Extraction d'itemsets distinctifs dans les flux de données

Chongsheng Zhang[1] et Florent Masseglia

INRIA Sophia Antipolis-Méditerranée
2004 Route des lucioles - BP 93
06902 Sophia Antipolis Cedex
{Prenom.Nom@sophia.inria.fr},

Résumé. L'extraction d'itemsets distinctifs est un sujet de recherche récent qui connait plusieurs algorithmes pour les données statiques (Knobbe et Ho, 2006; Heikinheimo et al., 2007). Ces solutions ne sont toutefois pas conçues pour le cas des flux de données, pour lesquels les temps de réponse doivent être aussi faibles que possible. Nous considérons le problème de l'extraction d'itemsets distinctifs dans les flux, qui peut avoir de nombreuses applications dans la sélection de variables, la classification ou encore la recherche d'information. Nous proposons l'heuristique *IDkF* (Itemsets Distinctifs dans les Flux) et des résultats d'expérimentations en comparaison d'une technique de la littérature.

1 Introduction

Les flux de données sont des sources produisant de grandes quantités de données à une grande vitesse. Ces caractéristiques nous obligent non seulement à travailler dans un espace mémoire limité par rapport à ces données mais aussi à envisager des méthodes de traitement qui optimisent un compromis indispensable entre la vitesse d'exécution et la précision des résultats. Un des défis les plus importants se situe certainement dans l'aspect évolutif des données mais aussi de leur distribution et des concepts qu'elles véhiculent. Ce dernier point incite à l'élaboration de techniques de traitement efficaces et adaptatives.

La fouille de flux permet d'extraire des connaissances dans les flux en garantissant des temps de réponse appropriés et une marge d'erreur la plus faible possible. Parmi ces connaissances, citons par exemple les itemsets fréquents (Teng et al., 2003; Giannella et al., 2003) (qui seront présentés plus en détail en section 3). Un des problèmes de l'extraction d'itemsets (fréquents ou distinctifs) vient du grand nombre de motifs extraits. Dans Knobbe et Ho (2006), les auteurs proposent l'extraction de motifs appelés Miki (Maximally Informative k-Itemsets). Ces itemsets sont différents des itemsets fréquents dans la mesure où ils minimisent la redondance tout en exprimant une information maximale sur les données. Cette information est mesurée en terme d'entropie.

Exemple 1 *Considérons une application de recherche de documents à partir des informations de la figure 1. Dans cette table, la valeur «1» signifie que la variable est présente dans le*

[1]Ces travaux sont issus du projet MIDAS ANR07-MDCO-008, financé par l ANR.

	A	B	C	D	E
D_1	1	0	1	1	1
D_2	1	1	0	0	1
D_3	1	0	0	1	1
D_4	1	0	1	1	1
D_5	1	1	0	1	1
D_6	0	1	1	1	1
D_7	0	0	1	1	1
D_8	0	1	0	1	1
D_9	0	0	1	1	1
D_{10}	0	1	0	1	1

FIG. 1 – *Variables de documents*

document et la valeur «0» signifie l'absence de cette variable. L'ensemble de variables (D,E) est fréquent car il apparaît avec la valeur «1» dans presque tous les documents. Toutefois, cet ensemble de variables n'est pas d'une grande aide pour la recherche d'un document (les variables ne permettent pas de distinguer les documents). En revanche, il est difficile de synthétiser les valeurs de l'ensemble de variables (A,B,C). Si on affecte à ces variables la combinaison de valeurs (A=1,B=0,C=0) alors on peut trouver le document D_3. En d'autres termes, il s'agit du seul document possèdant la variable A sans posséder les variables B et C. Avec les valeurs (0,1,1) on retrouve le document D_6. Bien que cet ensemble de variables soit non fréquent, étant donné que les variables qui le composent ont souvent une combinaison de valeurs différentes d'un document à l'autre, il permet de distinguer les enregistrements entre eux.

