisode. La détection de tels événements représente non seulement un nouveau défi, mais a également une grande importance, surtout dans le contexte de la prédiction d'événements. Nous nous sommes donc confrontés au défi de **la détection des événements dits influenceurs associés à des règles d'épisode dans une séquence d'événements**.
Afin de comprendre l'importance applicative de la détection d'événements influenceurs, considérons l'exemple suivant. Supposons que le processus de fouille de règles d'épisode a extrait un événement influenceur e associé avec la règle suivante $R : P \rightarrow Q$. Si l'influence de e réside dans l'augmentation de la confiance de R, cela veut dire que lorsque e apparaît après P, la probabilité de l'apparition de Q est augmentée. Si l'objectif de l'application est de maximiser l'apparition de Q dans la séquence, on doit forcer l'apparition de e une fois que P est apparu. À l'opposé, si e diminue la confiance de la règle R, on doit garantir que e n'apparaîtra pas.
La détection des événements influenceurs est intéressante pour de nombreuses applications. Prenons maintenant quelques exemples applicatifs :

— Dans le cadre de réseaux sociaux, les entreprises (une banque par exemple) ont pour but l'amélioration ou au moins le maintien de leur e-réputation. Par conséquent, elles sont très intéressées par l'identification des événements qui peuvent positivement impacter leur e-réputation. Considérons la règle d'épisode suivante : R : *(se plaindre de taux d'intérêt élevé de la banque), (menacer de quitter la banque)* $\rightarrow$ *(e-réputation négative de la banque).* Cela veut dire que si les deux événements de l'antécédent de la règle R apparaissent, la e-réputation de la banque va être négativement impactée. Un événement influenceur associé

avec cette règle peut être *e : (diffuser des messages de marketing sur les taux d'intérêts)*, qui diminue la probabilité de l'apparition de la conséquence de la règle R. Afin de préserver sa e-réputation, la banque doit donc diffuser des messages de marketing (l'événement e). À noter que la diffusion de messages de marketing dans un autre contexte peut avoir un effet inverse.

— Dans le contexte d'achats en ligne, on pourrait s'intéresser à l'identification des événements qui augmentent les achats des clients. Par exemple, lorsqu'un client achète une caméra, puis un trépied, une règle peut identifier qu'il va probablement acheter un objectif pour sa caméra : *R : (achat caméra), (achat trépied)* $\rightarrow$ *(achat objectif de caméra)*. Un événement influenceur dans ce cas peut être une promotion sur les objectifs des caméras, ce qui augmente la probabilité que le client achète un nouvel objectif. Par conséquent, si le magasin souhaite maximiser ses ventes, une promotion sur les objectifs doit être proposée au client. Notons qu'une telle promotion peut n'avoir aucun impact dans un autre contexte : pour l'achat d'un téléphone portable par exemple.

— Dans le cadre du contrôle des épidémies de maladies, comme par exemple la détection des attaques à l'anthrax (Wong et al., 2005), les informations sur les patients (consignées dans les fiches patients : date d'admission, âge, symptômes, traitements proposés, code postal, etc.) dont disposent les hôpitaux, sont en règle générale exploitées. Un exemple de règle fouillée peut être : R : ($symptôme_A$, $ville_1$ humide), ($symptôme_B$, $ville_1$ humide) $\rightarrow$ ($symptôme_A$, $ville_2$). Cette règle représente la propagation de $symptôme_A$ vers une ville voisine $ville_2$ lorsque les événements $symptôme_A$ et $symptôme_B$ sont présents dans une ville humide. Dans ce contexte, un événement influenceur e pertinent a pour effet de faire disparaître la conséquence de la règle, c'est-à-dire stopper la propagation de l'épidémie vers les autres villes. Concrètement, l'événement influenceur e peut être un traitement dédié à ce type de symptômes et aux caractéristiques des villes infectées. Notons qu'un traitement dédié aux symptômes présents peut n'avoir aucun impact voire un impact inverse dans le cas où les caractéristiques de villes infectées ne sont pas prises en compte lors de la proposition du traitement.

Dans ce chapitre, plusieurs contributions sont apportées :

— Nous introduisons un tout nouveau concept : les événements influenceurs dans des règles d'épisode.

— Nous définissons trois nouvelles mesures associées aux événements influenceurs : la mesure d'influence sur le support (la mesure de disparition), la mesure d'influence sur la confiance, et la mesure d'influence sur la distance, afin de détecter respectivement les événements influenceurs du support (les événements de disparition), les événements influenceurs de la confiance, et les événements influenceurs de la distance.

— Nous proposons un algorithme de détection des événements influenceurs appelé IE ($InfluencerEvents$), et nous proposons deux manières pour intégrer cet algorithme dans un algorithme de fouille de règles d'épisode.

Le reste de ce chapitre est organisé comme suit. Dans la section 5.2, nous présentons le principe de l'algorithme IE. Puis, nous introduisons les trois types d'événements influenceurs dans la section 5.3. Ensuite, dans la section 5.4 nous examinons différentes façon d'intégrer IE dans un algorithme de fouille de règles. Nous procédons à l'évaluation expérimentale de l'algorithme dans la section 5.5. Enfin, nous concluons dans la section 5.6.

5.2 Principe de l'algorithme IE

L'algorithme IE (Influencer Events) que nous proposons repose sur un ensemble de règles d'épisode ER résultat d'un algorithme de fouille de règles d'épisode. Il est important de rappeler que ces règles sont fréquentes et confiantes. Soit $R : P \rightarrow Q$ une règle d'épisode résultat d'un algorithme de fouille de règles d'épisode. R est associé à plusieurs caractéristiques comme par exemple la valeur de son support et de sa confiance. Considérons également $e \in I$ un événement faisant partie l'ensemble des événements I.
Le principe de IE est d'examiner le changement de caractéristiques de la règle R avec la présence de e en tant que suffixe de son antécédent, ce qui forme la règle $R' : P, e \rightarrow Q$ (Algorithme 6 lignes 1 et 2). Autrement dit, le principe de cet algorithme est de comparer les caractéristiques de chaque couple R et R' en posant les questions suivantes : À quel point les caractéristiques de R sont différentes avec la présence de e en tant que suffixe de son antécédent ? Est-ce que l'apparition de l'événement e change significativement une ou plusieurs caractéristiques de R' par rapport à R ? Si c'est le cas, nous appelons e un événement influenceur.
Présentons maintenant la définition formelle d'un événement influenceur.

Définition 5.2.1. *Un événement e est considéré comme un* **événement influenceur** *associé à la règle d'épisode $R : P \rightarrow Q$, si certaines caractéristiques de la règle $R' : P, e \rightarrow Q$, comme par exemple le support ou la confiance, sont significativement différentes des celles de la règle R. Évidement, un événement influenceur a un impact dans le contexte d'au moins une seule règle d'épisode, et peut n'avoir aucun impact sur d'autres règles d'épisode.*

Il est important de noter que le support de la règle R' est par définition inférieur ou égal à celui de R : $supp(R') \leq supp(R)$, car comme elle contient un événement supplémentaire (l'événement e) par rapport à R, le nombre de ses occurrences est égal à ou plus petit que celui de R (par propriété d'anti-monotonicité). À noter que R' peut : (i) être une règle fréquente $supp(R') \geq minsupp$, (ii) avoir un support très faible $supp(R') < minsupp$, (iii) avoir un support égal à zéro $supp(R') = 0$ dans le cas où aucune occurrence de R n'est trouvée dans la séquence d'événements, (iii) avoir une confiance plus petite ou plus grande que celle de R, etc.
Nous pouvons distinguer trois scénarios dans lesquels nous pouvons détecter un événement influenceur. Ces scénarios sont illustrés dans le tableau 5.1.

1. **Scénario 1** (Influence sur le support) : Nous nous intéressons aux cas où le support est fortement impacté. Il est important de mentionner que lorsque le support de R' est très faible, les autres caractéristiques, comme la confiance par exemple, ne peuvent pas être calculées car l'analyse ne sera ni significative ni statistiquement fiable. Lorsque R' est non fréquente, cela représente la disparition de la règle R' et donc de la règle R après l'apparition de l'événement e en tant que suffixe de son antécédent. Nous nous sommes intéressés à la raison pour laquelle le support de la règle peut devenir faible voire nul. Afin de montrer que l'occurrence de l'événement e n'est pas toujours le responsable du support faible de la règle R', nous identifions deux scénarios. Prenons l'exemple du *Scénario'* 1 présenté dans le tableau 5.1. Soit $R : a, b \rightarrow z$ une règle fréquente. La présence de l'événement e en tant que suffixe de son antécédent forme la règle $R' : a, b, e \rightarrow z$ qui n'est pas fréquente : $supp(R' : a, b, e \rightarrow z) = 1$. Cependant, le support du nouvel antécédent a, b, e est également faible, ce qui justifie le faible support de la règle R'. Nous concluons que e est un événement hors contexte de a, b : il n'existe aucun lien entre a, b et e. Par conséquent, e n'est pas considéré comme le responsable du faible support de la règle R' et donc il n'est pas un événement influenceur du support de la règle.

À l'opposé, dans le *Scénario* 1 du tableau 5.1, l'événement e apparaît fréquemment après l'apparition de a, b (le support de a, b, e est élevé), c'est-à-dire qu'il existe une association forte entre a, b et e. Dans ce cas, la présence de e peut être considéré comme la raison pour laquelle le support de la règle R' est très faible et donc la raison qui explique que la règle a disparu.

2. **Scénario 2** (Influence sur la confiance) : Lorsque le support de la règle R' tend à diminuer par rapport à celui de R tout en restant supérieur au seuil $minsupp$ (R' est fréquente), l'impact porté par e sur le support n'est donc pas significatif. Cependant, une autre caractéristique peut être impactée : la confiance de R (rappelons que la confiance est calculée uniquement pour des règles fréquentes) qui peut avoir une valeur significativement différente de celle de R : en diminution ou en augmentation. Prenons l'exemple du *Scénario* 2 (tableau 5.1). La règle $R' : a, b, e \rightarrow z$ formée est fréquente, ce qui permet de tracer d'autres caractéristiques de manière statistiquement fiable comme la confiance de la règle. Notons que la règle R' est non confiante ($conf(R') < minconf$). Dans ce cas, nous pouvons considérer que l'événement e est le responsable de la confiance élevée de la règle.
3. **Scénario 3** (Influence sur la distance) : Lorsque la règle R' est fréquente, une autre caractéristique peut être impactée : la distance entre le préfixe et la conséquence de la règle. En effet, pour chaque occurrence de la règle R, cette distance peut être mesurée de manière à ce qu'elle reflète la tendance d'éloignement de la conséquence. La présence d'un événement e en tant que préfixe de l'antécédent de R peut induire une diminution du nombre d'occurrences de R et peut donc influencer la tendance d'éloignement de la conséquence : en le rapprochant ou en l'éloignant. Par conséquent, examiner la distance entre le préfixe et la conséquence d'une règle permet de détecter les cas d'éloignement ou de rapprochement de la conséquence lorsque l'événement e est présent en tant que suffixe de l'antécédent de la règle. Cette caractéristique est particulière, car la distance entre le préfixe et la conséquence peut être différente d'une occurrence à l'autre au sein de la même règle. Pour cela, nous proposons d'utiliser la médiane de la distance entre le préfixe et la conséquence de la règle, qui sera notée $médiane(R)$, afin de n'avoir qu'une seule valeur qui représente cette distance.

   Étant donné que la distance entre le préfixe et la conséquence peut varier entre les occurrences de la même règle, il nous faut donc une valeur qui reflète la distribution de la distance, c'est-à-dire une valeur qui exprime de manière précise la distance entre le préfixe et la conséquence et qui ne sera pas influencée par la présence de valeurs très élevées de la distance ou au contraire très petites. Pour cela, la médiane de la distance est une bonne solution (et non pas la distance moyenne par exemple).

   Prenons l'exemple du *Scénario* 3 présenté dans le tableau 5.1. La règle R' est fréquente : le nombre de ses occurrences est suffisant pour permettre de vérifier la distance médiane au sein de ses occurrences de manière statistiquement fiable (notons que pour un très petit nombre d'occurrences, nous considérons qu'il n'est pas fiable de vérifier la distance médiane). Notons que la distance médiane de la règle R' est augmentée, ce qui veut dire que la conséquence z tend à s'éloigner du préfixe de la règle. Dans ce cas, nous pouvons considérer que l'événement e est le responsable de l'éloignement de la conséquence et donc qu'il influence la distance entre le préfixe et la conséquence de la règle.

L'algorithme IE exige donc que chaque règle soit associée à trois valeurs : (i) le support de la règle, (ii) la confiance de la règle, (iii) la médiane de la distance entre le préfixe et la conséquence de la règle. Cette dernière valeur n'est pas gérée automatiquement par les algorithmes de la

fouille de règles d'épisode. Cependant, elle peut facilement être calculée.
Un pseudo code de l'algorithme IE est présenté dans Algorithme 6.

TABLE 5.1 – Exemple de plusieurs scénarios en cas d'ajout d'un événement e en tant que suffixe de l'antécédent d'une règle jouet $R : a, b \rightarrow z$, pour $minsupp = 20$ et $minconf = 0.8$ ("- -" représentent des valeurs non importantes pour étudier le scénario lié).

Scénario	$R : a, b \rightarrow z$			$R' : a, b, e \rightarrow z$			
	$supp(R)$	$conf(R)$	$médiane(R)$	$supp(R')$	$supp(a, b, e)$	$conf(R')$	$médiane(R')$
Scénario' 1 influence sur le support	30	1	50	1	1	- -	- -
Scénario 1 influence sur le support	30	1	50	1	20	- -	- -
Scénario 2 influence sur la confiance	30	1	50	25	- -	0,5	- -
Scénario 3 influence sur la distance	30	1	50	25	- -		70

5.3 Les trois types d'événements influenceurs

Dans cette section, nous allons présenter les types d'événements influenceurs que nous proposons en commençant par introduire un type générique. Nous présentons également une proposition de mesures pour la détection de ces événements.
De manière générale et afin de mesurer l'impact de l'événement e sur la règle $R : P \rightarrow Q$, nous proposons une mesure d'influence générique $Infl(P \rightarrow Q, e)$. Soit $caractéristique_i$ la mesure d'une caractéristique donnée i de la règle. Dans ce cas, la mesure d'influence évalue le changement de la valeur de $caractéristique_i$ lors de la présence ou non de l'événement e en tant que suffixe de l'antécédent de la règle R, sous plusieurs conditions en fonction de la nature de la caractéristique. La mesure d'influence générique est représentée comme suit :

$$Infl(P \rightarrow Q, e) = \frac{caractéristique_i(P, e \rightarrow Q) \; - \; caractéristique_i(P \rightarrow Q)}{caractéristique_i(P \rightarrow Q))} \quad : \text{sous plusieurs conditions} \tag{5.1}$$

Les valeurs de la mesure $Infl(P \rightarrow Q, e)$ varient entre -1 et $+\infty$. À noter, qu'une valeur d'influence égale à 0 signifie qu'aucun changement n'est apporté par l'événement e et donc aucune influence de e n'existe sur la règle. Une valuer négative de $Infl(P \rightarrow Q, e)$ signifie que l'occurrence de l'événement e en tant que suffixe de l'antécédent de R augmente la valeur de $caractéristique_i$. Dans le cas contraire, une valeur positive de la mesure d'influence signifie que l'occurrence de l'événement e en tant que suffixe de l'antécédent de R diminue la valeur $caractéristique_i$. Un seuil d'influence θ_{Infl} doit être fixé afin de determiner les cas d'influence les plus significatifs.
Les trois types d'événements influenceurs, que nous allons présenter maintenant, diffèrent par la caractéristique évaluée et par les conditions exigées sur les règles $R : P \rightarrow Q$ et $R' : P, e \rightarrow Q$.

