Prédiction d'événements distants basée sur des règles séquentielles

Lina Fahed*, Philippe Lenca*, Yannis Haralambous*, Riwal Lefort**, Marie-Laure Tallec**

*IMT-Atlantique, Lab-STICC, F-29238 Brest, France
{lina.fahed, philippe.lenca, yannis.haralambous}@imt-atlantique.fr
**Crédit Mutuel ARKEA, Pôle Innovation et Opération, service Datalabs
{riwal.lefort, marie-laure.tallec}@arkea.com

Résumé. Dans cet article, nous nous concentrons sur la prédiction d'évènements distants à travers la fouille de règles séquentielles en proposant l'algorithme *D-SR* (Distant Sequential Rules). L'originalité de *D-SR* réside dans le fait qu'il fouille les règles avec un conséquent temporellement distant de l'antécédent, en appliquant une contrainte de gap minimal entre les deux éléments. Nous proposons d'intégrer *D-SR* dans les algorithmes traditionnels de fouille de règles en tant qu'étape de post-traitement ou pendant le processus de fouille. Les expérimentations montrent l'efficacité de *D-SR* en termes de scalabilité et de performance en prédiction.

1 Introduction

La fouille de données est un domaine qui a pour but la recherche de motifs, de tendances ou de relations cachées dans les données. Depuis son introduction en 1993 par Agrawal (Agrawal et al., 1993), la fouille de motifs séquentiels dans une base de séquences reste toujours aujourd'hui un domaine actif (Boudane et al., 2017). Un motif séquentiel, noté $P = <p_1, \ldots, p_k>$, est une liste ordonnée d'événements. Une occurrence du motif dans une séquence est la série d'instants d'apparition des événements qui le composent dans la limité d'une fenêtre de taille w de la séquence. Le support d'un motif, $supp(P)$, est le nombre de ses occurrences. Un motif est considéré comme fréquent si $\mathrm{supp}(P) \geq \mathrm{minsupp}$ où $\mathrm{minsupp}$ est un seuil prédéfini.

Mis à part la fouille de motifs séquentiels, il est également possible de fouiller des règles d'association séquentielles, appelées « règles séquentielles » et notées $R : P \rightarrow Q$ où $P = <p_1, \ldots, p_k>$ et $Q = <q_1, \ldots, q_e>$. Une règle séquentielle indique que si un ou plusieurs évènements arrivent dans un ordre donné (l'antécédent de la règle), alors un ou plusieurs évènements sont susceptibles d'arriver (le conséquent) avec une certaine probabilité, appelée la confiance de la règle : $\mathrm{conf}(P \rightarrow Q) = \frac{\mathrm{supp}(P \cdot Q)}{\mathrm{supp}(P)}$. Une règle est confiante si $\mathrm{conf}(R) \geq \mathrm{minconf}$, où $\mathrm{minconf}$ est un seuil prédéfini. Les règles séquentielles sont souvent utilisées pour prédire des évènements futurs (le conséquent des règles) (Mannila et al., 1997).

La tâche de fouille de règles séquentielles est souvent décomposée en deux sous-tâches (Agrawal et al., 1993) : la fouille de motifs séquentiels fréquents et la génération de règles confiantes à partir de motifs (Agrawal et al., 1993). Dans la fouille de motifs séquentiels, et

afin d'éviter d'explorer la totalité de l'espace de recherche, il devient important de concevoir des algorithmes efficaces qui imposent des contraintes sur les motifs. Nous pouvons citer par exemple la contrainte de gap (Achar et al., 2013), définie généralement comme une distante temporelle minimale ou maximale entre les occurrences de deux événements consécutifs dans un motif. Plusieurs contraintes ont été proposées pour les règles (et non les motifs), comme dans la fouille de règles non redondantes (Boudane et al., 2017), ou dans celle de règles munies d'une position précise (Ao et al., 2017). Cependant, ces algorithmes requièrent l'ensemble complet de motifs candidats, ce qui est extrêmement coûteux en termes de ressources de calcul. Dans l'état de l'art, certains algorithmes ont été proposés afin de fouiller les règles sans se baser sur une étape de fouille de motifs, comme l'algorithme *TRuleGrowth* (Fournier-Viger et al., 2015) et l'algorithme DEER (Fahed et al., 2018) pour une séquence d'événements.