L'exemple 1 illustre les applications possibles en recherche d'information, mais ces applications couvrent de nombreux autres domaines, comme la sélection de variables, la compression ou encore la classification. Nous abordons le problème de l'extraction de ces motifs distinctifs dans les flux de données. Dans le cas des flux, le problème des techniques existantes, comme par exemple l'heuristique *ForwardSelection* proposée par Knobbe et Ho (2006), réside dans l'utilisation de plusieurs passes sur les données. Notre objectif est de proposer une heuristique fonctionnant en temps réel sur les flux pour l'extraction de ces ensembles de variables ou itemsets distinctifs.

Cet article est organisé de la manière suivante : la section 2 donne les définitions préliminaires et la définition de notre problème de recherche. La section 3 présente un état de l'art sur l'extraction des itemsets distinctifs et l'extraction d'itemsets fréquents dans les flux. La section 4 et la section 5 présentent nos contributions pour l'extraction d'itemsets distinctifs dans les flux. Enfin, dans la section 6, nous présentons nos résultats d'expérimentation avant de conclure cet article.

2 Définitions

2.1 Définitions préliminaires

Un flux de données est une séquence infinie de données produite sans interruption et à grande vitesse. Par «grande vitesse» on exprime le fait que les moyens de traitement à disposition sont insuffisants pour appliquer les méthodes traditionnelles d'analyse de données sur

ces flux. Il est généralement admis que dans un flux de données, l'approximation est une clé pour obtenir des résultats avec un processus de fouille de données. Les raisons en sont principalement l'impossibilité de faire plusieurs passes sur les données et l'interdiction de bloquer le flux. Ce sont ces caractéristiques, combinées avec l'intérêt des motifs visés, qui motivent nos travaux.

Définition 1 *Soit $\mathcal{I}$ un ensemble de valeurs possibles. Une **variable**, également appelée **item** est un élément de $\mathcal{I}$.*

Définition 2 *Une* **transaction** *est un triplet $< cid, date, item >$ ou cid représente un identifiant de client, $date$ représente une estampille temporelle et $item$ est l'item associé à cid à cette date.*

Définition 3 *Un **flux de données de transactions** est un flux composé de transactions.*

Exemple 2 *Soit $DS_1 = \{T_1 < c_1, d_1, f_1 >, T_2 < c_2, d_1, f_1 >, T_3 < c_1, d_2, f_2 >, T_4 < c_3, d_2, f_4 >, T_5 < c_1, d_3, f_6 >, T_6 < c_2, d_4, f_2 >\}$ un sous ensemble d'un flux DS, avec $T_1 < c_1, d_1, f_1 >$ la transaction T_1 qui associe l'achat de f_1 par le client c_1 à la date d_1. DS_1 est un flux de données de transactions et (f_1, f_2, f_6) est l'itemset correspondant à c_1.*

Définition 4 *Un **itemset** est un ensemble d'items dans lequel chaque item n'apparait qu'une seule fois, au plus. La taille d'un itemset est le nombre d'items dans l'itemset. Un itemset de taille k est appelé un k-itemset. Dans l'exemple 2, (f_1, f_2, f_6) est un 3-itemset.*

Définition 5 *Un **itemset fréquent** dans un flux de données est un itemset dont la fréquence d'apparition dans les transactions du flux est supérieure à un support donné par l'utilisateur.*

Dans l'exemple 2, si la taille des itemsets fréquents est fixée à 2 et le support minimum est fixé à 2, alors (f_1, f_2) est un itemset fréquent car il apparait 2 fois (dans les transactions des clients c_1 et c_2).

Les définitions suivantes, issues de la théorie de l'information Shannon (2001), expriment des mesures d'incertitude liées aux données observées.

Définition 6 *L'**entropie** d'un item dans le flux est une mesure de l'incertitude liée à la probabilité des valeurs possibles pour cet item dans les données observées. Soit X un item, n le nombre de valeurs possibles pour X et x_i une de ses valeurs possibles. Soit $P(x_i)$ la probabilité que X prenne la valeur x_i. L'entropie de X, notée $H(X)$, est donnée par :*

$$H(X) = -\sum_{i=1}^{n} P(x_i) ln(P(x_i)) \quad (1)$$

Définition 7 *L'**entropie conjointe d'un couple d'items** (X, Y) dans le flux mesure l'information apportée par ces deux items. Soient X et Y deux items et soient x et y les valeurs possibles pour X et Y. Soit $P(x, y)$ la probabilité du couple de valeurs (x, y) pour le couple d'items (X, Y) dans le flux. L'entropie jointe de (X, Y) est donnée par :*

$$H(X, Y) = -\sum_{x,y} P(x, y) ln(P(x, y)) \quad (2)$$

$$MAX(H(X), H(Y)) \leq H(X,Y) \leq H(X) + H(Y) \quad (3)$$

L'équation 3 illustre la monotonicité de l'entropie. Plus le nombre d'items impliqués dans le calcul de l'entropie est grand, plus l'entropie jointe est grande. La différence se situe dans le gain d'information important de certains items alors que d'autres n'apporteront que peu d'information.