5.3.1 Événements influenceurs du support : événements de disparition

Soit R une règle d'épisode fréquente et confiante. Comme nous l'avons mentionné, le nombre d'occurrences de la règle $R' : P, e \rightarrow Q$ est plus petit ou égal que celui de R. Par conséquent, le support de la règle R' peut devenir inférieur à *minsupp*, la règle devient alors non fréquente.
Comme nous l'avons présenté dans la Scénario 1 de la section 5.2, le support faible de la règle R ne signifie pas forcément que l'événement e est la raison pour laquelle la règle disparaît, car l'événement e peut tout simplement être hors contexte de la règle. Afin de détecter les événements influenceurs du support, nous proposons de comparer les valeurs du support de R et R' sous la condition que (i) R est fréquente et confiante, (ii) R' est non fréquente et, (P, e) l'antécédent de R' est fréquent. Lorsque le support de la règle R' est significativement différent de celui de R et qu'il passe sous *minsupp*, nous considérons que l'événement e est un événement influenceur du support ou autrement dit un événement de disparition. Cette différence est qualifiée par la mesure d'influence sur le support (la mesure de disparition) que nous proposons définir comme suit :

Définition 5.3.1. *L'influence de l'événement e sur le support de la règle R est calculée par la* **mesure d'influence sur le support (la mesure de disparition)**. *Cette mesure évalue le score de diminution du support de la règle R (supp($P \rightarrow Q$)) lorsque l'on ajoute l'événement e en tant que suffixe de son antécédent (supp($P, e \rightarrow Q$)), comme suit :*

$$Infl_{supp}(P \rightarrow Q, e) = \frac{supp(P,e \rightarrow Q) \ - \ supp(P \rightarrow Q)}{supp(P \rightarrow Q))} \quad : \quad \begin{array}{l} P \rightarrow Q \text{ est fréquente \& confiante} \\ P, e \rightarrow Q \text{ est non fréquente} \\ (P, e) \text{ est fréquent} \end{array} \tag{5.2}$$

Étant donné que le support de la règle $R' : P, e \rightarrow Q$ ne peut que diminuer, les valeurs de la mesure d'influence sur le support varient entre -1 et 0.
Nous fixons également $\theta_{Infl_{supp}}$, un seuil minimal d'influence sur le support, qui permet de déterminer si un événement e est influenceur du support d'une règle ou non (voir Algorithme 6 ligne 4 et la section 5.5 d'Expérimentations).

5.3.2 Événements influenceurs de la confiance

Nous nous intéressons maintenant à l'influence portée par certains événements sur la confiance de la règle, selon le Scénario 2 (section 5.2). Nous proposons de comparer les valeurs de confiance de R et R', toujours sous la condition que les deux règles sont fréquentes (voir Algorithme 6 lignes 7, 8 et 10). Lorsque la confiance de la règle R' est significativement différente de celle de R, nous considérons dans ce cas que l'événement e est un événement influenceur de la confiance de la règle R. Nous proposons une mesure d'influence sur la confiance qui quantifie cette différence :

Définition 5.3.2. *L'influence portée par l'événement e sur la confiance de la règle R est calculée par la* **mesure d'influence sur la confiance**. *Cette mesure représente le score d'augmentation ou de diminution de la confiance de la règle R (conf(R)) lorsque l'on ajoute l'événement e en tant que suffixe de son antécédent (conf(R, e)), comme présenté dans l'équation (5.3).*

$$Infl_{conf}(P \rightarrow Q, e) = \frac{conf(P,e \rightarrow Q) \ - \ conf(P \rightarrow Q)}{conf(P \rightarrow Q)} \qquad : \text{si } P \rightarrow Q \text{ \& } P, e \rightarrow Q \text{ sont fréquentes} \tag{5.3}$$

Comme dans le cas de la mesure d'influence générique, les valeurs de la mesure $Infl_{conf}(P \rightarrow Q, e)$ varie entre -1 et $+\infty$. À rappeler, qu'une valeur de risque égale à 0 signifie qu'aucun changement n'est apporté par l'événement e.
La mesure d'influence sur la confiance peut avoir une valeur positive ou négative :

— Si la présence de l'événement e dans R' maintient les occurrences de la règle pour lesquelles la confiance de R' est plus grande que celles de R, dans ce cas $Infl_{conf}(P \rightarrow Q, e)$ a une valeur positive et e représente un événement qui accroît la confiance de la règle et donc la probabilité d'apparition de la conséquence suite à l'apparition de l'antécédent.

— À l'opposé, si la présence de l'événement e dans R' maintient les occurrences de la règle pour lesquelles la confiance est plus petite que celle de R, alors $Infl_{conf}(P \rightarrow Q, e)$ a une valeur négative et e représente un événement qui diminue la confiance de la règle et donc la probabilité de l'apparition de la conséquence suite à celle de l'antécédent.

Soit $\theta_{Infl_{conf}}$ un seuil minimal de l'influence sur la confiance, qui permet de déterminer si un événement e est un événement influenceur de la confiance d'une règle ou non (voir Algorithme 6 lignes 8 et 10, et la section 5.5 du chapitre d'Expérimentations). Nous pouvons distinguer deux cas selon que la valeur de $\theta_{Infl_{conf}}$ est positive ou négative, comme suit :

— Lorsque $\theta_{Infl_{conf}} > 0$, les événements pour lesquels $Infl_{conf}(P \rightarrow Q, e) \geq \theta_{Infl_{conf}}$ sont considérés comme des événements influenceurs de la confiance de R (voir Algorithme 6 ligne 8). Dans ce cas, l'apparition de l'événement e augmente significativement la confiance de la règle et donc la probabilité de l'apparition de la conséquence. Autrement dit, l'événement e renforce l'association entre l'antécédent P et la conséquence Q de la règle R en cours d'étude.

— Lorsque $\theta_{Infl_{conf}} < 0$, les événements pour lesquels $Infl_{conf}(P \rightarrow Q, e) \leq \theta_{Infl_{conf}}$ sont considérés comme des événements influenceurs de la confiance (voir Algorithme 6 ligne 10). Dans ce cas, l'apparition de l'événement e diminue significativement la confiance de la règle et donc la probabilité d'apparition de la conséquence. Autrement dit, l'événement e réduit l'association entre l'antécédent P et la conséquence Q de la règle R.

5.3.3 Événements influenceurs de la distance

Nous allons maintenant proposer un nouveau type d'événements influenceurs en suivant le Scénario 3 (section 5.2). Soit $R : P \rightarrow Q$ une règle d'épisode. La médiane de la distance entre son préfixe et sa conséquence est notée $médiane(R)$ (ou $médiane(P \rightarrow Q)$). Considérons également la règle $R' : P, e \rightarrow Q$. Nous nous intéressons ici à l'influence portée par l'événement e sur la règle R, sous la condition que R et R' soient toutes les deux fréquentes, ce qui garantit une analyse significative et statistiquement fiable.
Si la distance médiane entre le préfixe et la conséquence dans la règle R' est significativement différente de celle dans R, nous concluons que l'événement e est un événement influenceur de la distance, associé à la règle d'épisode R (voir Algorithme 6 lignes 7, 12 et 14). Nous proposons la mesure d'influence sur la distance suivante :

Définition 5.3.3. *L'influence sur la distance portée par l'événement e sur la règle R est évaluée par la* **mesure d'influence sur la distance**. *Cette mesure représente l'augmentation ou la diminution de la valeur de médiane de la distance entre le préfixe et la conséquence de la règle R', par rapport à celle de la règle R, sous la condition que les deux règles soient fréquentes, comme*

présenté dans l'équation (5.4).

$$Infl_{dist}(P \rightarrow Q, e) = \frac{médiane(P,e \rightarrow Q) \ - \ médiane(P \rightarrow Q)}{médiane(P \rightarrow Q)} \qquad : si\ P \rightarrow Q\ \&\ P, e \rightarrow Q\ sont\ fréquentes \tag{5.4}$$

Comme dans le cas générique, les valeurs de la mesure $Infl_{dist}(R, e)$ varie entre -1 et $+\infty$. Rappelons qu'une valeur de risque égale à 0 signifie qu'aucun changement n'est apporté par l'événement e.
La mesure de risque de distance $Infl_{dist}(P \rightarrow Q, e)$ peut avoir une valeur positive ou négative :

- Si la présence de l'événement e dans R' maintient les occurrences de la règle pour lesquelles la distance médiane est plus grande que celle de R, alors $Infl_{dist}(R, e)$ a une valeur positive et e représente un événement qui éloigne la conséquence du préfixe de la règle.
- À l'opposé, si la présence de l'événement e dans R' maintient les occurrences de la règle pour lesquelles la distance médiane est plus petite que celle de R, alors $Infl_{dist}(R, e)$ a une valeur négative et e représente un événement qui fait rapprocher la conséquence du préfixe de la règle.

Nous proposons l'utilisation de $\theta_{Infl_{dist}}$ seuil de risque de distance qui sera utilisé pour déterminer si l'événement e est un événement influenceur de la distance ou non (voir Algorithme 6 lignes 12 et 14, et la section 5.5 du chapitre d'Expérimentations). Nous pouvons distinguer deux cas selon que la valeur concrète de $\theta_{Infl_{dist}}$ est positive ou négative :

- Lorsque $\theta_{Infl_{dist}} > 0$, alors les événements influenceurs sont ceux pour lesquels $Infl_{dist}(P \rightarrow Q, e) \geq \theta_{R_{dist}}$, ce qui veut dire que l'apparition de l'événement e fait éloigner la conséquence de la règle (voir Algorithme 6 ligne 12).
- Lorsque $\theta_{Infl_{dist}} < 0$, nous considérons les cas où $Infl_{dist}(P \rightarrow Q, e) \leq \theta_{Infl_{dist}}$, ce qui veut dire que l'apparition de l'événement e fait rapprocher la conséquence de la règle (voir Algorithme 6 ligne 14).

5.4 Intégration de l'algorithme IE dans un algorithme de fouille de règles d'épisode

Comme nous l'avons mentionné dans les chapitre précédents, nous nous intéressons dans ce manuscrit à la prédiction au plus tôt. Nous partons de l'hypothèse que plus tôt la prédiction est effectuée, plus efficacement les événements prédits sont influencés. Nous nous intéressons donc dans cette section à la détection des événements influenceurs dédiés à influencer des événements prédits au plus tôt.
Afin d'effectuer une prédiction tôt, les algorithmes de la littérature procèdent en fouillant des règles avec un antécédent le plus petit possible en terme de nombre d'événements. Fouiller de telles règles consiste soit à fouiller l'ensemble complet de règles puis conserver uniquement les règles ayant un antécédent le plus petit possible en effectuant une étape de post-traitement, soit à appliquer un algorithme dédié, comme l'algorithme proposé par (Rahal et al., 2004) ou l'algorithme $DEER$ que nous avons proposé dans le chapitre 3.
Nous allons présenter dans cette section deux propositions pour l'intégration de l'algorithme IE dans un algorithme traditionnel de fouille de règles et dans un algorithme dédié à l'extraction de règles avec un antécédent le plus petit possible. Nous mènerons une étude théorique pour comparer ces deux propositions en terme de performance.
Il est important de préciser que peu importe l'algorithme de fouille de règles adopté, l'ensemble résultat d'événements influenceurs reste le même. Cependant, la consommation de ressources en

Algorithm 6: IE : Influencer Events

Données: ER : l'ensemble des règles d'épisode, I : l'ensemble des événements, $\theta_{Infl_{supp}}$, $\theta_{Infl_{dist}}$, $\theta_{Infl_{conf}}$

Résultat: Les événements influenceurs

1 **pour chaque** $(R : P \rightarrow Q) \in ER$ **faire**
2 **pour chaque** $e \in I$ **faire**
/* la présence de e forme la règle $R' : P, e \rightarrow Q$ */
3 **si** $P \rightarrow Q$ *est fréquente et confiante* $\wedge$ $P, e \rightarrow Q$ *est non fréquente* $\wedge$ (P, e) *est fréquent* **alors**
4 **si** $Infl_{supp}(P \rightarrow Q, e) \leq \theta_{Infl_{supp}}$ **alors**
5 e est un événement influenceur du support : événement de disparition
6 **sinon**
7 **si** $P \rightarrow Q \wedge P, e \rightarrow Q$ *sont fréquentes* **alors**
8 **si** $Infl_{conf}(P \rightarrow Q, e) > 0 \wedge Infl_{conf}(P \rightarrow Q, e) \geq \theta_{Infl_{conf}}$ **alors**
9 e est un événement influenceur de la confiance en augmentation
10 **si** $Infl_{conf}(P \rightarrow Q, e) < 0 \wedge Infl_{conf}(P \rightarrow Q, e) \leq \theta_{Infl_{conf}}$ **alors**
/* pour $\theta_{Infl_{conf}} < 0$ */ ;
11 e est un événement influenceur de la confiance en diminution
12 **si** $Infl_{dist}(P \rightarrow Q, e) > 0 \wedge Infl_{dist}(P \rightarrow Q, e) \geq \theta_{Infl_{dist}}$ **alors**
13 e est un événement influenceur de la distance en éloignement
14 **si** $Infl_{dist}(P \rightarrow Q, e) < 0 \wedge Infl_{dist}(P \rightarrow Q, e) \leq \theta_{Infl_{dist}}$ **alors**
/* pour $\theta_{Infl_{dist}} < 0$ */ ;
15 e est un événement influenceur de la distance en rapprochement

temps varie significativement en fonction de l'algorithme adopté.

5.4.1 Se baser sur un algorithme traditionnel

Rappelons que les algorithmes de fouille de règles d'épisodes sans contraintes extraient dans un premier temps les épisodes, puis forment les règles d'épisode en identifiant les conséquences (un événement de l'épisode) pour lesquelles la règle est confiante. Par conséquent, au cours de la construction d'un épisode, généralement par l'ajout itératif d'événements, la conséquence de la règle liée n'est pas connue à ce stade. Ainsi, l'influence d'un événement dans une règle épisode ne peut pas être étudiée avant la fin de la construction de l'épisode et de la règle. Pour cela, une étape de post-traitement est exigée. Nous proposons donc les sous-étapes suivantes de post-traitement :

1. Dans l'ensemble de règles d'épisode ER fouillées par l'algorithme traditionnel, identifier les règles d'épisode dans lesquelles l'antécédent est le plus petit possible, ce qui est nécessaire pour la prédiction tôt. Ces règles représenteront les règles R nécessaires pour l'algorithme IE (Algorithme 6).
2. Identifier les couples R, R' de manière à ce que R' diffère de R par la présence d'un événement supplémentaire e (en tant que suffixe de l'antécédent) qui est un événement influenceur candidat. Chaque règle R doit être associée à de nombreuses règles R' qui diffèrent uniquement par le type de l'événement e. Cette étape semble consommer beaucoup

de ressources en temps. En effet, il peut arriver qu'une règle particulière R' ne soit pas extraite par l'algorithme en raison de la manière dont les épisodes sont formés. Pour cela, il faut former les règles R' manquantes en ajoutant un événement e qui n'est pas inclus dans une autre règle R'.

Ces règles R' formées pendant la deuxième sous-étape de post-traitement peuvent être fréquentes ou non fréquentes mais elles sont forcément non confiantes (sinon elles auraient été extraites dès le début par l'algorithme). Former des règles fréquentes et non confiantes est faisable par un algorithme traditionnel mais reste quand-même très coûteux en temps et en mémoire. Cependant, la consommation de ressources nécessaires pour former des règles non fréquentes peut exploser en raison de leur nombre énorme. De plus, la majorité de ces règles et de leur antécédents sont potentiellement inexistantes, c'est-à-dire qu'aucune occurrence ne peut être trouvée pour ces règles R' ou pour leur antécédents, ce qui représente une perte importante du temps d'exécution. Ainsi, nous considérons qu'il est relativement impossible de se baser sur un algorithme traditionnel pour former des règles non fréquentes et donc pour la détection des événements influenceurs du support (événements de disparition).
À noter que lorsque l'objectif est d'influencer des événements prédits sans contraintes (ne sont pas forcément prédits au plus tôt), toutes les règles fréquentes et confiantes sans aucune contrainte doivent être exploitées. Ces règles sont fouillées à la base par un algorithme de l'état de l'art. Par conséquent, il faut passer directement à la deuxième sous-étape de post-traitement.

5.4.2 Se baser sur un algorithme dédié

Comme nous l'avons mentionné, les algorithmes de la fouille de règles d'épisode dédiées à la prédiction au plus tôt (c'est-à-dire avec un antécédent le plus petit possible) extraient chaque règle en fixant sa conséquence tôt dans le processus de la fouille. Par conséquent, l'ensemble de règles résultat représente les règles R sur lesquelles repose l'algorithme IE (Algorithme 6).
Nous proposons de tirer profit de l'algorithme que nous avons proposé dans le chapitre 3 : l'algorithme $DEER$. Rappelons que $DEER$ procède à la fouille de règles d'épisode en trois étapes : (i) le préfixe de chaque règle est fixé, (ii) la conséquence est identifiée, (iii) l'antécédent est itérativement complété en ajoutant des événements à sa droite. La complétion de l'antécédent s'arrête une fois que la règle est fréquente et confiante (et donc que la règle est dite essentielle). Afin de détecter les événements influenceurs, nous proposons des modifier $DEER$ en ajoutant une étape supplémentaire : lors de la complétion de l'antécédent d'une règle, une fois que la règle est essentielle (fréquente et confiante), nous proposons de faire une seule itération supplémentaire afin d'étendre l'antécédent de la règle R en y ajoutant un seul événement e, ce qui permet de former la règle R' (nécessaires pour l'algorithme IE).
Cette proposition d'adaptation de l'algorithme $DEER$ comporte deux avantages :

- Les règles de type R' sont immédiatement construites après la construction de règles de type R. Ainsi, les couples R, R' sont identifiés durant le processus de la fouille, sans avoir besoin de faire un filtre très coûteux. Par conséquent, l'algorithme IE peut être appliqué directement.
- Lors de l'itération supplémentaire dans la complétion de l'antécédent dans $DEER$ et l'ajout d'un événement e dans l'antécédent, la règle résultat R' peut être non fréquente. Dans ce cas, en une seule étape supplémentaire, relativement non coûteuse, nous pouvons obtenir toutes les règles non fréquentes de type R'. L'influence portée par l'événement e peut être étudiée et les événements influenceurs du support peuvent être extraits.