Définition 8 *L'**entropie conjointe d'un itemset** I dans le flux (ou **entropie de l'itemset** I) mesure l'information apportée par cet itemset. Soit $\{0,1\}^{|I|}$ l'ensemble des combinaisons possibles de valeurs pour l'itemset I (l'ensemble des combinaisons de valeurs 0 et 1 pour les items de I). L'entropie de I se calcule à partir de $P(I = C)$ pour chaque combinaison C dans $\{0,1\}^{|I|}$ et avec $(I = C)$ l'instanciation de I avec les valeurs de C.*

$$H(I) = -\sum_{C \in \{0,1\}^{|I|}} P(I = C) ln(P(I = C)) \quad (4)$$

Par définition, l'entropie d'un item est comprise entre 0 et 1 et l'entropie d'un itemset est positive ou nulle. Plus l'entropie d'un itemset est forte, plus cet itemset présente de combinaisons de valeurs différentes dans les données. Dans l'exemple 1 l'itemset (A,B,C) a une entropie forte et l'itemset (D,E) une entropie faible.

2.2 Définition du problème

Définition 9 *Un **itemset distinctif** dans le flux de données est un itemset dont l'entropie jointe est supérieure à un seuil minimum donné par l'utilisateur.*

Un itemset distinctif est donc un itemset dont les valeurs observées pour ses items sont très indépendantes. Cette indépendance est mesurée par l'entropie jointe de l'itemset. Plus l'entropie est élevée, plus l'itemset est distinctif.

Définition 10 *L'**itemset distinctif maximal** (IDM) dans le flux de données est l'itemset de taille k ayant la plus forte entropie (comme exprimé par la formule 5).*

$$H(IDM) = MAX\{H(IS_k), IS_k \in SS_k\} \quad (5)$$

Avec :
$IS = \{I_1, , I_2, ..., I_n\}$ l'ensemble des n items existants.
$IS_k = \{I_{m_1}, I_{m_2}, ..., I_{m_k}\}$ un k-itemset (avec $k < n$).
$SS_k = \{IS_k\}$ l'ensemble de tous les itemsets de taille k possibles.

Nous considérons un flux de données de transactions dans lequel les données brutes pour chaque client arrivent de manière ordonnée dans le temps. Le flux de données de transactions nous oblige à extraire les itemsets distinctifs sur une fenêtre d'observation assez grande pour retenir le plus de transactions possible pour chaque client. Par exemple, avec les données de l'exemple 2, si on oublie les informations de la transaction T_2 trop tôt, alors on peut perdre une information importante concernant c_2 et permettant de savoir que, sur le long terme, l'itemset

de c_2 est (f_1,f_2). Un défi préalable à l'extraction des itemsets distinctifs consiste donc à proposer une structure de données capable de résumer les transactions du flux. Cette structure sera décrite en section 4. Notre objectif est d'extraire, en continu, les itemsets distinctifs dans un flux de données de transactions. À notre connaissance, ce problème n'a pas encore été abordé dans la littérature.

3 Travaux existants

L'extraction d'itemsets est un sujet important ayant connu de nombreuses contributions ces dernières années. Toutefois, à notre connaissance, il n'existe aucune proposition pour extraire des itemsets distinctifs dans les flux de données.