Si l'on se base sur l'algorithme $DEER$, les événements influenceurs peuvent être détectés durant le processus de la fouille de règles d'épisode avec une consommation de ressources beaucoup

moins coûteuse qu'en se basant sur un algorithme traditionnel de fouille de règles, comme nous allons le montrer dans la section d'expérimentations qui suit.
À noter que lorsque l'objectif est d'influencer des événements prédits sans contraintes (ne sont pas forcément prédits au plus tôt), toutes les règles fréquentes et confiantes sans aucune contrainte doivent être exploitées. L'algorithme $DEER$ n'est pas dédié à ce type d'objectif car l'ensemble de règles extraites ne sera pas complèt. Une modification dans le cœur de l'algorithme est alors nécessaire.

5.4.3 Synthèse

Nous avons présenté dans les deux sections précédentes deux propositions pour intégrer IE dans un algorithme de fouille de règles d'épisodes de l'état de l'art et dans l'algorithme $DEER$ dédié pour la prédiction au plus tôt. Les étapes de cette intégration et la difficulté de chaque étape sont résumées dans le tableau 5.2. La difficulté est exprimée par : (i) "0" pour une étape qui ne nécessite aucun traitement, (ii) par "+" pour une étape plus difficile, (iii) par "++" pour une étape encore plus difficile, etc.
Nous pouvons conclure que l'algorithme IE a l'originalité d'être intégré dans tous les algorithmes de fouille de règles d'épisodes.

TABLE 5.2 – Synthèse des étapes et de difficultés pour l'intégration de IE dans un algorithme de la littérature $Algo_{littérature}$ et dans l'algorithme $DEER$.

Étape	Difficulté	
	$Algo_{littérature}$	$DEER$
R fréquentes et confiantes (sans d'autres contraintes)	0 (fouillées à la base)	$++++$ (l'ensemble de règles n'est pas complet)
R avec un antécédent le plus petit possible	++ (filtre en post-traitement)	0 (fouillées à la base)
couples R, R'	$++++$ (R' fréquentes : filtre en post-traitement & construction de règles manquantes)	+ (une itération supplémentaire)
	$+++++++++$ (R' non fréquentes : très couteux)	

5.5 Expérimentations

Dans cette section, nous présentons les études expérimentales que nous avons menées afin de valider l'algorithme IE. Pour cela, nous allons dans un premier temps présenter le temps d'exécution de l'algorithme IE en fonction de l'algorithme de fouille de règles d'épisode adopté. Puis, dans un deuxième temps nous nous focaliserons sur les caractéristiques des événements influenceurs détectés et des règles associées.

Il est important de mentionner que, à notre connaissance, aucun travail dans la littérature n'a été dédié à la détection des événements influenceurs associés aux règles d'épisode. Pour cette raison, l'algorithme IE n'est pas directement comparable avec aucun autre algorithme de l'état de l'art.
Dans cette expérimentation, nous utilisons le même corpus utilisé pour valider nos autres contributions, présenté dans la section 1.4.
Rappelons que pour la détection des événements influenceurs, l'algorithme IE exige un ensemble de règles d'épisode fouillées par un algorithme de fouille de règles. Par conséquent, nous proposons de nous baser sur deux algorithmes dans un contexte de prédiction au plus tôt : l'algorithme traditionnel de fouille de règles d'épisode $MINEPI$ (Mannila et al., 1997) et l'algorithme dédié à la fouille de règles essentielles $DEER$ (voir chapitre 3). Les deux algorithmes sont lancés pour $minsupp = 20$, $minconf = 0,4$ et $w = 100$. Ces valeurs de seuil de support et de confiance sont les mêmes utilisées pour lancer les expérimentations dans les chapitres précédents (section 3.5 et 4.4), et pour les quelles le nombre de règles fréquentes et confiantes est assez élevé. Quant à la taille des fenêtres, nous avons fixé une taille relativement grande afin de pouvoir étudier au mieux l'influence sur la distance (détecter les événements qui éloignent ou qui rapprochent considérablement les conséquences).

5.5.1 Temps d'exécution

Nous allons étudier l'efficacité, en terme de temps d'exécution, apportée en fonction de l'algorithme adopté pour la fouille de règles d'épisode. Dans le tableau 5.3, nous présentons en détail une comparaison de temps d'exécution en cas d'intégration de l'algorithme IE dans l'algorithme traditionnel $MINEPI$ et dans notre algorithme $DEER$ (voir section 5.4.3 pour plus de détails sur cette intégration, et section 3.4.3 pour plus de détails sur le choix de l'algorithme $MINEPI$).
Les seuils nécessaires pour la détection des événements influenceurs dans IE sont fixés comme suit [7] : $\theta_{Infl_{supp}} = -0,8$, $\theta_{Infl_{conf}} = \pm 0,7$, et $\theta_{Infl_{dist}} = \pm 0,2$. Le choix de ces valeurs sera justifiée dans les sections dédiées à l'étude de caractéristiques des événements influenceurs correspondants.
À rappeler qu'en cas d'adoption de l'algorithme $MINEPI$, le temps d'exécution requis pour cette intégration représente le temps nécessaire pour identifier les règles R avec un antécédent le plus petit possible et le temps nécessaire pour identifier les couples R, R', alors que si l'on choisit l'algorithme $DEER$, le temps requis représente uniquement le temps nécessaire pour l'itération supplémentaire : étendre les règles R pour obtenir R'.
Comme prévu (voir tableau 5.3 dernière ligne), l'intégration de IE dans l'algorithme $DEER$ est beaucoup plus efficace en terme de temps d'exécution : elle est 8 fois plus rapide qu'en cas d'adoption de $MINEPI$.
Dans le cas du choix de $MINEPI$, la première sous-étape de post-traitement dédiée à l'identification de règles avec la caractéristique d'avoir un antécédent le plus petit possible (voir section 5.4.1) exige un temps relativement élevé. Contrairement à $DEER$ où les règles sont à la base fouillées avec cette caractéristique, et donc le temps requis est nul (voir tableau 5.3 ligne "Identification de R").
De plus, pour $MINEPI$, la deuxième sous-étape de post-traitement dédiée à l'identification de couples R, R' (en considérant le temps nécessaire pour former les règles fréquentes R' manquantes, voir section 5.4.1) exige un temps relativement petit par rapport à $DEER$ dans lequel nous

7. Rappelons que la mesure d'influence sur le support a toujours une valeur positive et donc le seuil d'influence est positif, et que les mesures d'influence sur la confiance et sur la distance peuvent avoir des valeurs positives et négatives ce qui justifie leur seuils positifs et négatifs (voir section 5.3)

proposons de fouiller les règles de type R', incluant celles qui sont non fréquentes, en effectuant une seule itération supplémentaire (voir tableau 5.3 ligne "Identification de couples R, R'"). À rappeler que $MINEPI$ n'extrait pas les règles non fréquentes, donc ces deux temps ne sont pas tout à fait comparables.
Comme nous l'avons mentionné, une fois que les couples R, R' sont identifiés, l'algorithme de détection d'événements influenceurs IE exige un temps d'exécution faible. Ce temps reste le même, peu importe l'algorithme de fouille de règles adopté (voir tableau 5.3 lignes 4 et 5).
Nous tenons à mentionner que la détection des événements de disparition en se basant sur $MINEPI$ est extrêmement coûteux. Pour cette raison, ce temps d'exécution n'est pas présenté dans le tableau 5.3.
Nous pouvons conclure que IE est beaucoup plus efficace en terme de temps d'exécution lorsqu'il est intégré dans un algorithme dédié à la fouille de règles avant un antécédent le plus petit possible.

Table 5.3 – Comparaison de temps d'exécution (en secondes) pour l'intégration de l'algorithme IE dans deux algorithmes de fouille de règles d'épisode.

	Temps d'exécution en secondes	
	$MINEPI$	$DEER$
Identification de R	5320 s (*1ère sous-étape de*) (*post-traitement*)	0 s (*fouillée à la base*)
Identification de couples R, R'	160 s (*2ème sous-étape de*) (*post-traitement*)	575 s (1 *itération supplémentaire*)
Détection d'événements influenceurs de confiance & de distance	30 s	30 s
Détection d'événements de disparition	- -	50 s
Total de temps supplémentaire	**5510 s**	**655 s**

5.5.2 Caractéristiques des événements influenceurs

Nous allons maintenant étudier les caractéristiques des événements influenceurs détectés lorsque l'on se base sur notre algorithme $DEER$.
$76k$ règles sont fouillées par l'algorithme $DEER$, parmi lesquelles $44k$ règles possèdent un antécédent de taille 1 (un seul événement qui compose l'antécédent), et $32k$ règles avec un antécédent de taille 2. Ces règles représentent les règles de type R requises pour l'algorithme IE et représentent son point de départ (voir Algorithme 6).
Afin de détecter les événements influenceurs, l'algorithme dédié doit procéder à une itération supplémentaire pour former des règles de type R' (avec un seul événement supplémentaire dans l'antécédent). Deux cas sont possibles :

— La règle R' est non fréquente : elle sera utilisée pour étudier l'influence de l'événement ajouté sur la disparition de la règle R.
— La règle résultat R' est toujours fréquente : elle sera utilisée pour étudier l'influence sur la confiance et la distance (qui exige une règle fréquente) de l'événement ajouté, comme nous allons l'expliquer par la suite.

Ajouter un seul événement sur l'antécédent de $76k$ règles de type R donne environ $58k$ règles fréquentes R', la plupart d'entre elles ont un antécédent de taille 2 (seulement 17 règles ont un antécédent de taille 3). Ce nombre de règles R' fréquentes est relativement petit. Ce qui peut être justifié par le fait qu'ajouter un événement sur l'antécédent d'une règle fréquente résulte souvent en une règle non fréquente (qui ne fait pas partie des $58k$ règles étudiées à ce stade), ce qui nous allons présenter dans la section 5.5.2.1 pour la disparition de règles.
Dans la suite de ces expérimentations, nous allons étudier quatre éléments : (i) les cas d'influence R' : les cas où une règle R est influencée par un événement e, (ii) les règles influencées R : une seule règle R peut être influencée par de nombreux événements, ce qui représente plusieurs cas d'influence, (iii) les événements influenceurs : un événement peut influencer plusieurs règles à la fois, (iv) les conséquences influencées : un événement conséquence peut appartenir à plusieurs règles et donc peut être influencé plusieurs fois.

5.5.2.1 Événements de disparition

Nous nous focalisons dans cette section sur les cas de disparition des événements. Ces cas sont présentés dans la figure 5.1 pour $\theta_{Infl_{supp}} = -0,8$. 1181 règles R' sont obtenues, dans lesquelles l'événement ajouté en tant que suffixe de l'antécédent est la cause du faible support de la règle et donc de la disparition de la règle. Ces cas correspondent à : 282 règles disparues R distinctes, 65 événements distincts de disparition et 33 conséquences disparues distinctes.
Dans la figure 5.1 (A), nous présentons le nombre de fois qu'une règle est influencée (a disparu). Nous remarquons que 66% de règles disparues sont influencées au moins deux fois chacune car elles sont influencées dans au moins deux règles R'. Et que d'environ 16% d'entre elles sont influencées plus de 10 fois chacune (c'est-à-dire que plus de 10 événements les font disparaitre).
Dans la figure 5.1(B), nous présentons le nombre de fois qu'un événement influenceur fait disparaître des règles. Nous montrons que la plupart des événements de disparition influencent plusieurs règles à la fois. Ces événements ont une grande influence sur le support de ces règles, et ils les font disparaître.
En outre, nous remarquons dans la figure 5.1(C), qui représente le nombre de fois qu'une conséquence a disparu, que parmi les 33 conséquences disparues, certaines d'entre elles sont influencées plus de 100 fois, ce qui veut dire qu'elles disparaissent dans de nombreuses règles. Une analyse plus approfondie montre que cette disparition est due à un petit nombre d'événements et concerne quelques conséquences.
Prenons maintenant un exemple concret de règle disparue : R :*(prix non adéquat)* $\rightarrow$ *(acheter un produit)* avec $supp(R) = 236$. Lorsque l'on ajoute l'événement e : *(client en attente)*, la règle résultat R' : *(prix non adéquat), (client en attente)* $\rightarrow$ *(acheter un produit)* a un support égal à 1, tandis que le support de son antécédent est égal à 97 et donc nous pouvons calculer $Infl_{supp}$*(R, (acheter un produit))*$= -0,99$. Par conséquent, l'événement *(client en attente)* est un événement influenceur du support car il implique la disparition de la règle. Dans le contexte de l'application bancaire, la banque a pour objectif l'augmentation des achats. Si le client n'est pas satisfait du prix d'un produit, et si l'événement *(client en attente)* apparaît, cela va impliquer la disparition de l'événement de l'achat du produit. Par conséquent, nous recommandons à la banque de ne pas faire attendre le client, sinon il/elle ne va pas acheter le produit.

5.5.2.2 Événements influenceurs de confiance

Dans cette section, nous nous intéressons à étudier plusieurs éléments en fonction du seuil d'influence sur la confiance $\theta_{Infl_{conf}}$ (voir définition 5.3.2) : le nombre de cas d'influence sur la

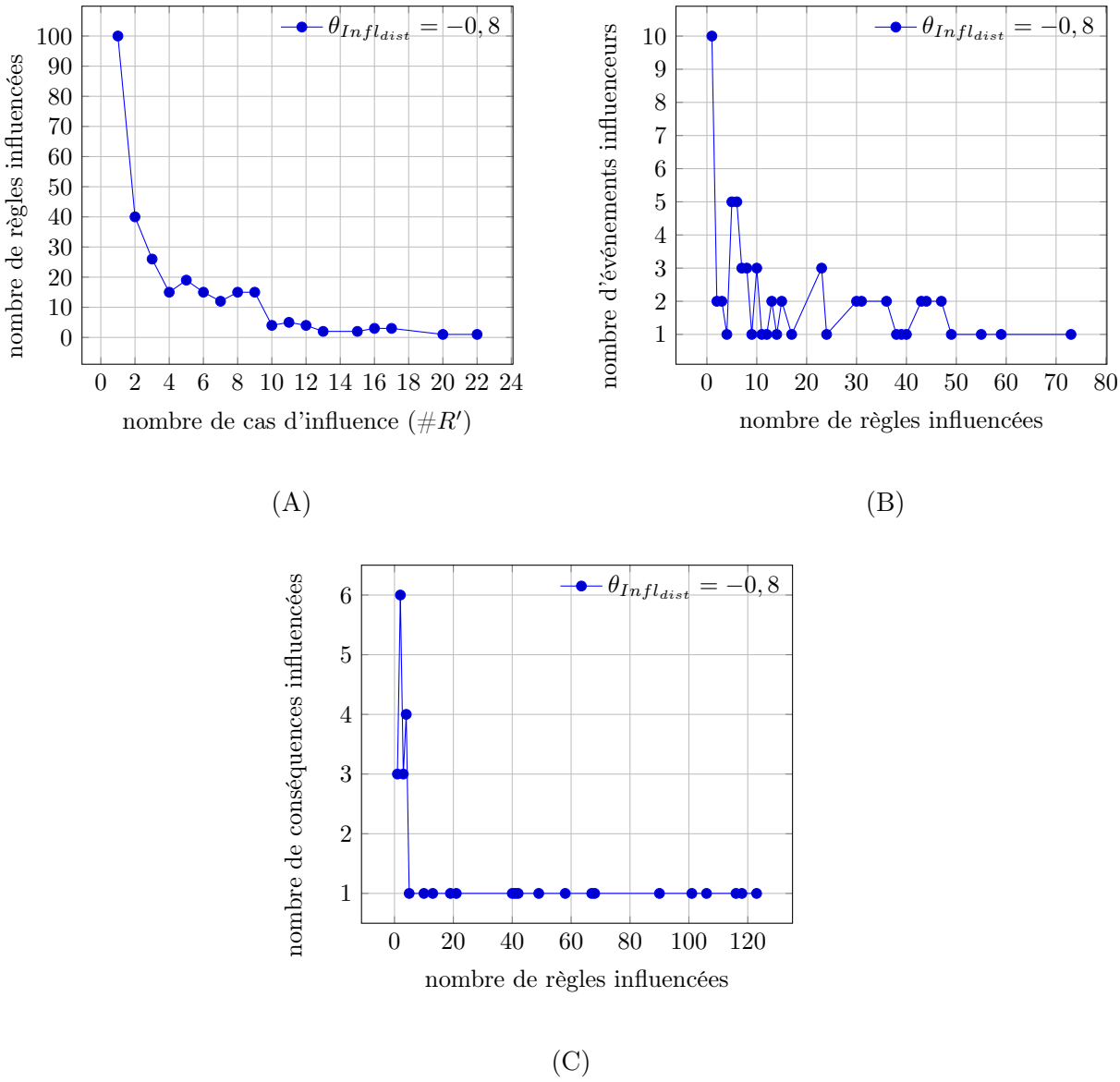


FIGURE 5.1 – Influence sur le support (la disparition) : (A) les règles influencées, (B) les événements influenceurs et (C) les conséquences influencées, pour $\theta_{Infl_{dist}} = -0,8$.

confiance, les règles influencées, les événements influenceurs, et les conséquences influencées, ce qui est présenté dans les figures 5.2 et 5.3.

Les cas d'influence : Dans la figure 5.2, nous montrons la distribution du nombre de cas d'influence sur la confiance ($\#R'$), en fonction des valeurs d'influence sur la confiance. Rappelons qu'une valeur négative d'influence sur la distance $Infl_{dist}$ représente les cas où l'événement ajouté diminue la confiance de la règle par rapport à sa confiance avant l'ajout de cet événement, et vice versa. Dans la figure 5.2, prenons l'exemple de la barre bleue associée avec $Infl_{conf} \in [0.2, 0.4[$ qui représente le nombre de règles R' pour lesquelles la confiance est augmentée de 20% à 40% par rapport à celle de R.
La distribution des cas d'influence sur la confiance ($\#R'$) peut être considérée comme relative-

ment normale : le nombre de règles R' lorsque $Infl_{conf} > 0$ est presque équivalent à celui lorsque $Infl_{conf} < 0$ et pour la plupart de règles R' : $Infl_{conf} \in [-0.2, 0.2[$.
Nous nous focalisons maintenant sur les événements qui influencent *significativement* la confiance de la règle. Une valeur du seuil d'influence sur la confiance doit être fixée. Rappelons qu'une valeur négative de $\theta_{Infl_{conf}}$ permet de détecter les événements influenceurs qui diminuent la confiance de la règle ($Infl_{conf} \leq \theta_{Infl_{conf}}$). De la même manière, une valeur positive de $\theta_{Infl_{conf}}$ permet de détecter les événements influenceurs qui augmentent la confiance de la règle ($Infl_{conf} \geq \theta_{Infl_{conf}}$). Nous fixons donc $\theta_{R_{conf}} = -0,7$ et $\theta_{R_{conf}} = 0,7$, ce qui est représenté dans la figure 5.3.

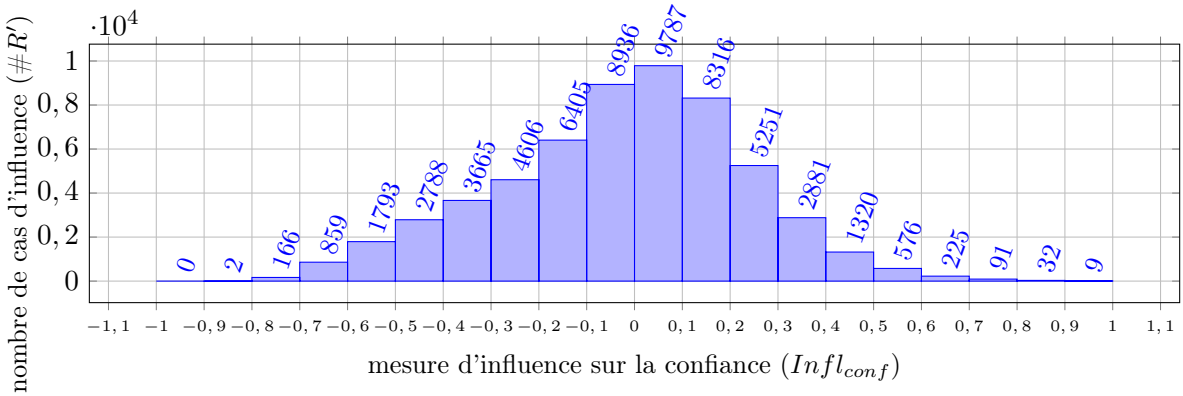


FIGURE 5.2 – Les cas d'influence sur la confiance ($\#R'$) en fonction de la mesure d'influence sur la confiance.

Lorsque $\theta_{\mathbf{Infl_{conf}}} = -0,7$: Les courbes bleues dans la figure 5.3 montrent en détail les 168 cas d'influence sur la confiance ($\#R'$) pour $\theta_{R_{conf}} = -0,7$, qui correspond à : 81 règles influencées distinctes, 18 événements influenceurs distincts et 24 conséquences influencées distinctes.
Dans la figure 5.3(A), parmi les 81 règles influencées R, 45(55%) sont influencées uniquement une seule fois chacune. Le reste de règles est influencé jusqu'à 8 fois chacune.
Dans la figure 5.3(B), nous montrons que les 18 événements influenceurs impactent, avec une distribution équitable, de 1 à 30 règles. Ce qui est similaire à la figure 5.3(C) qui montre que les conséquences influencées appartiennent chacune à 1 jusqu'à 25 règles, et qu'elles sont équitablement distribuées.

Lorsque $\theta_{\mathbf{R_{conf}}} = 0,7$: Les courbes rouges dans la figure 5.3 montrent en détail les 132 cas d'influence sur la confiance ($\#R'$) pour $\theta_{Infl_{conf}} = 0,7$. Ces cas correspondent à : 69 règles influencées distinctes, 88 événements influenceurs distincts et 42 conséquences influencées distinctes.
Les conclusions tirées de la courbe rouge dans figure 5.3(A) sont conformes à celles de la courbe bleue de la même figure (celle pour $\theta_{Infl_{conf}} = -0,7$).
Dans la figure 5.3(B), nous montrons que parmi les 88 événements influenceurs, 58 (65%) événements influencent uniquement une seule règle chacun, et qu'aucun événement n'influence plus de 4 règles différentes. Cette conclusion est contraire à celle tirée pour $\theta_{Infl_{conf}} = -0,7$. Notons également que le nombre d'événements influenceurs est 4 fois plus grand lorsque $\theta_{Infl_{conf}} = 0,7$, que lorsque $\theta_{Infl_{conf}} = -0,7$, tandis que le nombre de règles est comparable. Cela peut signifier que les événements qui font diminuer la confiance des règles sont plus importants : ils ne sont pas seulement peu nombreux mais également ils influencent plus de règles.
Dans la figure 5.3(C)) (courbe rouge), nous remarquons que les conséquences influencées appar-

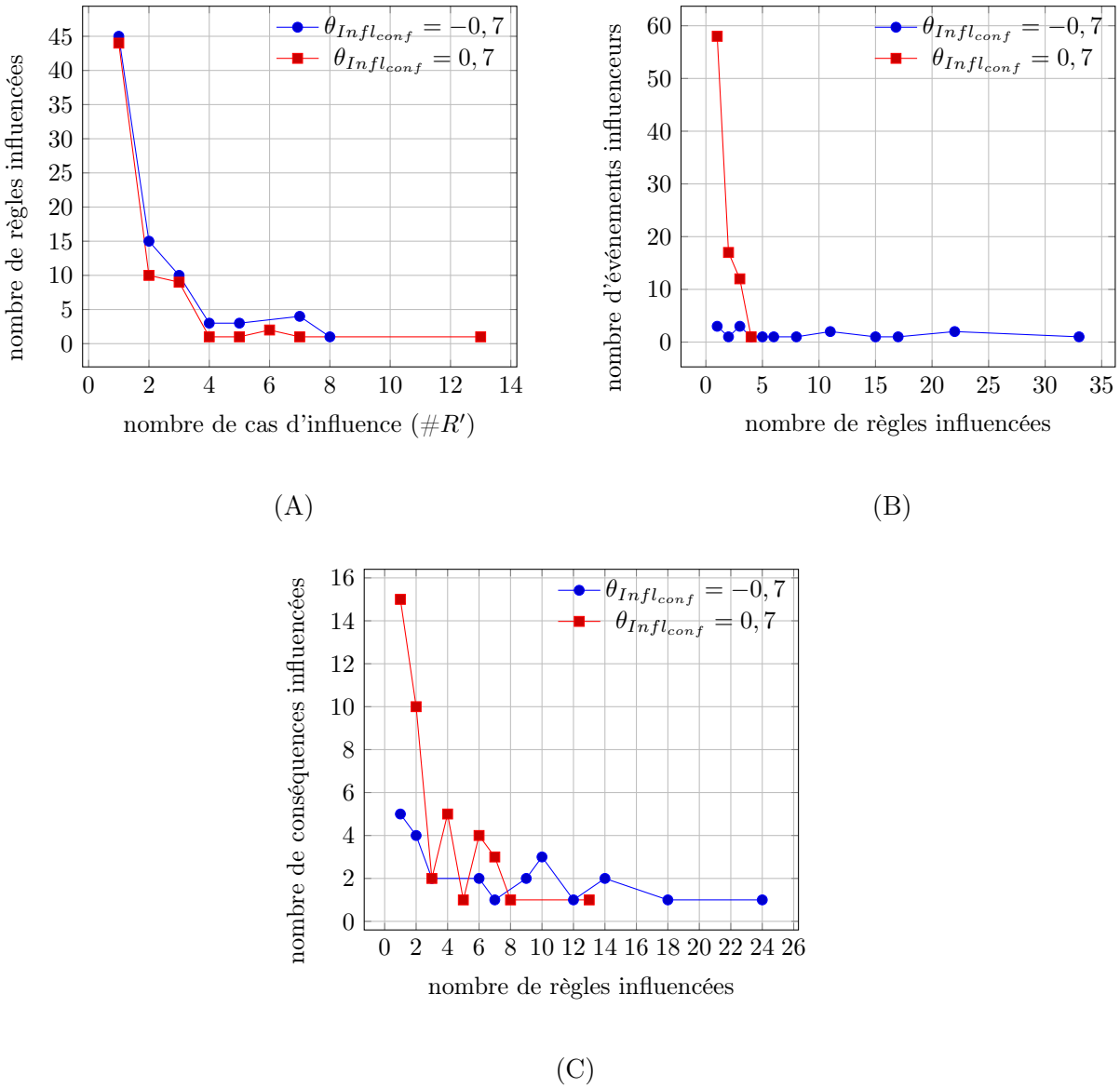


FIGURE 5.3 – Influence sur la confiance : (A) les règles influencées, (B) les événements influenceurs et (C) les conséquences influencées.

tiennent à 13 règles maximum et que la plupart de ces conséquences appartiennent à une seule règle chacun. Cette distribution est différente de celle lorsque $\theta_{Infl_{conf}} = -0,7$ (la courbe bleue de la même figure). De plus, le nombre des conséquences influencées est environ le double de celui pour $\theta_{Infl_{conf}} = -0,7$, tandis que le nombre de règles est comparable dans les deux cas. Encore une fois, cela peut signifier que les conséquences appartenant aux règles pour lesquelles la confiance diminue significativement sont plus importantes : elles ne sont pas seulement peu nombreuses, mais également elles appartiennent à plus de règles.
Nous pouvons conclure que, dans ce corpus, de nombreuses conséquences ne sont pas stables : elles sont sensibles et peuvent facilement être influencées par des événements ajoutées à l'antécédent de leur règles. Une attention particulière doit être accordée à ces conséquences afin de les influencer au bon moment, surtout lors de l'injection des événements à double influence.

Prenons maintenant un exemple concret d'un événement influenceur de la confiance. Soit la règle *R : (problème épargne)→ (contacter un concurrent)*. Pour l'événement influenceur *(indisponibilité d'un prêt à taux zéro)*, nous avons une influence sur la confiance $Infl_{conf}$*(R, (indisponibilité d'un prêt à taux zéro))*$= 0,75$. Cet événement augmente significativement la confiance de la règle et donc la probabilité que le client va contacter une entreprise concurrente (ce qui doit être évité). Une étude plus approfondie de ce cas montre que $Infl_{dist}$*(R, (indisponibilité d'un prêt à taux zéro))*$= -0,18$. Cela signifie que l'événement influenceur *(indisponibilité d'un prêt à taux zéro)* a une double influence : il ne fait pas uniquement augmenter la probabilité de l'occurrence de la conséquence, mais il la rapproche.

5.5.2.3 Événements influenceurs de distance

Dans cette section, nous nous focalisons sur les événements influenceurs de distance. Nous allons étudier quatre éléments (présentés dans les figures 5.4 et 5.5) : le nombre de cas d'influence sur la distance, les règles influencées, les événements influenceurs, et les conséquences influencées. Nous allons étudier ces cas en fonction de seuil de la mesure d'influence sur la distance $\theta_{Infl_{dist}}$ (voir définition 5.3.3).

Les cas d'influence : Nous présentons dans la figure 5.4 la distribution du nombre de cas d'influence, c'est-à-dire $\#R'$, en fonction de l'influence sur la distance ($Infl_{dist}$).
La barre bleue (dans la figure 5.4) associée aux valeurs $Infl_{dist} \in [-0.4, -0.2[$ (celui où le nombre de règles R est 131) représente le nombre de règles R' où un événement fait rapprocher la conséquence de 20% à 40% par rapport à sa distance avant d'ajouter l'événement, ce qui représente une influence significative. Rappelons qu'une valeur négative de l'influence sur la distance $Infl_{dist}$ représente les cas où l'événement ajouté fait rapprocher la conséquence vers l'antécédent, et vice versa. La barre associée aux valeurs de $Infl_{dist} \in [2, 2.2[$ (où le nombre de règles R est 228) représente le nombre de cas où la conséquence a été éloignée de 200% à 220% par rapport à sa distance avant d'ajouter l'événement.
La distribution des cas d'influence sur la confiance ($\#R'$) est différente de celle dans la section précédente (pour les cas d'influence sur la confiance). Nous pouvons remarquer que l'influence sur la distance varie entre $-0,4$ et $2,8$ et que la plupart de règles R' sont associées avec $Infl_{dist} \in [0,2, 0,8[$.
Sans grosse surprise, le nombre de règles R' avec $Infl_{dist} < 0$ est assez petit. En effet, le fait d'avoir une conséquence qui s'est rapprochée veut dire que le nombre d'occurrences de la règle R (qui ne contient pas l'événement influenceur et pour laquelle la conséquence était plus éloignée) est plus petit que le nombre d'occurrences de la règles R' (qui contient l'événement influenceur et dans laquelle la conséquence est plus proche). Ce cas est peu probable, d'où le petit nombre de règles R' avec $Infl_{dist} < 0$.
Afin d'étudier les cas où l'événement influenceur impacte *significativement* la distance de la conséquence, un seuil d'influence sur la distance $\theta_{Infl_{dist}}$ doit être fixé. Rappelons qu'une valeur négative de $\theta_{Infl_{dist}}$ permet de détecter les événements influenceurs qui font rapprocher la conséquence ($Infl_{dist} \leq \theta_{Infl_{dist}}$). De la même manière, une valeur positive de $\theta_{Infl_{dist}}$ permet de détecter les événements influenceurs qui font éloigner la conséquence ($Infl_{dist} \geq \theta_{Infl_{dist}}$).
Dans cette expérimentation, nous choisissons de fixer $\theta_{Infl_{dist}} = -0,2$ et $\theta_{Infl_{dist}} = 2$, ce qui est présenté dans la figure 5.5 et que nous allons étudier maintenant.