3.1 Extraction d'itemsets et entropie (données statiques)

Dans Heikinheimo et al. (2007) les auteurs proposent deux algorithmes destinés à extraire l'ensemble des itemsets présentant une entropie supérieure (respectivement inférieure) à un seuil fixé par l'utilisateur. L'extraction d'itemsets présentant une entropie faible est également un sujet d'importance dans la mesure où de tels itemsets expriment de fortes corrélations entre les valeurs des items qui le composent. Les auteurs exposent un solide cadre théorique sur lequel ils appuyent leurs propositions. Enfin, une analyse des propriétés de l'entropie jointe permet un calcul rapide des itemsets dans une approche par niveaux (pour les itemsets de faible entropie en utilisant la monotonicité de l'entropie) et au moyen de réseaux Bayesiens (pour les itemsets de forte entropie). Toutefois, comme les auteurs le soulignent, la taille du résultat peut être très grande. Malgré l'utilisation d'une structure d'arbre (pourtant destinée à réduire la la taille du résultat) le nombre d'arbres extraits varie de 532 à 44156, pour une base de 2405 observations et 5201 variables. Il apparaît donc très important de filtrer ces résultats. Les auteurs de Knobbe et Ho (2006) proposent une heuristique dont l'objectif est de n'extraire que k itemsets de taille 1 à k, ayant la plus forte entropie possible. Ces itemsets sont nommés miki (Maximally Informative k-Itemsets). L'heuristique *ForwardSelection* effectue plusieurs passes sur les données, en augmentant la taille de l'itemset candidat à chaque passe. Le candidat de taille $t + 1$ est créé en ajoutant au miki de taille t la prochaine variable ayant la plus forte entropie et qui n'appartient pas au miki de taille t. Les auteurs montrent de manière expérimentale l'efficacité *ForwardSelection* par rapport à une approche en force brute qui ferait une évaluation exhaustive des entropies de tous les itemsets possibles de tailles 1 à k.

3.2 Extraction d'itemsets fréquents dans les flux de données

En raison des caractéristiques et contraintes des flux de données, l'approximation est reconnue comme une clé permettant de satisfaire au mieux les objectifs d'analyse de leurs contenus. Dans Giannella et al. (2003), les auteurs proposent FPStream, un algorithme d'extraction d'itemsets fréquents dans un environnement de découpage du flux par batch. FPStream est basé sur la structure de FP-tree sur laquelle l'algorithme FP-Growth (Han et al. (2000)) est appliqué après la lecture de chaque batch. Une fois les itemsets fréquents extraits, ils sont stockés dans la structure de FPTree et leur support pour ce batch est enregistré. L'historique des supports est ensuite peu à peu dégradé dans un système de fenêtres retenant l'information récente avec

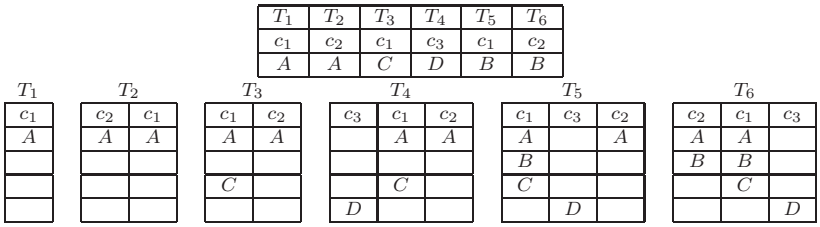

FIG. 2 – *Un flux de données de transactions et sa représentation en FIFO*

un niveau de détail plus élevé que pour les informations anciennes. Teng et al. (2003) utilise l'approximation à la fois dans la phase d'extraction des motifs et dans la phase de gestion de l'historique de leur support. La phase d'extraction est inspirée d'Apriori (Agrawal et al. (1993)) avec une génération d'un ensemble de candidat à chaque batch. La gestion de l'historique du support des motifs extraits est basée sur une technique de régression qui intègre un facteur de vieillissement (la régression se fait sur les anciens segments en priorité).

4 Structure de données

Nous proposons une structure de données permettant d'observer le flux en reconstituant les transactions des clients sur la plus grande fenêtre possible. Cette structure est une file de type First In First Out (FIFO) de taille fixe. Une cellule dans le FIFO représente un client et ses items. On retient pour un client une table de hachage permettant d'indexer les items qui lui sont associés dans les transactions passées. À l'arrivée d'une transaction T_i de la forme $Transaction < client, date, item >$, il y a deux possibilités. Soit le client existe dans le FIFO, alors on le déplace en tête (première place) du FIFO et on met à jour son ensemble d'items avec l'item de la transaction. Soit le client c n'existe pas dans le FIFO, alors on le crée en tête du FIFO et on ajoute i à son ensemble d'items. Quand la taille du FIFO dépasse sa taille maximum alors le dernier client (en queue de FIFO) est supprimé.