Lorsque $\theta_{\mathbf{Infl_{dist}}} = -0,2$: les courbes bleues dans la figure 5.5, correspondent aux cas où $\theta_{Infl_{dist}} = -0,2$. 132 cas d'influence sont présentes (les règles R' où $Infl_{dist} \leq -0,2$), ce qui

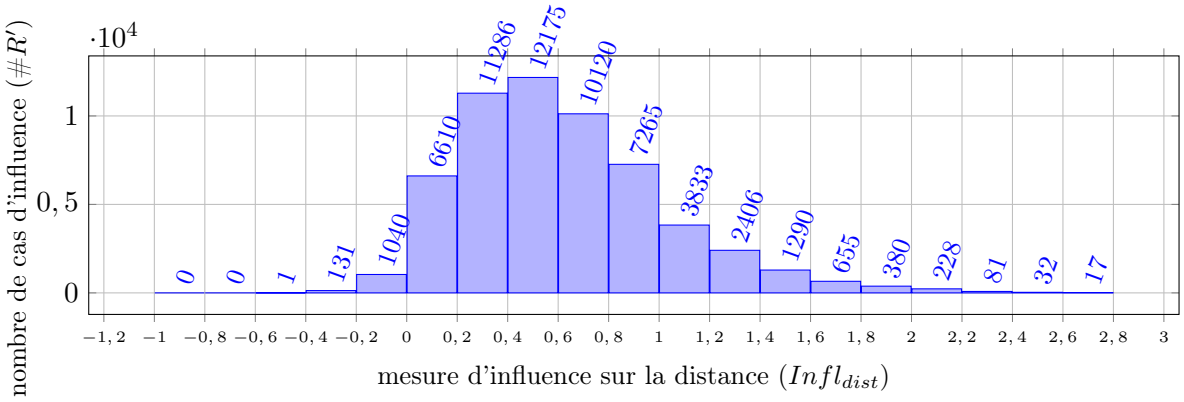
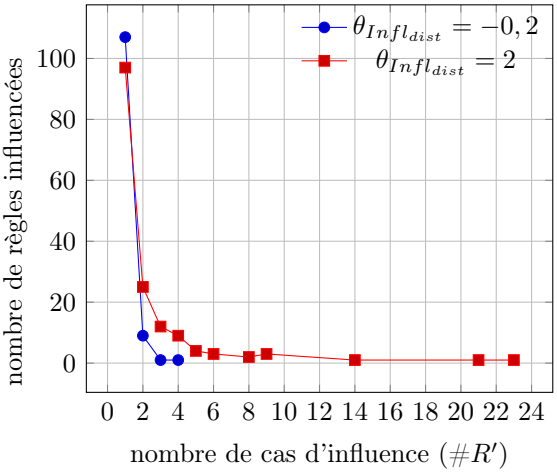


FIGURE 5.4 – Les cas d'influence sur la distance ($\#R'$) en fonction de la mesure d'influence sur la distance.

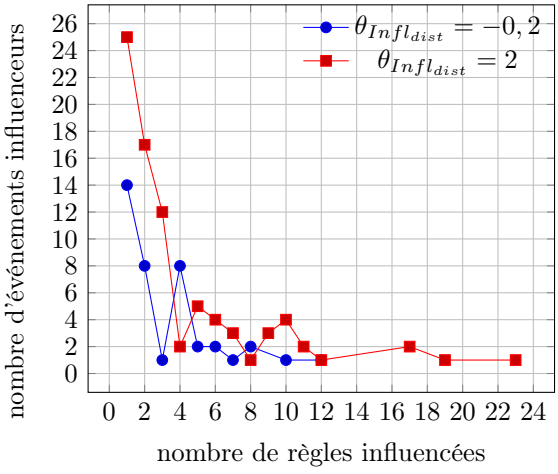
correspond à : 118 règles influencées R distinctes, 40 événements influenceurs distincts et 61 conséquences influencées distinctes.
Il est important de noter qu'il est préférable que ce seuil soit fixé par un expert du domaine, comme chaque valeur de seuil nécessite une analyse applicative particulière.
Dans la figure 5.5(A), nous présentons le nombre de fois qu'une règle est influencée. Nous remarquons que la plupart des 118 règles R sont influencées une seule fois chacune, et donc par un seul événement.
La figure 5.5(B) représente le nombre de règles qu'un événement influence. Remarquons que 14 événements (25%) influencent uniquement une règle chacun, et que certains événements influencent jusqu'à 12 règles chacun.
Il est important de mentionner qu'un événement qui influence plusieurs règles doit être utilisé avec précaution : il peut y avoir une influence significative sur l'occurence de plusieurs conséquences à la fois, et donc un effet non désirable peut se produire.
Dans la figure 5.5(C), nous présentons le nombre de fois qu'une conséquence est influencée. Nous remarquons que la plupart de conséquences appartiennent à moins de 5 règles (c'est-à-dire qu'elles sont influencées moins de 5 fois dans des règles distinctes), plus de 50% des conséquences appartiennent à plusieurs règles. Ce qui veut dire qu'elle sont influencées par plusieurs événements dans plusieurs règles. Ces conséquences sont très sensibles car elles se font facilement rapprocher du préfixe de la règle.

Lorsque $\theta_{\mathbf{Infl_{dist}}} = \mathbf{2}$: les courbes rouges dans la figure 5.5, correspondent aux cas où $\theta_{Infl_{dist}} = 2$. Notons que 358 cas d'influence sont obtenus (où $Infl_{dist} \geq 2$), ce qui correspond à : 158 règles influencées distinctes, 82 événements influenceurs distincts et 50 conséquences influencées distinctes.
Les observations effectuée dans la figure 5.5(A) et 5.5(B) où $\theta_{Infl_{dist}} = -0.2$ sont en accord avec celles où $\theta_{Infl_{dist}} = 2$.
La figure 5.5(C) montre le nombre de fois qu'une conséquence est influencée. Un nombre significatif de conséquences appartenant chacune à plus de 7 règles est obtenu, et donc ces conséquences sont influencées dans plus de 7 contextes différents. Contrairement au cas de $\theta_{Infl_{dist}} = -0.2$ (courbe bleue), dans lequel la plupart de conséquences appartiennent chacune à moins de 7 règles. Nous concluons qu'à partir de l'ensemble d'événements influenceurs détectés, certains d'entre eux influencent significativement la distance de la conséquence dans plusieurs règles. Nous considérons

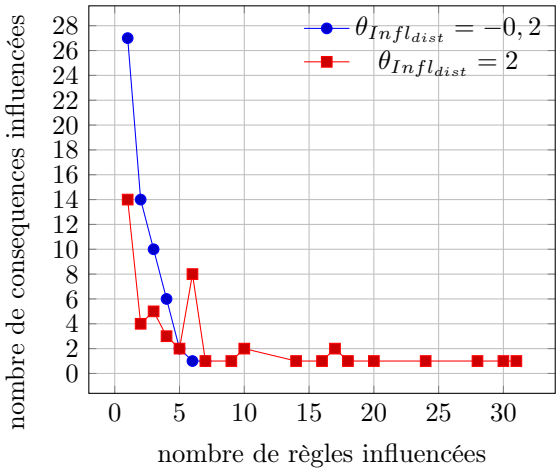
que ces événements sont plus utiles que les autres, en raison de leur grand impact. De manière similaire, certaines conséquences et règles sont facilement influencées par plusieurs événements. Elles doivent être considérées attentivement comme elles vont être souvent impactées. Cependant, dans le corpus utilisé, plusieurs conséquences ne sont jamais influencées, comme par exemple l'événement *(crise économique, négatif)*, ce qui est évident dans ce contexte.



(A)



(B)



(C)

FIGURE 5.5 – Influence sur la distance : (A) les règles influencées, (B) les événements influenceurs et (C) les conséquences influencées.

Prenons maintenant un exemple concret. Soit la règle R : *(souscription)* $\rightarrow$ *(acheter un produit)* avec $Infl_{dist}(R, (client\ en\ attente)) = 2,5$. Cela veut dire que l'événement influenceur *(client en attente)* fait éloigner de $2,5$ fois la conséquence. Une analyse plus approfondie montre que la différence concrète de distance est égale à 44 instants, ce qui signifie que lorsque l'événement *(client en attente)* apparaît après l'apparition de l'antécédent de R, la conséquence de R s'éloigne de 44 instants du préfixe, ce qui est assez élevé. Dans le contexte de l'application bancaire, cela

veut dire que lorsque la banque fait attendre le client, celui-ci va acheter un produit plus tard. Comme nous l'avons mentionné, nous nous intéressons à la détection des conséquences qui sont influencées plusieurs fois et aux événements qui influencent plusieurs règles à la fois. Dans le tableau 5.4, nous présentons quelques exemples de conséquences influencées et d'événements influenceurs les plus intéressants détectés dans le corpus et liés aux trois cas d'influence : l'influence sur le support, sur la confiance et sur la distance.

TABLE 5.4 – Un exemple concret de conséquences influencées et d'événements influenceurs.

	Influence sur le support pour $\theta_{Infl_{supp}} = -0,8$	Influence sur la confiance pour $\theta_{Infl_{conf}} = 0,7$	Influence sur la distance pour $\theta_{Infl_{dist}} = 0,2$
Conséquence influencée	problème agence pas de vente souscription	problème chéquier contacter la concurrence achat produit	ouvrir compte épargne achat produit plan épargne logement
Événement influenceur	pas de proposition prix élevé condition strict de contrat	coût élevé client en attente plan épargne logement	coût élevé client en attente prêt à taux élevé

5.5.2.4 Discussion

Les études expérimentales menées dans cette section ont montré la performance de l'algorithme IE en terme de temps d'exécution. Nous considérons que cette performance est valable indépendamment du corpus étudié car elle dépend uniquement de choix d'algorithme de fouille de règles dans lequel IE est intégré (voir section 5.4.3). Cependant, en terme de qualité et de nombre d'événements influenceurs, le degré d'influence sur des événements varie forcément d'un corpus à un autre. Par conséquent, nous proposons comme perspective de lancer ces expérimentations sur d'autres corpus d'une nature différente afin de donner une conclusion générique concernant l'efficacité de notre algorithme.
De plus, il est important de mentioner que malgré l'utilisation de seuils pour déterminer les événements influenceurs les plus significatifs, une analyse par un expert du domaine est requise afin d'affiner le résultat final, car l'utilité de ces événements est dépendante du domaine d'application.

5.6 Synthèse

Dans ce chapitre, nous avions pour but de pouvoir influencer l'apparition de futurs événements dans une séquence d'événements. Nous avons introduit un nouveau concept : les événements influenceurs, et montré que ces événements ont un rôle différent des événements d'une grande utilité proposés dans l'état de l'art.
L'algorithme IE que nous avons proposé détecte trois types d'événements influenceurs : les événements influenceurs du support, les événements influenceurs de la confiance de règles et les événements influenceurs de la distance entre le préfixe et la conséquence de règles. Chacun de ces événements est associé avec une nouvelle mesure d'influence. Nous avons précisé qu'il est préférable que les seuils d'influence soient fixés par un expert du domaine.
Nous avons également proposé une approche pour intégrer l'algorithme IE dans un algorithme traditionnel de fouille de règles d'épisode et dans notre algorithme $DEER$ dans le but d'effectuer une prédiction au plus tôt. Nous avons montré la différence de performance entre l'intégration de ces deux algorithmes.

Les expérimentations menées ont montré l'efficacité de IE pour la détection des événements influenceurs, et surtout pour la détection de double influence portée par certains événements et la facilité d'influencer certaines règles et conséquences. Nous avons insisté sur le fait que ces derniers cas doivent être considérés avec précaution puisque plusieurs événements peuvent influencer ou être influencés plusieurs fois.
Le travail présenté dans ce chapitre représente la dernière contribution apportée dans ce manuscrit. Plusieurs perspectives et travaux futurs sont envisagés autour du problème de données complexes, de prédiction et d'influence au plus tôt, que nous allons présenter dans le chapitre suivant.

Chapitre 6

Conclusion et perspectives

Sommaire

6.1 Résumé de contributions 126

6.1.1 La prédiction au plus tôt des événements futurs dans une séquence d'événements 126

6.1.2 La détection au plus tôt de l'émergence de nouvelles règles d'épisode dans un flux d'événements 126

6.1.3 La détection des événements "influenceurs" des événements futurs dans une séquence d'événements 127

6.2 Perspectives 128

Dans cette thèse, nous nous intéressons aux données apparues depuis plusieurs années et désormais disponibles, que nous qualifions de "données complexes". Les données complexes sont très volumineuses, véloces et variées, ces trois caractéristiques sont traditionnellement représentées par la règle des "3V".
Depuis de nombreuses années, les modèles prédictifs de fouille de données ont été dédiés et validés sur des données dites simples (provenant d'une source unique, avec une structure nette, etc.). Cependant, la généralisation et la prédominance de données complexes amènent à de nouveaux défis principalement liés à la prise en compte des caractéristiques spécifiques de ces données en modélisation prédictive. De plus, les données complexes sont très riches en informations cachées, ce qui augmente leur pouvoir prédictif. Par conséquent, proposer un modèle prédictif adapté aux données complexes est devenu incontournable pour mieux prédire voire influencer le futur. Parce que prédire au plus tôt l'occurrence d'un événement permet de réagir tôt, d'influencer l'apparition des événements futurs et donc de diminuer au maximum le risque associé, si nécessaire. C'est la raison pour laquelle nous nous intéressons à ce sujet.
Les séquences complexes d'événements représentent le type de données complexes qui nous intéresse plus particulièrement. La modélisation prédictive dans une séquence d'événements est ardue, en raison de la nature de séquence d'événements (elle peut être infinie, la zone d'impact d'un événement n'est pas limitée, etc.). Cependant, la modélisation dédiée à la prédiction au plus tôt et à l'influence sur des événements futurs dans une séquence complexes d'événements comporte plusieurs défis, auxquels s'ajoute la difficulté d'une séquence complexe.
L'objectif de ce chapitre est de donner un aperçu global de différents défis auxquels nous avons fait face et les contributions apportées. Nous concluons ce chapitre en présentant les perpectives de nos travaux et les questions de recherche qui restent ouvertes.

6.1 Résumé de contributions

Dans cette thèse, nous nous intéressons à la problématique suivante : “comment prédire au plus tôt et influencer l'apparition des événements futurs dans une séquence complexe d'événements ?”. Pour répondre à cette question, nous avons apporté plusieurs contributions qui sont résumées dans les sous-sections suivantes.

6.1.1 La prédiction au plus tôt des événements futurs dans une séquence d'événements

La contribution principale réside dans la maîtrise de l'horizon d'apparition des événements futurs. Dans la littérature, les événements futurs sont prédits par extraction de règles d'épisodes dans une séquence d'événements. Lorsque ces règles sont exploitées, elles prédisent l'occurrence d'un événement proche qui apparaît peu de temps après qu'il ait été prédit en particulier en raison de la vélocité de la séquence. Par conséquent, l'événement prédit apparaît avant que l'on puisse avoir le temps de réagir. Pour cette raison, la prédiction au plus tôt d'événements est de grand intérêt.
Deux défis se présentent pour atteindre l'objectif de la prédiction au plus tôt par règles d'épisodes : (i) la fouille de règles d'épisode avec une conséquence distante, (ii) la fouille de règles avec un antécédent le plus petit possible. Nous avons montré, dans la partie dédiée à cette contribution, que les algorithmes traditionnels ne sont pas adaptés à relever ces défis au cours du processus de fouille, mais uniquement durant une étape de post-traitement.
Nous avons proposé l'algorithme $DEER$ (Distant and Essential Episode Rules) qui comporte trois points originaux : (i) l'absence d'une étape d'extraction des épisodes fréquents, ce qui permet d'extraire les règles directement ; (ii) l'identification de la conséquence très tôt dans le processus de la fouille, ce qui permet d'imposer une distance entre l'antécédent (le déclencheur) et la conséquence (l'événement futur), et donc de préciser l'horizon d'apparition de ce dernier événement. Cette manière de construire des règles n'a jamais été proposée dans la littérature ; (iii) la construction de règles dites essentielles, c'est-dire avec un antécédent le plus petit possible en nombre d'événements et en durée : ce qui permet d'effectuer une prédiction au plus tôt en utilisant ces règles.
Nous avons mené une étude comparative théorique entre $DEER$ et les algorithmes de l'état de l'art. Cette étude a montré que notre algorithme est plus rapide qu'un algorithme traditionnel. Cela a été également prouvé lors d'une étude expérimentale menée sur un corpus de données réelles : $DEER$ est 4 fois plus rapide que l'algorithme traditionnel $MINEPI$ de l'état de l'art. Nous avons montré également que $DEER$ est beaucoup plus performant dans une tâche de prédiction (en précision et en rappel). Cette étude expérimentale a révélé l'existence d'une dépendance temporelle entre les événements dans le corpus utilisé.

6.1.2 La détection au plus tôt de l'émergence de nouvelles règles d'épisode dans un flux d'événements

Dans cette partie, nous nous sommes intéressés à la détection de l'émergence afin d'améliorer la prédiction dans un flux d'événements qui représente une séquence infinie. Traditionnellement, un modèle de prédiction utilise les règles extraites auparavant pour effectuer cette prédiction. Cependant, en raison de la vélocité et de la variété du flux d'événements, des nouvelles règles peuvent apparaître, mais elles ne sont pas intégrées dans le modèle de prédiction. D'autre part, pour pouvoir intégrer rapidement ces règles, la solution réside dans la détection de l'émergence

de leur apparition.
Pour répondre à ce défi, nous avons proposé l'algorithme EER (Emergent Episode Rules) qui se base sur l'hypothèse que les nouvelles règles ne sont pas complètement indépendantes, elles sont forcément liées ou influencées par d'autres règles déjà connues. Pour cela, nous considérons que les nouvelles règles qui tendent à apparaître et qui sont *similaires* à des règles connues, vont émerger dans le futur. Ainsi, EER détecte l'émergence uniquement des règles similaires aux règles déjà extraites (que nous avons appelées : règles de référence). Nous avons choisi d'utiliser la distance d'édition comme mesure de similarité. L'algorithme EER s'exécute sur une plateforme de traitement distribué : $STORM$. Nous avons proposé une *topologie* pour $STORM$ composée de *spouts* et de deux niveaux de *bolts*. Nous avons proposé une phase de détection d'émergence basée dans un premier temps sur la détection de l'émergence préliminaire et dans un deuxième temps sur la détection de l'émergence finale, par la proposition d'une mesure du taux de croissance que nous avons appelée une mesure de taux de croissance finale.
Une étude expérimentale a été menée afin d'évaluer la performance de l'algorithme EER sur un flux d'événements composé de données réelles. Nous avons montré que la performance d'EER est assez satisfaisante en raison du corpus utilisé : lorsque les occurrences d'une règle sont très espacées, l'algorithme considère que le support de la règle est stable et donc décide la non émergence de la règle. Cette étude a prouvé également l'importance de la prise en compte de la similarité, ce qui augmente la performance de l'algorithme.