Exemple 3 *Considérons le flux représenté par la figure 2 (tableau du haut). Les différentes étapes du flux correspondent aux arrivées des transactions T_1 à T_6 et se traduisent par les tableaux du bas dans la figure 2. Le dernier tableau représente l'état du FIFO après le traitement de la transaction T_6.*

5 Heuristiques pour l'extraction d'IDM dans les flux

Nous avons implémenté deux heuristiques d'extraction d'itemsets distinctifs dans les flux de données. Chacun dépend de trois paramètres : M (la taille du FIFO), n (la fréquence de raffraichissement) et k (la taille de l'itemset à extraire). La fréquence de raffraichissement représente le nombre de transactions lues dans le flux avant que le résultat d'extraction soit mis à jour.

5.1 Évaluation de l'entropie d'un candidat dans la structure

Pour les deux heuristiques, il est important d'optimiser l'évaluation des candidats (mesure de leur entropie dans le FIFO). Pour cela, nous exploiterons le fait que, par convention, $0\ log\ 0\ =\ 0$. Ainsi, il n'est pas nécessaire de calculer toutes les combinaisons de valeurs possibles (*i.e.* $C \in \{0,1\}^{|I|}$) pour l'itemset I comme exprimé à l'équation 4. En effet, il suffit de compter les probabilités d'apparition des valeurs effectivement présentes dans les données. Pour cela, nous retenons les combinaisons des valeurs de l'itemset I pour c dans le FIFO. À la fin de la passe sur le FIFO on peut calculer $H(I) = -\sum_{C\in\{0,1\}^{|I|}} P(I=C)ln(P(I=C))$ par addition des probabilités des combinaisons existantes. L'énumération et le comptage de toutes les combinaisons possibles ($C \in \{0,1\}^{|I|}$) est ainsi évitée.

5.2 Heuristique naïve

La première heuristique que nous proposons pour l'extraction des IDM dans les flux (décrite par la figure 3) se nomme *NaïfID*. Elle consiste à extraire l'IDM en appliquant l'heuristique *ForwardSelection* de Knobbe et Ho (2006) sur les données du FIFO toutes les n transactions. Le principe de *ForwardSelection* repose sur une construction de l'IDM dans une approche par niveau. À chaque étape i, l'item ayant la pus forte entropie et n'appartenant pas à l'IDM de taille i est sélectionné. Cet item est ajouté à l'IDM de taille i puis son entropie est évaluée sur les données. Notre objectif, avec *NaïfID*, est de maintenir une connaissance sur l'itemset distinctif grâce à l'heuristique *ForwardSelection* appliquée régulièrement dans le flux. Ce calcul de l'IDM de taille k sur le FIFO est donc fait de manière systématique.

Heuristique NaïfID
Entrée : F, un flux de données de transactions.
k, la longueur de l'itemset distinctif à extraire.
M, la taille du FIFO.
n, la fréquence de raffraichissement.
Sortie : À chaque lecture de n transactions, IDM : l'itemset distinctif maximal de taille k extrait dans F.
Tant que F produit de nouvelles transactions **Faire**

1. Lire n transactions et mettre à jour le FIFO.
2. $IDM \leftarrow$ *ForwardSelection*(FIFO,k).

Fin Faire
Fin NaïfID

FIG. 3 – *Heuristique NaïfID*

5.3 Heuristique IDkF

IDkF, notre proposition pour extraire les itemsets distinctifs dans les flux de données, applique un principe permettant d'optimiser les temps de calcul. L'initialisation dans le flux se fait grâce à l'heuristique *ForwardSelection* (*i.e. ForwardSelection* est appliquée pour les n premières transactions). Ensuite, après chaque lecture de n transactions, *IDkF* évalue deux

Heuristique *IDkF*
Entrée : F, un flux de données de transactions.
k, la longueur de l'itemset distinctif à extraire.
M, la taille du FIFO.
n, la fréquence de raffraichissement.
Sortie : À chaque lecture de n transactions, IDM : l'itemset distinctif maximal de taille k extrait dans F.

1. Lire n transactions et mettre à jour le FIFO.
2. $IDM \leftarrow$ *ForwardSelection*(FIFO,k).
3. top-k+1 $\leftarrow$ k+1 items ayant la plus forte entropie sur le FIFO.
4. $Cand \leftarrow$ GenererCandidats(IDM,top-k+1).
5. **Tant que** F produit de nouvelles transactions **Faire**
   (a) Lire n transactions et mettre à jour le FIFO.
   (b) $nouvTop \leftarrow$ k+1 items ayant la plus forte entropie sur le FIFO.
   (c) **Si** ($Cand \neq \emptyset$) **Alors** $nouvIDM \leftarrow$ Evaluer($Cand$,FIFO).
   (d) **Sinon** $nouvIDM \leftarrow IDM$
   (e) **Fin Si**
   (f) **Si** ($nouvIDM \neq IDM$) **OU** ($nouvTop \neq$ top-k+1) **Alors**
      i. $IDM \leftarrow nouvIDM$
      ii. top-k+1 $\leftarrow nouvTop$
      iii. $Cand \leftarrow$ GenererCandidats(IDM,top-k+1).
   (g) **Sinon** $Cand \leftarrow \emptyset$
   (h) **Fin Si**
6. **Fin Faire**

Fin *IDkF*

FIG. 4 – *Heuristique IDkF*

conditions afin de déclencher, ou ne pas déclencher, de nouveaux calculs. Ces conditions sont basées sur le fait que :

- l'itemset distinctif a changé depuis les n dernières transactions.
- le classement des $k+1$ variables ayant la plus forte entropie a changé depuis les n dernières transactions.

Enfin, dans *IDkF*, le calcul des itemsets candidats est différent de celui proposé par l'heuristique *ForwardSelection* dans le sens où nos candidats sont générés à partir de IDM_k, l'itemset distinctif maximal de taille k obtenu sur les n précédentes transactions. Ces candidats sont obtenus en calculant toutes les possibilités de remplacer un item de IDM_k par le premier item du top-k+1 qui n'appartient pas déjà à IDM_k. Ce principe de génératon des candidats est décrit par l'algorithme GénérerCandidats à la figure 5. Les candidats de l'ensemble $Cand$ sont ensuite évalués sur les n derniers clients dans le FIFO (*i.e.* Evaluer($Cand$,FIFO)). Si un itemset de l'ensemble des candidats présente une entropie plus forte que celle de l'IDM actuel, alors il devient le nouvel IDM. Notre objectif, avec ce principe, est double : 1) économiser les calculs d'entropie (qui sont coûteux) en les déclenchant uniquement si on observe une variation dans le top-k+1 items et 2) générer plus de candidats que ceux testés par l'heuristique *ForwardSelection* qui peut converger vers un optimum local. Il est évident que pour améliorer les temps de réponse avec *IDkF*, il faut que le deuxième point (générer plus de candidats) soit compensé par le premier (moins de calculs déclenchés). Cette opposition sera analysée en section 6. L'heuristique *IDkF* est décrite par la figure 4.

Algorithme GénérerCandidats
Entrée : I, un itemset.
top, un ensemble d'items tel que $|top| = |I| + 1$.
Sortie : $Cand$, un ensemble de candidats tel que $|Cand| = |I|$.

1. $it \leftarrow i \in top \mid i \notin I$
2. $\forall j \in |I|$ **Faire**
   (a) $Cand \leftarrow Cand \bigcup [\{I/j\} \cup \{it\}]$

Fin GénérerCandidats

FIG. 5 – *Algorithme GénérerCandidats*

6 Expérimentations

Nous avons évalué les performances des heuristiques *NaïfID* et *IDkF* en termes de temps d'exécution et d'entropie de l'IDM extrait sur un jeu de données réelles. Ces mesures permettent d'évaluer le degré d'approximation introduit par chacune de ces heuristiques dans la découverte de l'IDM. Notre jeu de données est issu des navigations sur le portail mobile d'un opérateur de télécom. Il s'agit du flux de clics effectués sur les pages du portail par les utilisateurs à partir de leur téléphone mobile. Les programmes sont écrits en C++ et exécutés sur une machine équipée d'un processeur Intel cadencé à 2,33 Ghz, avec un noyau Linux 2.6.27.