6.1.3 La détection des événements "influenceurs" des événements futurs dans une séquence d'événements

La contribution apportée dans cette partie réside dans l'étude de l'influence des événements, plus précisément leur impact sur d'autres événements. De manière générale, des événements sont manuellement choisis et injectés dans la séquence de façon à impacter l'occurrence de l'événement prédit. Nous avons choisi d'automatiser cette tâche par la proposition d'un algorithme de détection des événements que nous appelons "influenceurs" dans une séquence d'événements.
En effet, très peu de travaux de la littérature s'intéressent à influencer des événements futurs. Nous avons montré que le seul lien que nous pouvons trouver entre la fouille de données et influencer des événements réside dans l'extraction des motifs dits d'une grande utilité.
Par conséquent, nous introduisons un nouveau concept : les événements influenceurs et montrons que ces événements ont un rôle différent des événements d'une grande utilité proposés dans l'état de l'art.
L'algorithme IE (Influencer Events) que nous avons proposé détecte trois types d'événements influenceurs : les événements influenceurs du support, les événements influenceurs de la confiance des règles et les événements influenceurs de la distance entre le préfixe et la conséquence de règles. Chacun de ces événements est associé avec une nouvelle mesure d'influence. Nous avons précisé qu'il est préférable que les seuils d'influence soient fixés par un expert du domaine.
Nous avons également proposé une approche pour intégrer l'algorithme IE dans un algorithme traditionnel de fouille de règles d'épisode et dans notre algorithme $DEER$ dans le but d'effectuer une prédiction au plus tôt. Nous avons montré la différence de performance entre l'intégration dans ces deux algorithmes.
Les expérimentations menées ont montré l'efficacité de IE pour la détection des événements influenceurs, et surtout pour la détection de double influence portée par certains événements et la facilité d'influencer certaines règles et conséquences. Nous avons insisté sur le fait que ces derniers cas doivent être considérés avec précaution car plusieurs événements influencent ou sont influencés plusieurs fois.

6.2 Perspectives

Les contributions apportées dans ce manuscrit ont été validées et les algorithmes proposés ont été comparés avec des algorithmes de la littérature. Cependant, comme nous l'avons mentionné, les expertises de domaine sont importantes dans le contexte de nos travaux, et il est préférable de laisser aux experts la possibilité de fixer les différents seuils et paramètres nécessaires pour nos algorithmes en fonction de leur besoins. Étant donné que cette thèse est financée par le Groupe Crédit Agricole S.A., il est intéressant de voir comment leurs experts adaptent et appliquent nos algorithmes dans le contexte bancaire.
Dans le cadre de nos perspectives de recherche à moyen et à long terme, nous envisageons dans un premier temps d'intégrer un nouvelle dimension de données dans nos modèles. Étant donné que la plupart des sources de données complexes est issue du Web, des liens particuliers peuvent être cachés entre les données. Prenons par exemple les liens spatiaux entre les messages utilisateurs ou les tweets. Ces liens sont crées en fonction de l'emplacement géographique de l'utilisateur au moment de l'écriture de son message ou tweet. Parmi les défis de fouille de données dans Twitter, nous pouvons mentionner par exemple la prédiction de l'apparition d'un tweet d'un artiste aux État-Unis, lorsque un tweet est tweeté par un autre artiste en France. Nous nous intéressons à **intégrer les liens spatiaux** dans les modèles de prédiction : ces liens doivent impacter la nature des associations extraites et la manière de l'utilisation de ces associations dans une tâche de prédiction.
Nous envisageons dans un deuxième temps d'approfondir nos études dans le traitement de flux d'événements, plus particulièrement dans la détection de nouvelles règles émergentes. En effet, les règles que nous détectons comme émergentes sont intégrées dans le modèle de prédiction et sont donc utilisées pour prédire les événements futurs. Cependant, cette prédiction n'est pas fiable car les règles émergentes ne sont pas encore fréquentes, et il est impossible de faire confiance à une association tant qu'elle n'est pas assez fréquente. Nous souhaitons étudier la fiabilité dans une tâche de prédiction pour les associations émergentes. Nous envisageons de proposer une approche pour **estimer la confiance des associations dans un flux d'événements**, qui pourrait être inspirée de l'estimation de la fréquence des associations et donc la détection de leur émergence.
D'autre part, nous souhaitons intégrer notre contribution sur la détection des événements influenceurs dans le cadre de règles émergentes. En effet, lorsqu'une règle est détectée comme émergente, il est intéressant de détecter quels événements nous pouvons injecter dans le flux d'événements afin d'influencer cette émergence : accélérer l'émergence ou bien l'annuler complètement en fonction du risque porté par la règle par exemple. Nous envisageons donc de proposer une approche pour la **détection des événements influenceurs des règles émergentes**.
Dans ce manuscrit nous avons traité des données complexes qui ont trois caractéristiques représentées par la règle de 3V : volumineuses, variables et véloces. Cependant, nous pouvons constater que les données peuvent perdre de leur intérêt au fil du temps : elles deviennent peu importantes, et impactent de moins en moins les autres données, voire elles n'apparaissent plus, ce qui impacte la fiabilité de la détection de l'émergence (car les règles que nous avons détectées comme émergentes, n'ont pas réellement émergé). Par conséquent, nous pouvons considérer que les données s'évanouissent au fil du temps, ce qui doit être intégré dans les modèles de prédiction et d'influence. Nous nous intéressons à étudier les données complexes en ajoutant une quatrième caractéristique de données : **la volatilité de données** à la règle de 3V. Cette étude sera plus intéressante dans le cadre de traitement de flux de données, où la volatilité de données est clairement constatée, ce qui, à notre connaissance, n'a jamais été étudié dans la littérature.
Pour faire une synthèse des apports de ces deux dernières perspectives, nous pouvons conclure que lorsqu'un règle est détectée comme émergente, mais qu'elle n'émerge pas réellement, cela

peut être dû à deux raisons : (i) un événement influenceur a été injecté dans le contexte de la règle qui a fait que la règle n'a pas émergée, (ii) la volatilité de données fait que les événements qui composent la règle ne sont plus vrais et donc la règle n'existe plus. Cela nous permet d'avoir une influence et une compréhension meilleure des événements futurs.

Bibliographie

Abdi, M. J. and Giveki, D. (2013). Automatic detection of erythemato-squamous diseases using pso–svm based on association rules. *Engineering Applications of Artificial Intelligence*, 26(1) :603–608.

Achar, A., Laxman, S., Viswanathan, R., and Sastry, P. (2012). Discovering injective episodes with general partial orders. *Data Mining and Knowledge Discovery*, 25(1) :67–108.

Achar, A., Sastry, P., et al. (2013). Pattern-growth based frequent serial episode discovery. *Data & Knowledge Engineering*, 87 :91–108.

Agarwal, R. C., Aggarwal, C. C., and Prasad, V. (1999). *Depth First Generation of Large Itemsets for Association Rules*. IBM Thomas J. Watson Research Division.

Agarwal, R. C., Aggarwal, C. C., and Prasad, V. (2000). Depth first generation of long patterns. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 108–118. ACM.

Aggarwal, C. (2015). *Data Mining : The Textbook*. Springer.

Aggarwal, C. C. (2007). *Data streams : models and algorithms*, volume 31. Springer Science & Business Media.

Aggarwal, C. C. and Han, J. (2014). *Frequent pattern mining*. Springer.

Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. (2003). A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 81–92. VLDB Endowment.

Aggarwal, C. C. and Subbian, K. (2012). Event detection in social streams. In *SDM*, volume 12, pages 624–635. SIAM.

Aggarwal, C. C. and Yu, P. S. (1998). Mining large itemsets for association rules. *IEEE Data Eng. Bull.*, 21(1) :23–31.

Aggarwal, C. C. and Zhai, C. (2012). *Mining text data*. Springer Science & Business Media.

Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM.

Agrawal, R., Srikant, R., et al. (1994). Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, volume 1215, pages 487–499.

Ahmed, C. F., Tanbeer, S. K., and Jeong, B.-S. (2010). Mining high utility web access sequences in dynamic web log data. In *Software Engineering Artificial Intelligence Networking and Parallel/Distributed Computing (SNPD), 2010 11th ACIS International Conference on*, pages 76–81. IEEE.

Ahmed, C. F., Tanbeer, S. K., Jeong, B.-S., and Lee, Y.-K. (2009). Efficient tree structures for high utility pattern mining in incremental databases. *Knowledge and Data Engineering, IEEE Transactions on*, 21(12) :1708–1721.

Ahmed, C. F., Tanbeer, S. K., Jeong, B.-S., and Lee, Y.-K. (2011). Huc-prune : an efficient candidate pruning technique to mine high utility patterns. *Applied Intelligence*, 34(2) :181–198.

Ahuja, A. and Chakka, R. (2014). Analysis of the performance-improvement due to a new receiver for baseband telecommunications, and development of a fast predictive model. *Journal of Scientific & Industrial Research*, 73 :577–587.

Alvanaki, F., Sebastian, M., Ramamritham, K., and Weikum, G. (2011). Enblogue : emergent topic detection in web 2.0 streams. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 1271–1274. ACM.

Andreeva, A., Howorth, D., Chothia, C., Kulesha, E., and Murzin, A. G. (2014). Scop2 prototype : a new approach to protein structure mining. *Nucleic acids research*, 42(D1) :D310–D314.

Aniello, L., Baldoni, R., and Querzoni, L. (2013). Adaptive online scheduling in storm. In *Proceedings of the 7th ACM international conference on Distributed event-based systems*, pages 207–218. ACM.

Ao, X., Luo, P., Li, C., Zhuang, F., and He, Q. (2015). Online frequent episode mining. In *The int ?l conf. on data engineering. Seoul, Korea*.

Apache Hadoop (2005). `http://mahout.apache.org`.

Apache Mahout (2011). `http://storm.apache.org/index.html`.

Apache Storm (2012). `http://storm.apache.org/index.html`.

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

Beck, F., Burch, M., Diehl, S., and Weiskopf, D. (2014). The state of the art in visualizing dynamic graphs. *EuroVis STAR*.

Bellman, R., Holland, J., and Kalaba, R. (1959). On an application of dynamic programming to the synthesis of logical systems. *Journal of the ACM (JACM)*, 6(4) :486–493.

Benevenuto, F., Magno, G., Rodrigues, T., and Almeida, V. (2010). Detecting spammers on twitter. In *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)*, volume 6, page 12.

Bifet, A. (2013). Mining big data in real time. *Informatica*, 37(1).

Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. (2010). Moa : Massive online analysis. *The Journal of Machine Learning Research*, 11 :1601–1604.

Bifet, A. and Morales, G. D. F. (2014). Big data stream learning with samoa. In *2014 IEEE International Conference on Data Mining Workshop*, pages 1199–1202. IEEE.

Bilbrey Jr, J. K., Riley, N. F., and Sams, C. L. (2013). Short-term prediction of exchange traded funds (etfs) using logistic regression generated client risk profiles. *Journal of Finance and Accountancy*, 14 :1.

Bogle, S. and Potter, W. (2015). A machine learning predictive model for the jamaica frontier market. In *Proceedings of the 2015 Int'l Conference of Data Mining and Knowledge Engineering*.

Bu, Y., Howe, B., Balazinska, M., and Ernst, M. D. (2010). Haloop : efficient iterative data processing on large clusters. *Proceedings of the VLDB Endowment*, 3(1-2) :285–296.

Budak, C., Agrawal, D., and El Abbadi, A. (2011). Structural trend analysis for online social networks. *Proceedings of the VLDB Endowment*, 4(10) :646–656.

Cai, C. H., Fu, A. W., Cheng, C., and Kwong, W. (1998). Mining association rules with weighted items. In *Database Engineering and Applications Symposium, 1998. Proceedings. IDEAS'98. International*, pages 68–77. IEEE.

Calì, A., Calvanese, D., De Giacomo, G., and Lenzerini, M. (2013). Data integration under integrity constraints. In *Seminal Contributions to Information Systems Engineering*, pages 335–352. Springer.

Cataldi, M., Di Caro, L., and Schifanella, C. (2010). Emerging topic detection on twitter based on temporal and social terms evaluation. In *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, page 4. ACM.

Chan, R. C., Yang, Q., and Shen, Y.-D. (2003). Mining high utility itemsets. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 19–26. IEEE.

Chi, Y., Wang, H., Yu, P. S., and Muntz, R. R. (2004). Moment : Maintaining closed frequent itemsets over a stream sliding window. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*, pages 59–66. IEEE.

Cho, C., Zheng, Y., and Chen, A. (2007). Continuously matching episode rules for predicting future events over event streams. In *Adv. in Data and Web Man.* Springer.

Cho, C.-W., Zheng, Y., Wu, Y.-H., and Chen, A. L. (2008). A tree-based approach for event prediction using episode rules over event streams. In *Database and Expert Systems Applications*, pages 225–240. Springer.

Choi, J., Earley, B., and Mungin, L. (2015). Predicting customer address changes using transaction behavior patterns. In *Systems and Information Engineering Design Symposium (SIEDS), 2015*, pages 273–277. IEEE.

Christen, U. and Busch, R. (2015). The art of control engineering : Science meets industrial reality. *Oil & Gas Science and Technology–Revue d ?IFP Energies nouvelles*, 70(1) :31–39.

Cormack, G. V. (2007). Email spam filtering : A systematic review. *Foundations and Trends in Information Retrieval*, 1(4) :335–455.

Cortes, C., Pregibon, D., and Volinsky, C. (2012). Computational methods for dynamic graphs. *Journal of Computational and Graphical Statistics.*

Csernel, B., Clerot, F., and Hébrail, G. (2005). Traitement des flux de données. *37emes Journées de Statistique (SFDS).*

Cugola, G. and Margara, A. (2012). Processing flows of information : From data stream to complex event processing. *ACM Computing Surveys (CSUR)*, 44(3) :15.

Cule, B., Tatti, N., and Goethals, B. (2013). Marbles : Mining association rules buried in long event sequences. *Statistical Analysis and Data Mining.*

Cumming, J. G., Davis, A. M., Muresan, S., Haeberlein, M., and Chen, H. (2013). Chemical predictive modelling to improve compound quality. *Nature reviews Drug discovery*, 12(12) :948–962.

Dai, H.-J., Chang, Y.-C., Tsai, R. T.-H., and Hsu, W.-L. (2010). New challenges for biological text-mining in the next decade. *Journal of computer science and technology*, 25(1) :169–179.

De Francisci Morales, G. (2013). Samoa : A platform for mining big data streams. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 777–778. International World Wide Web Conferences Steering Committee.

Dean, J. and Ghemawat, S. (2008). Mapreduce : simplified data processing on large clusters. *Communications of the ACM*, 51(1) :107–113.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

Desmier, E., Plantevit, M., Robardet, C., and Boulicaut, J.-F. (2013). Trend mining in dynamic attributed graphs. In *Machine Learning and Knowledge Discovery in Databases*, pages 654–669. Springer.

DETSKY, A. S. (1995). Clinical prediction models. *Acta Anaesthesiologica Scandinavica*, 39(s105) :134–135.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7) :1895–1923.

Dong, G. and Li, J. (1999). Efficient mining of emerging patterns : Discovering trends and differences. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 43–52. ACM.

Dong, G. and Pei, J. (2007). *Sequence data mining*, volume 33. Springer Science & Business Media.

Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. (1998). *Biological sequence analysis : probabilistic models of proteins and nucleic acids*. Cambridge university press.

Džeroski, S. (2010). *Relational data mining*. Springer.

Eirinaki, M. and Vazirgiannis, M. (2003). Web mining for web personalization. *ACM Transactions on Internet Technology (TOIT)*, 3(1) :1–27.

Erwig, M., Gu, R. H., Schneider, M., Vazirgiannis, M., et al. (1999). Spatio-temporal data types : An approach to modeling and querying moving objects in databases. *GeoInformatica*, 3(3) :269–296.