6.1 Description des données

Les données brutes représentent 24 Go pour un échantillon prélevé sur 3 mois d'enregistrements, de mai 2008 à juillet 2008. Une fois transformée, ces données présentent sur chaque ligne une transaction de la forme $< cid, date, variable >$ où cid est l'identifiant unique d'un client, $date$ est la date de la transaction et $variable$ représente la page demandée par cid au temps $date$. Les variables prennent la forme d'un entier entre 1 et 23.

6.2 Critères d'évaluation

Il n'existe aucune méthode à laquelle nous comparer dans la littérature. Nos expérimentations vont donc porter sur la comparaison de *NaïfID* et *IDkF* sur notre jeu de données. Nous allons mesurer leur temps d'exécution et l'entropie de l'IDM extrait pour chaque proposition. L'influence des paramètres sera mesurée (en particulier k, la longueur de l'IDM extrait) ainsi que l'influence du nombre de variables dans les données. Nos données d'origine présentent 23 variables. Nous avons dérivé plusieurs jeux de données en réduisant le nombre de variables à 15 et à 10 (la sélection est faite en gardant les 15 premières ou les 10 premières variables). Dans ces expérimentations, nous notons M la taille du FIFO, n le nombre de transactions entre chaque rafraîchissement, k la taille de l'IDM et nf le nombre de variables dans le jeu de données.

6.3 Temps d'exécution

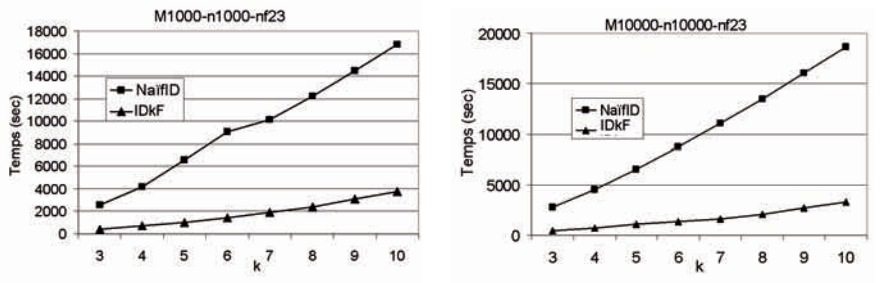

FIG. 6 – *Temps d'exécution de NaïfID et IDkF en fonction de la taille de l'IDM recherché*

Généralement, nous avons constaté une forte différence dans les temps d'exécution en faveur de *IDkF*. En particulier pour les valeurs élevées de k. Cette valeur représente le nombre de passes que doit effectuer *NaïfID* sur le FIFO (toutes les n transactions). La figure 6 illustre ces différences de temps de réponses avec une taille de FIFO M prenant les valeurs 1000 et 10000. Nous avons également constaté l'impact du nombre de variables sur les temps d'exécution. La figure 7 illustre la différence de temps d'exécution entre *NaïfID* et *IDkF* selon que le nombre de variables est 10, 15 ou 23. On peut y observer le caractère exponentiel du temps de réponse en fonction du nombre de variables dans les données, mais aussi la supériorité de *IDkF* en temps d'exécution par rapport à *NaïfID*.

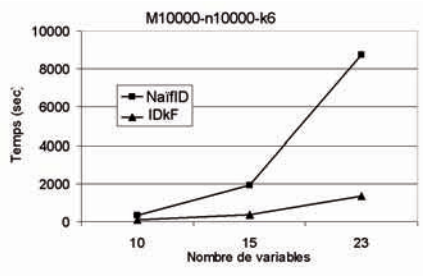

FIG. 7 – *Temps d'exécution de NaïfID et IDkF en fonction du nombre de variables*

6.4 Entropie

Nous avons mesuré l'entropie de l'IDM extrait par *NaïfID* et *IDkF* pour chaque paquet de n transactions lues dans le flux. Sur toutes nos expérimentations, nous avons constaté que l'entropie de l'IDM extrait était identique (ainsi que l'IDM lui même) dans 96% des cas. Dans les 4% restants, nous avons constaté une répartition égale des entropies les plus fortes entre *NaïfID* et *IDkF*. La figure 8 montre l'entropie mesurée toutes les n transactions dans deux cas, pour lesquels *IDkF* (trait épais dans la figure 8) donne de meilleurs résultats que *NaïfID* (trait fin). Les raisons d'une meilleure entropie obtenue par *NaïfID* ou *IDkF* (selon les cas) sont discutées en section 6.5.