Fahed, L., Brun, A., and Boyer, A. (2014a). Episode rules mining algorithm for distant event prediction. In *KDIR 2014 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval, Rome, Italy, 21 - 24 October, 2014*, pages 5–13.

Fahed, L., Brun, A., and Boyer, A. (2014b). Extraction de règles d'épisodes minimales dans des séquences complexes. In *14èmes Journées Francophones Extraction et Gestion des Connaissances, EGC 2014, Rennes, France, 28-32 Janvier, 2014*, pages 545–548.

Fahed, L., Brun, A., and Boyer, A. (2014c). Prédiction au plus tôt d'événements par régles d'épisodes. In *Atelier Francophone : Fouille de données Spatiales et Temporelles (FST), Extraction et Gestion des Connaissances (EGC)*, pages 27–37.

Fahed, L., Brun, A., and Boyer, A. (2015a). Efficient discovery of episode rules with a minimal antecedent and a distant consequent. In *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, volume 553, pages 3–18. Springer.

Fahed, L., Brun, A., and Boyer, A. (2015b). Influencer events in episode rules : A way to impact the occurrence of events. *19th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, Procedia Computer Science*, 60 :527–536.

Fairon, C., Klein, J. R., and Paumier, S. (2007). *Le langage SMS : étude d'un corpus informatisé à partir de l'enquête «Faites don de vos SMS à la science»*. Presses univ. de Louvain.

Faith, P., Siegel, K. P., and Hammad, A. (2013). Pre-authorization of a transaction using predictive modeling. US Patent 8,352,315.

Fan, W. and Bifet, A. (2013). Mining big data : current status, and forecast to the future. *ACM sIGKDD Explorations Newsletter*, 14(2) :1–5.

Fei, G., Mukherjee, A., Liu, B., Hsu, M., Castellanos, M., and Ghosh, R. (2013). Exploiting burstiness in reviews for review spammer detection. *ICWSM*, 13 :175–184.

Feldman, J., Muthukrishnan, S., Sidiropoulos, A., Stein, C., and Svitkina, Z. (2010). On distributing symmetric streaming computations. *ACM Transactions on Algorithms (TALG)*, 6(4) :66.

Feldman, R. (2002). Link analysis : Current state of the art. *Tutorial at the KDD*, 2.

Feldman, R. and Sanger, J. (2007). *The text mining handbook : advanced approaches in analyzing unstructured data*. Cambridge University Press.

Freitas, A. A. (1999). On rule interestingness measures. *Knowledge-Based Systems*, 12(5) :309–315.

Gaber, M. M., Zaslavsky, A., and Krishnaswamy, S. (2005). Mining data streams : a review. *ACM Sigmod Record*, 34(2) :18–26.

Gan, M. and Dai, H. (2011). Fast mining of non-derivable episode rules in complex sequences. In *Modeling Decision for Artificial Intelligence*, pages 67–78. Springer.

Gao, C. and Wang, J. (2009). Efficient itemset generator discovery over a stream sliding window. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 355–364. ACM.

Gao, C., Wang, J., and Yang, Q. (2011). Efficient mining of closed sequential patterns on stream sliding window. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 1044–1049. IEEE.

Getoor, L. (2003). Link mining : a new data mining challenge. *ACM SIGKDD Explorations Newsletter*, 5(1) :84–89.

Getoor, L. and Diehl, C. P. (2005). Link mining : a survey. *ACM SIGKDD Explorations Newsletter*, 7(2) :3–12.

Geweke, J. and Porter-Hudak, S. (1983). The estimation and application of long-memory times series models.

Ghosh, D., Mishra, A., Ghose, T., and Mohanta, D. K. (2014). A graph theory approach for reliability analysis of phasor measurement units using frequency and duration technique. *Proceedings of the Institution of Mechanical Engineers, Part O : Journal of Risk and Reliability*, 228(6) :567–577.

Ghosh, S., Viswanath, B., Kooti, F., Sharma, N. K., Korlam, G., Benevenuto, F., Ganguly, N., and Gummadi, K. P. (2012). Understanding and combating link farming in the twitter social network. In *Proceedings of the 21st international conference on World Wide Web*, pages 61–70. ACM.

Giannella, C., Han, J., Pei, J., Yan, X., and Yu, P. S. (2003). Mining frequent patterns in data streams at multiple time granularities. *Next generation data mining*, 212 :191–212.

Global Pulse (2009). United nations global pulse. `http://www.unglobalpulse.org/about-new`.

Golab, L. and Özsu, M. T. (2003). Issues in data stream management. *ACM Sigmod Record*, 32(2) :5–14.

Granger, C. W. J. and Newbold, P. (2014). *Forecasting economic time series*. Academic Press.

Greene, S. and Resnik, P. (2009). More than words : Syntactic packaging and implicit sentiment. In *Proceedings of human language technologies : The 2009 annual conference of the north american chapter of the association for computational linguistics*, pages 503–511. Association for Computational Linguistics.

Gündüz, Ş. and Özsu, M. T. (2003). A web page prediction model based on click-stream tree representation of user behavior. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–540. ACM.

Guo, G., Zhang, L., Liu, Q., Chen, E., Zhu, F., and Guan, C. (2014). High utility episode mining made practical and fast. In *Advanced Data Mining and Applications*, pages 71–84. Springer.

Guo, T., Lin, S., Wang, Y., and Qiao, J. (2012). A new framework for detecting high-utility episodes in event sequence. In *2012 IEEE International Conference on Oxide Materials for Electronic Engineering (OMEE)*.

Gupta, V. and Lehal, G. S. (2009). A survey of text mining techniques and applications. *Journal of emerging technologies in web intelligence*, 1(1) :60–76.

Halpin, H. and Blanco, R. (2012). Machine-learning for spammer detection in crowd-sourcing. In *Workshop on Human Computation at AAAI, Technical Report WS-12-08*, pages 85–86.

Han, J., Cai, Y., and Cercone, N. (1992). Knowledge discovery in databases : An attribute-oriented approach. In *VLDB*, volume 92, pages 24–27.

Han, J., Kamber, M., and Pei, J. (2011). *Data mining : concepts and techniques*. Elsevier.

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

Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M. (2001). Prefixspan : Mining sequential patterns efficiently by prefix-projected pattern growth. In *proceedings of the 17th international conference on data engineering*, pages 215–224.

Han, J., Pei, J., and Yin, Y. (2000b). Mining frequent patterns without candidate generation. In *ACM SIGMOD Record*, volume 29, pages 1–12. ACM.

Hanson, R., Stutz, J., and Cheeseman, P. (1991). *Bayesian classification theory*. NASA Ames Research Center, Artificial Intelligence Research Branch.

Harms, S. K. and Deogun, J. S. (2004). Sequential association rule mining with time lags. *Journal of Intelligent Information Systems*, 22(1) :7–22.

He, W., Zha, S., and Li, L. (2013). Social media competitive analysis and text mining : A case study in the pizza industry. *International Journal of Information Management*, 33(3) :464–472.

Ho, G., Lau, H., Lee, C., Ip, A., and Pun, K. (2006). An intelligent production workflow mining system for continual quality enhancement. *The International Journal of Advanced Manufacturing Technology*, 28(7-8) :792–809.

Holmes, G., Donkin, A., and Witten, I. H. (1994). Weka : A machine learning workbench. In *Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on*, pages 357–361. IEEE.

Hu, J. and Mojsilovic, A. (2007). High-utility pattern mining : A method for discovery of high-utility item sets. *Pattern Recognition*, 40(11) :3317–3324.

Hu, Z., Liu, W., and Wang, H. (2013). Mining both frequent and rare episodes in multiple data streams. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2013 10th International Conference on*, pages 753–761. IEEE.

Huang, K.-Y. and Chang, C.-H. (2008). Efficient mining of frequent episodes from complex sequences. *Information Systems*, 33(1) :96–114.

IBM (2015). Le big data à l'écoute de votre business. `https://www-01.ibm.com/software/fr/data/bigdata/`.

Ilcinkas, D., Klasing, R., and Wade, A. M. (2014). Exploration of constantly connected dynamic graphs based on cactuses. In *Structural Information and Communication Complexity*, pages 250–262. Springer.

Iwanuma, K., Ishihara, R., Takano, Y., and Nabeshima, H. (2005). Extracting frequent subsequences from a single long data sequence a novel anti-monotonic measure and a simple on-line algorithm. In *Data Mining, Fifth IEEE International Conference on Data Mining*, pages 8–pp. IEEE.

Jain, A. K. and Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc.

Jain, D., Khatri, P., Soni, R., and Chaurasia, B. K. (2012). Hiding sensitive association rules without altering the support of sensitive item (s). In *Advances in Computer Science and Information Technology. Networks and Communications*, pages 500–509. Springer.

Jiang, N. and Gruenwald, L. (2006a). Cfi-stream : mining closed frequent itemsets in data streams. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 592–597. ACM.

Jiang, N. and Gruenwald, L. (2006b). Research issues in data stream association rule mining. *ACM Sigmod Record*, 35(1) :14–19.

Jiang, X., Zhang, X., Gui, W., Gao, F., Wang, P., and Zhou, F. (2012). Summarizing semantic associations based on focused association graph. In *Advanced Data Mining and Applications*, pages 564–576. Springer.

Joshi, M. V., Karypis, G., and Kumar, V. (1999). A universal formulation of sequential patterns. In *Proceedings of the KDD ?2001 workshop on Temporal Data Mining*. Citeseer.

Kaufman, L. and Rousseeuw, P. J. (2009). *Finding groups in data : an introduction to cluster analysis*, volume 344. John Wiley & Sons.

Kaur, M. and Kang, S. (2016). Market basket analysis : Identify the changing trends of market data using association rule mining. *Procedia Computer Science*, 85 :78–85.

Kim, D. and Yun, U. (2016). Efficient mining of high utility pattern with considering of rarity and length. *Applied Intelligence*, pages 1–22.

Korzaan, M. L. (2003). Going with the flow : Predicting online purchase intentions. *The Journal of Computer Information Systems*, 43(4) :25.

Kosala, R. and Blockeel, H. (2000). Web mining research : A survey. *ACM Sigkdd Explorations Newsletter*, 2(1) :1–15.

Krempl, G., Žliobaite, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M., et al. (2014). Open challenges for data stream mining research. *ACM SIGKDD Explorations Newsletter*, 16(1) :1–10.

Kumar, V., Andrade, H., Gedik, B., and Wu, K.-L. (2010). Deduce : at the intersection of mapreduce and stream processing. In *Proceedings of the 13th International Conference on Extending Database Technology*, pages 657–662. ACM.

Lakhmi, J. and Wu, X. (2005). *Advanced Methods for Knowledge Discovery from Complex Data*. Springer.

Lamirel, J.-C. and Cuxac, P. (2013). Nouvelles méthodes statistiques pour la traitement des données textuelles volumineuses et changeantes. In *Premières Rencontres Scientifiques du Réseau Mixte LaFEF (Langue Française et Expressions Francophones)*.

Laney, D. (2001). 3d data management : Controlling data volume, velocity, and variety. `http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf`.

Laxman, S., Sastry, P., Shadid, B. Q., and Unnikrishnan, K. (2009). Root cause diagnostics using temporal data mining. US Patent 7,509,234.

Laxman, S., Sastry, P., and Unnikrishnan, K. (2005). Discovering frequent episodes and learning hidden markov models : A formal connection. *Knowledge and Data Engineering, IEEE Transactions on*, 17(11) :1505–1517.

Laxman, S., Sastry, P., and Unnikrishnan, K. (2007). A fast algorithm for finding frequent episodes in event streams. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 410–419. ACM.

Laxman, S. and Sastry, P. S. (2006). A survey of temporal data mining. *Sadhana*, 31(2).

Laxman, S., Tankasali, V., and White, R. W. (2008). Stream prediction using a generative model based on frequent episodes in event sequences. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 453–461. ACM.

Lee, K., Caverlee, J., and Webb, S. (2010). Uncovering social spammers : social honeypots+ machine learning. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 435–442. ACM.

Leleu, M., Méger, N., and Rigotti, C. (2004). Extraction de motifs séquentiels fréquents sous contraintes dans des données contenant des répétitions consécutives. *Ingénierie des systèmes d'information*, 9(3-4) :133–159.

Letham, B., Rudin, C., and Madigan, D. (2013). Sequential event prediction. *Machine Learning Journal*, 93 :357–380.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Forschungsbericht.–707–710 S.*

Li, J., Fu, A. W.-c., and Fahey, P. (2009). Efficient discovery of risk patterns in medical data. *Artificial intelligence in Medicine*, 45(1) :77–89.

Li, J., Fu, A. W.-c., He, H., Chen, J., Jin, H., McAullay, D., Williams, G., Sparks, R., and Kelman, C. (2005). Mining risk patterns in medical data. In *11th ACM SIGKDD*, pages 770–775. ACM.

Li, S., Lee, S. Y. M., Chen, Y., Huang, C.-R., and Zhou, G. (2010). Sentiment classification and polarity shifting. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 635–643. Association for Computational Linguistics.

Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., and Thrun, S. (2008). Anytime search in dynamic graphs. *Artificial Intelligence*, 172(14) :1613–1643.

Lin, J. C.-W., Gan, W., Hong, T.-P., and Tseng, V. S. (2015a). Efficient algorithms for mining up-to-date high-utility patterns. *Advanced Engineering Informatics*, 29(3) :648–661.

Lin, Y.-F., Wu, C.-W., Huang, C.-F., and Tseng, V. S. (2015b). Discovering utility-based episode rules in complex event sequences. *Expert Systems with Applications*, 42(12) :5303–5314.

Linoff, G. S. and Berry, M. J. (2011). *Data mining techniques : for marketing, sales, and customer relationship management*. John Wiley & Sons.

Liu, B. (2011). Association rules and sequential patterns. In *Web Data Mining*, pages 17–62. Springer.

Liu, M. and Qu, J. (2012). Mining high utility itemsets without candidate generation. In *21st ACM international conference on Information and knowledge management*, pages 55–64. ACM.

Liu, Y., Liao, W.-k., and Choudhary, A. (2005). A two-phase algorithm for fast discovery of high utility itemsets. In *Advances in Knowledge Discovery and Data Mining*, pages 689–695. Springer.

Lohr, S. (2012). The age of big data , *the new york times*. `http://www.nytimes.com/2012/02/12/sunday-review/big-datas-impact-in-the-world.html?_r=0`.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee.

Lu, P. (1997). Entry guidance and trajectory control for reusable launch vehicle. *Journal of Guidance, Control, and Dynamics*, 20(1) :143–149.

Lu, Q. and Getoor, L. (2003). Link-based classification. In *ICML*, volume 3, pages 496–503.

Luhr, S., Venkatesh, S., and West, G. (2005). Emergent intertransaction association rules for abnormality detection in intelligent environments. In *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2005. Proceedings of the 2005 International Conference on*, pages 343–347. IEEE.

Luo, J. and Bridges, S. M. (2000). Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection. *International Journal of Intelligent Systems*, 15(8) :687–703.

Ma, Z., Sun, A., and Cong, G. (2013). On predicting the popularity of newly emerging hashtags in twitter. *Journal of the American Society for Information Science and Technology*, 64(7) :1399–1410.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Mannila, H. and Toivonen, H. (1996). Discovering generalized episodes using minimal occurrences. In *KDD*, volume 96, pages 146–151.

Mannila, H., Toivonen, H., and Verkamo, A. I. (1995). Discovering frequent episodes in sequences extended abstract. In *1st Conference on Knowledge Discovery and Data Mining, Montreal, CA*.

Mannila, H., Toivonen, H., and Verkamo, A. I. (1997). Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3) :259–289.

Marcoccia, M. (2000). Les smileys : une représentation iconique des émotions dans la communication médiatisée par ordinateur. *Les émotions dans les interactions communicatives*, pages 249–263.

Masseglia, F., Cathala, F., and Poncelet, P. (1998). The psp approach for mining sequential patterns. In *Principles of Data Mining and Knowledge Discovery*, pages 176–184. Springer.

Mastorocostas, P., Hilas, C. S., Dova, S. C., Varsamis, D. N., et al. (2012). Forecasting of telecommunications time-series via an orthogonal least squares-based fuzzy model. In *Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on*, pages 1–8. IEEE.

Mathioudakis, M. and Koudas, N. (2010). Twittermonitor : trend detection over the twitter stream. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 1155–1158. ACM.

Maynard, D., Bontcheva, K., and Rout, D. (2012). Challenges in developing opinion mining tools for social media. *Proceedings of the@ NLP can u tag# usergeneratedcontent*, pages 15–22.

McAfee, A., Brynjolfsson, E., Davenport, T. H., Patil, D., and Barton, D. (2012). Big data. *The management revolution. Harvard Bus Rev*, 90(10) :61–67.

McGregor, A. (2014). Graph stream algorithms : A survey. *ACM SIGMOD Record*, 43(1) :9–20.