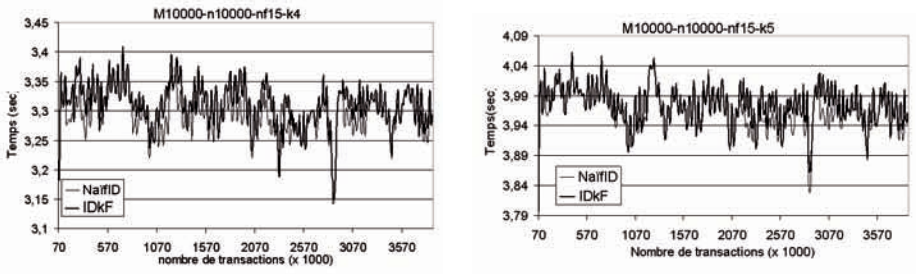

FIG. 8 – *Entropie de l'IDM extrait par NaïfID et IDkF, étape par étape*

6.5 Discussion

Dans 4% des cas, l'entropie obtenue par *IDkF* et *NaïfID* est différente avec une répartition égale des meilleurs résultats. Il est donc nécessaire d'expliquer le fait que dans la moitié de ces 4%, *NaïfID* obtient de meilleurs résultats. En effet, dans certains cas, l'heuristique de déclenchement de la génération des candidats dans *IDkF* n'est pas optimale. Il peut arriver que le top-k+1 ne change pas et que l'IDM réel soit différent de celui extrait sur les n précédentes transactions. Il s'agit d'un temps de décalage qui se produit parfois selon les données.

7 Conclusion

Nous avons présenté le problème de l'extraction des IDM (des itemsets à forte entropie) dans les flux de données. Pour extraire ces IDM, nous avons proposé deux heuristiques. La première est basée sur une technique existante destinée aux données statiques appliquée dans un cadre de fenêtre glissante. La deuxième permet 1) d'optimiser les calculs en sélectionnant leurs déclenchements et 2) de proposer des candidats qui couvrent un plus large spectre par rapport à la première heuristique. Nos expérimentations ont montré la performance de notre approche qui est, à notre connaissance, la première à réaliser cette extraction dans les flux.

Références

Agrawal, R., T. Imieliński, et A. Swami (1993). Mining association rules between sets of items in large databases. In *SIGMOD '93 : Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, New York, NY, USA, pp. 207–216. ACM.

Giannella, C., J. Han, J. Pei, X. Yan, et P. Yu (2003). *Mining Frequent Patterns in Data Streams at Multiple Time Granularities*. In H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha (eds.), Next Generation Data Mining. AAAI/MIT.

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

Heikinheimo, H., E. Hinkkanen, H. Mannila, T. Mielikäinen, et J. K. Seppänen (2007). Finding low-entropy sets and trees from binary data. In *KDD '07 : Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, USA, pp. 350–359. ACM.

Knobbe, A. J. et E. K. Y. Ho (2006). Maximally informative k-itemsets and their efficient discovery. In *KDD '06 : Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, USA, pp. 237–244. ACM.

Shannon, C. E. (2001). A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev. 5*(1), 3–55.

Teng, W.-G., M.-S. Chen, et P. S. Yu (2003). A Regression-Based Temporal Pattern Mining Scheme for Data Streams. In *VLDB*, pp. 93–104.

Summary

Mining informative itemsets is a new research topic in recent years. There are several existing informative itemset mining algorithms for static data. However, they were designed towards static data and are not adaptable to data streams.We provide a heuristic algorithm for highly informative itemset mining over data streams. Finally, we validate its efficiency and effectiveness through experiments on real usage streaming data.