Méger, N. and Rigotti, C. (2004). Constraint-based mining of episode rules and optimal window sizes. In *Knowledge Discovery in Databases : PKDD 2004*, pages 313–324. Springer.

Midas, C. D. et al. (2010). Résumé généraliste de flux de données. In *EGC*, pages 255–260.

Morris, J. P. and Smith, D. L. (2008). Projectile trajectory control system. US Patent 7,354,017.

Muthukrishnan, S. (2005). *Data streams : Algorithms and applications.* Now Publishers Inc.

Nakahara, T. and Yada, K. (2012). Analyzing consumers ? shopping behavior using rfid data and pattern mining. *Advances in Data Analysis and Classification*, 6(4) :355–365.

Neeraj, S. and Swati, L. S. (2012). Overview of non-redundant association rule mining. *Research Journal of Recent Sciences ISSN*, 2277 :2502.

Neumeyer, L., Robbins, B., Nair, A., and Kesari, A. (2010). S4 : Distributed stream computing platform. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 170–177. IEEE.

Nguyen, H. L. (2015). An efficient algorithm for mining weighted frequent itemsets using adaptive weights. *International Journal of Intelligent Systems and Applications (IJISA)*, 7(11) :41.

Nikolopoulos, S. D., Palios, L., and Papadopoulos, C. (2012). A fully dynamic algorithm for the recognition of p4-sparse graphs. *Theoretical Computer Science*, 439 :41–57.

O'Callaghan, D., Harrigan, M., Carthy, J., and Cunningham, P. (2012). Network analysis of recurring youtube spam campaigns. In *ICWSM.*

Ochsenbein, F. and Ortiz, P. F. (2001). Data mining across heterogeneous data. In *Mining the Sky*, pages 664–670. Springer.

Oda, K., Kim, J.-D., Ohta, T., Okanohara, D., Matsuzaki, T., Tateisi, Y., and Tsujii, J. (2008). New challenges for text mining : mapping between text and manually curated pathways. *BMC bioinformatics*, 9(Suppl 3) :S5.

Pasquier, N., Bastide, Y., Taouil, R., and Lakhal, L. (1999). Discovering frequent closed itemsets for association rules. In *Database Theory-ICDT*, pages 398–416. Springer.

Piateski, G. and Frawley, W. (1991). *Knowledge discovery in databases.* MIT press.

Pietsch, W. (2013). Big data–the new science of complexity. *Munich Center for Technology in Society, Technische Universität München, Germany.*

Prasad, B. R. and Agarwal, S. (2014). Handling big data stream analytics using samoa framework-a practical experience. *International Journal of Database Theory & Application*, 7(4).

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1) :81–106.

Radinsky, K. (2014). You need an algorithm, not a data scientist. *Harvard Business Review.*

Rahal, I., Ren, D., Wu, W., and Perrizo, W. (2004). Mining confident minimal rules with fixed-consequents. In *16th IEEE ICTAI 2004*, pages 6–13.

Rajpathak, D., Chougule, R., and Bandyopadhyay, P. (2012). A domain-specific decision support system for knowledge discovery using association and text mining. *Knowledge and information systems*, 31(3) :405–432.

Rajpathak, D. G. (2013). An ontology based text mining system for knowledge discovery from the diagnosis data in the automotive domain. *Computers in Industry*, 64(5) :565–580.

Ramsay, J. O. (2006). *Functional data analysis*. Wiley Online Library.

Raza, K. (2012). Application of data mining in bioinformatics. *arXiv preprint arXiv :1205.1125*.

Rebholz-Schuhmann, D., Kirsch, H., and Couto, F. (2005). Facts from text-is text mining ready to deliver. *PLoS Biol*, 3(2) :e65.

Richardson, M. and Domingos, P. M. (2001). The intelligent surfer : Probabilistic combination of link and content information in pagerank. In *NIPS*, pages 1441–1448.

Rishe, N., Athauda, R. I., Yuan, J., and Chen, S.-C. (2000). Semantic relations : The key to integrating and query processing in heterogeneous databases. In *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics*, volume 7, pages 717–722.

Saha, A. and Sindhwani, V. (2012). Learning evolving and emerging topics in social media : a dynamic nmf approach with temporal regularization. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 693–702.

Samb, M. L., Camara, F., Ndiaye, S., Slimani, Y., and Esseghir, M. A. (2012). Approche de sélection d ?attributs pour la classification basée sur l ?algorithme rfe-svm.

Samet, H. (1990). Applications of spatial data structures.

Smart Grid Big Data Management Team (2014). The right big data technology for seat grid distributed stream computing. *Accenture Blog*.

Soni, J., Ansari, U., Sharma, D., and Soni, S. (2011). Predictive data mining for medical diagnosis : An overview of heart disease prediction. *International Journal of Computer Applications*, 17(8) :43–48.

Srikant, R. and Agrawal, R. (1996). *Mining sequential patterns : Generalizations and performance improvements*. Springer.

Srikant, R., Vu, Q., and Agrawal, R. (1997). Mining association rules with item constraints. In *KDD*, volume 97, pages 67–73.

Srinivasan, S. M., Vural, S., King, B. R., and Guda, C. (2013). Mining for class-specific motifs in protein sequence classification. *BMC bioinformatics*, 14(1) :96.

Štěpánek, P., Zahradníček, P., and Huth, R. (2011). Interpolation techniques used for data quality control and calculation of technical series : an example of central european daily time series. *Időjárás*, 115(1-2) :87–98.

Steyerberg, E. (2008). *Clinical prediction models : a practical approach to development, validation, and updating*. Springer Science & Business Media.

Stone, H. S. and Sipala, P. (1986). The average complexity of depth-first search with backtracking and cutoff. *IBM Journal of Research and Development*, 30(3) :242–258.

Tan, A.-H. et al. (1999). Text mining : The state of the art and the challenges. In *Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases*, volume 8, page 65.

Tan, P.-N., Kumar, V., and Srivastava, J. (2002). Selecting the right interestingness measure for association patterns. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 32–41. ACM.

Tatti, N. and Cule, B. (2012). Mining closed strict episodes. *Data Mining and Knowledge Discovery*, 25(1) :34–66.

Töws, D., Hassani, M., Beecks, C., and Seidl, T. (2015). Optimizing sequential pattern mining within multiple streams. In *BTW Workshops*, pages 223–232.

Tsai, P. S. (2009). Mining frequent itemsets in data streams using the weighted sliding window model. *Expert Systems with Applications*, 36(9) :11617–11625.

Tseng, J. C., Gu, J.-Y., Wang, P., Chen, C.-Y., and Tseng, V. S. (2015). A novel complex-events analytical system using episode pattern mining techniques. In *Intelligence Science and Big Data Engineering. Big Data and Machine Learning Techniques*, pages 487–498. Springer.

Tseng, V. S., Chu, C.-J., and Liang, T. (2006). Efficient mining of temporal high utility itemsets from data streams. In *Second International Workshop on Utility-Based Data Mining*, page 18. Citeseer.

Tseng, V. S., Shie, B.-E., Wu, C.-W., and Yu, P. S. (2013). Efficient algorithms for mining high utility itemsets from transactional databases. *Knowledge and Data Engineering, IEEE Transactions on*, 25(8) :1772–1786.

Tseng, V. S., Wu, C.-W., Shie, B.-E., and Yu, P. S. (2010). Up-growth : an efficient algorithm for high utility itemset mining. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 253–262. ACM.

Tung, A. K., Lu, H., Han, J., and Feng, L. (1999). Breaking the barrier of transactions : Mining inter-transaction association rules. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 297–301. ACM.

Uryas'ev, S. (1988). On the anti-monotonicity of differential mappings connected with general equilibrium problem. *Optimization*, 19(5) :693–709.

van Dam, J.-W. and van de Velden, M. (2015). Online profiling and clustering of facebook users. *Decision Support Systems*, 70 :60–72.

Vapnik, V. N. and Vapnik, V. (1998). *Statistical learning theory*, volume 1. Wiley New York.

Veltkamp, R. C. (2001). Shape matching : similarity measures and algorithms. In *Shape Modeling and Applications, SMI 2001 International Conference on.*, pages 188–197. IEEE.

Verhoef, P. C., Antonides, G., and de Hoog, A. N. (2004). Service encounters as a sequence of events the importance of peak experiences. *Journal of Service Research*, 7(1) :53–64.

Vincini, M., Beneventano, D., and Bergamaschi, S. (2013). Semantic integration of heterogeneous data sources in the momis data transformation system. *J. UCS*, 19(13) :1986–2012.

Vinodhini, G. and Chandrasekaran, R. (2012). Sentiment analysis and opinion mining : a survey. *International Journal*, 2(6).

Viswanathan, V. and Krishnamurthi, I. (2012). Finding relevant semantic association paths through user-specific intermediate entities. *Human-centric Computing and Information Sciences*, 2(1) :1–11.

Wagner, R. A. and Fischer, M. J. (1974). The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1) :168–173.

Wan, E. A. (1989). Neural network classification : a bayesian interpretation. *IEEE transactions on neural networks/a publication of the IEEE Neural Networks Council*, 1(4) :303–305.

Wang, S.-L. and Jafari, A. (2005). Hiding sensitive predictive association rules. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 1, pages 164–169. Ieee.

Wang, W., Yang, J., and Yu, P. (2004). War : weighted association rules for item intensities. *Knowledge and Information Systems*, 6(2) :203–229.

Wang, W., Yang, J., and Yu, P. S. (2000). Efficient mining of weighted association rules (war). In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 270–274. ACM.

Wang, Y., Stash, N., Aroyo, L., Hollink, L., and Schreiber, G. (2009). Using semantic relations for content-based recommender systems in cultural heritage. In *Proceedings of the Workshop on Ontology Patterns (WOP) at ISWC*, pages 16–28.

Washio, T. and Motoda, H. (2003). State of the art of graph-based data mining. *Acm Sigkdd Explorations Newsletter*, 5(1) :59–68.

Weiss, S. M. and Indurkhya, N. (1998). *Predictive data mining : a practical guide*. Morgan Kaufmann.

Wong, W.-K., Moore, A., Cooper, G., and Wagner, M. (2005). What's strange about recent events (wsare) : An algorithm for the early detection of disease outbreaks. *The Journal of Machine Learning Research*, 6 :1961–1998.

Wu, C.-W., Lin, Y.-F., Yu, P. S., and Tseng, V. S. (2013). Mining high utility episodes in complex event sequences. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 536–544. ACM.

Wu, X., Zhu, X., Wu, G.-Q., and Ding, W. (2014). Data mining with big data. *Knowledge and Data Engineering, IEEE Transactions on*, 26(1) :97–107.

Xiong, J., Tang, S., Guo, J., and Wu, X. (2013). Rapid ascent trajectory optimization for a solid rocket engine vehicle. In *Information Science and Technology (ICIST), 2013 International Conference on*, pages 142–145. IEEE.

Yang, W., Liu, X., Zhang, L., and Yang, L. T. (2013). Big data real-time processing based on storm. In *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 1784–1787. IEEE.

Yao, H., Hamilton, H. J., and Butz, C. J. (2004). A foundational approach to mining itemset utilities from databases. In *SDM*, volume 4, pages 215–221. SIAM.

Yin, X. and Han, J. (2003). Cpar : Classification based on predictive association rules. In *SDM*, volume 3, pages 331–335. SIAM.

Yun, U. and Lee, G. (2016). Sliding window based weighted erasable stream pattern mining for stream data applications. *Future Generation Computer Systems*.

Yun, U. and Leggett, J. J. (2005). Wfim : Weighted frequent itemset mining with a weight range and a minimum weight. In *SDM*, pages 636–640. SIAM.

Yun, U., Pyun, G., and Yoon, E. (2015). Efficient mining of robust closed weighted sequential patterns without information loss. *International Journal on Artificial Intelligence Tools*, 24(01) :1550007.

Yun, U. and Ryu, K. H. (2013). Efficient mining of maximal correlated weight frequent patterns. *Intelligent Data Analysis*, 17(5) :917–939.

Zaki, M. J. (1998). Efficient enumeration of frequent sequences. In *Proceedings of the seventh international conference on Information and knowledge management*, pages 68–75. ACM.

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

Zakrzewska, A. and Bader, D. A. (2015). A dynamic algorithm for local community detection in graphs. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 559–564. ACM.

Zargayouna, H. and Salotti, S. (2004). Mesure de similarité dans une ontologie pour l'indexation sémantique de documents xml. In *15èmes Journées francophones d'Ingénierie des Connaissances*, pages 249–260. Presses universitaires de Grenoble.

Zhang, J., Borisov, N., and Yurcik, W. (2006). Outsourcing security analysis with anonymized logs. In *Securecomm and Workshops, 2006*, pages 1–9. IEEE.

Zhang, S., Zhang, C., and Yan, X. (2003). Post-mining : maintenance of association rules by weighting. *Information Systems*, 28(7) :691–707.

Zhang, Y. and Jiao, J. R. (2007). An associative classification-based recommendation system for personalization in b2c e-commerce applications. *Expert Systems with Applications*, 33(2) :357–367.

Zhang, Y. and Lee, W. (2000). Intrusion detection in wireless ad-hoc networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 275–283. ACM.

Zhao, X. W., Guo, Y., He, Y., Jiang, H., Wu, Y., and Li, X. (2014). We know what you want to buy : a demographic-based system for product recommendation on microblogs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1935–1944. ACM.

Zhou, W., Liu, H., and Cheng, H. (2010). Mining closed episodes from event sequences efficiently. In *Advances in Knowledge Discovery and Data Mining*, pages 310–318. Springer.

Zhu, H., Wang, P., He, X., Li, Y., Wang, W., and Shi, B. (2010). Efficient episode mining with minimal and non-overlapping occurrences. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 1211–1216. IEEE.

Zhu, S., Ju, M., Yu, J., Cai, B., and Wang, A. (2015). A review of contrast pattern based data mining. In *Seventh International Conference on Digital Image Processing (ICDIP15)*, pages 96311U–96311U. International Society for Optics and Photonics.

Zhu, Y. and Shasha, D. (2002). Statstream : Statistical monitoring of thousands of data streams in real time. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 358–369. VLDB Endowment.

Zhu, Y. and Shasha, D. (2003). Efficient elastic burst detection in data streams. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 336–345. ACM.

Résumé

Depuis plusieurs années, un nouveau phénomène lié aux données numériques émerge : des données de plus en plus volumineuses, variées et véloces, apparaissent et sont désormais disponibles, elles sont souvent qualifiées de données complexes. Dans cette thèse, nous focalisons sur un type particulier de données complexes : les séquences complexes d'événements, en posant la question suivante : "comment prédire au plus tôt et influencer l'apparition des événements futurs dans une séquence complexe d'événements ?". Tout d'abord, nous traitons le problème de prédiction au plus tôt des événements. Nous proposons un algorithme de fouille de règles d'épisode $DEER$ qui a l'originalité de maîtriser l'horizon d'apparition des événements futurs à travers une distance imposée au sein de règles extraites. Dans un deuxième temps, nous focalisons sur la détection de l'émergence dans un flux d'événements. Nous proposons l'algorithme EER pour la détection au plus tôt de l'émergence de nouvelles règles. Afin d'augmenter la fiabilité de nouvelles règles lorsque leur support est très faible, EER s'appuie sur la similarité entre ces règles et les règles déjà connues. Enfin, nous étudions l'impact porté par des événements sur d'autres dans une séquence d'événements. Nous proposons donc l'algorithme IE qui introduit la notion des "événements influenceurs" et étudie l'influence sur le support, la confiance et la distance à travers trois mesures d'influence proposées. Ces travaux sont évalués et validés par une étude expérimentale menée sur un corpus de données réelles issues de blogs.

Mots-clés: fouille de données, règles d'épisodes, séquence d'événements, prédiction d'événements, détection de l'émergence, événements influenceurs.

Abstract

For several years now, a new phenomenon related to digital data is emerging : data which are increasingly voluminous, varied and rapid, appear and become available, they are often referred to as complex data. In this dissertation, we focus on a particular type of data : complex sequence of events, by asking the following question : "how to predict as soon as possible and to influence the appearance of future events within a complex sequence of events ?". First of all, we focus on the problem of predicting events as soon as possible in a sequence of events. We propose $DEER$: an algorithm for mining episode rules, which has the originality of controlling the horizon of the appearance of future events by imposing a temporal distance within the extracted rules. In a second phase, we address the problem of emergence detection in an events stream. We propose EER : an algorithm for detecting new emergent rules as soon as possible. In order to increase the reliability of new rules, EER relies on the similarity between theses rules and previously extracted rules. At last, we study the impact carried by events on other events within a sequence of events. We thus propose IE : an algorithm that introduces the concept of "influencer events" and studies the influence on the support, on the confidence and on the distance through three proposed measures. Our work is evaluated and validated through an experimental study carried on a real data set of blogs messages.

Keywords: data mining, episode rules, events sequence, events prediction, emergent events, influencer events .