Du point de vue de l'exploitation des règles séquentielles, comme seul l'ordre des évènements est considéré, celles-ci peuvent être utilisées pour prédire l'évènement suivant. Cependant, aucune information ne peut être fournie sur l'horizon d'apparition de l'évènement futur, ce qui pourrait être important pour certaines applications où il est préférable de prédire des évènements temporellement distants. C'est, par exemple, le cas quand il est utile de disposer d'un certain intervalle de temps avant l'apparition de l'évènement prédit. Considérons, par exemple, le domaine bancaire : une prédiction intéressante serait le fait qu'un client ayant un profil donné pourrait quitter la banque sous x mois. Il peut s'avérer utile de faire cette prédiction au plus tôt, puisque la banque aurait ainsi le temps de réagir en essayant d'éviter l'évènement.

Dans cet article, nous traitons la fouille de règles séquentielles dans le but de prédire des évènements distants. Pour cela, les règles doivent refléter une relation distante entre l'antécédent et le conséquent. Nous nous confrontons ainsi au défi de la *fouille de règles séquentielles avec conséquent distant*. Pour relever ce défi, nous proposons un nouvel algorithme, appelé *D-SR* (Distant-Sequential Rules). À notre connaissance, la fouille de règles séquentielles distantes n'a pas encore été proposée pour les bases de séquences [1].

Nous introduisons l'algorithme *D-SR* dans la section 2. Nos résultats expérimentaux sont présentés en section 3. Nous concluons dans la section 4.

2 Algorithme *D-SR* : comment fouiller des règles séquentielles pour la prédiction d'événements distants ?

Dans cette section, nous présentons l'algorithme *D-SR* (Distant Sequential Rules). Afin de fouiller des règles distantes, l'algorithme impose une contrainte de gap entre l'antécédent et le conséquent pour chaque occurrence de la règle. Nous formalisons la définition de l'occurrence de la règle et la définition du gap comme suit :

Definition 1 *Soit* $R : P \rightarrow Q$ *une règle séquentielle composée du motif* $P.Q = <p_1, \ldots, p_k, q_1, \ldots, q_e>$. *Une* ***occurrence*** *de* R, *notée* $\mathrm{occ}(R)$, *est représentée comme suit :* $\mathrm{occ}(R : P \rightarrow Q) = \mathrm{occ}(P.Q : <p_1, \ldots, p_k, q_1, \ldots, q_e>) = (t_{p_k}, t_{q_1})$.

Definition 2 *Soit l'occurrence* $\mathrm{occ}(R) = \mathrm{occ}(P.Q) = (t_{p_k}, t_{q_1})$. *Elle respecte une* ***contrainte de gap minimal*** *de taille* gap, *si* $(t_{q_1} - t_{p_k}) \geq \mathrm{gap}$.

1. L'idée de règles avec un conséquent distant a été introduite dans le contexte de fouille de règles d'épisode dans une seule et longue séquence d'évènements (Fahed et al., 2018), ce qui est différent de l'objectif du présent article de fouille de règles séquentielles dans une base de séquences (les contraintes de fouille n'étant pas les mêmes).

Afin de pouvoir intégrer notre algorithme dans les algorithmes traditionnels de fouille de règles, nous classons ces derniers en deux familles : (i) ceux qui fouillent les motifs puis construisent les règles, et (ii) ceux qui fouillent directement les règles en fixant le conséquent tôt, durant le processus de fouille. Pour chaque famille d'algorithmes, nous proposons une version de l'algorithme *D-SR* en tant qu'extension qui permette de fouiller des règles distantes. Nous allons présenter deux versions : la fouille de règles distantes (i) durant une étape de post-traitement, et (ii) durant le processus de la fouille.

La fouille de règles distantes durant une étape de post-traitement Pour les algorithmes de l'état de l'art qui fouillent les motifs avant de construire les règles, imposer la contrainte du gap entre l'antécédent et le conséquent ne peut être réalisé durant le processus de fouille, car lorsqu'on ajoute un événement au motif en cours de construction, on ne peut pas savoir s'il fera partie du conséquent ou de l'antécédent de la future règle. Pour cela, nous proposons d'intégrer la contrainte du gap comme étape de post-traitement :

1. Durant la fouille d'un motif, nous proposons de tracer la distance temporelle entre le suffixe du motif et chaque événement ajouté, en sauvegardant leurs instants d'apparition.
2. Une fois la règle construite, ses occurrences qui ne respectent pas la contrainte du gap minimal sont filtrées (voir définition 2). La fréquence et la confiance de la règle sont mises à jour suite à ce filtrage. Seules les règles fréquentes et confiantes pour lesquelles toutes les occurrences respectent la contrainte du gap sont retenues.

Il est important de mentionner que les algorithmes traditionnels (qui fouillent les motifs puis construisent les règles), sur lesquels se base cette version de *D-SR*, consomment beaucoup de ressources en temps et en mémoire afin de construire toutes les règles qui sont par la suite filtrées lors de l'application de notre algorithme. Par conséquent, nous considérons que ces algorithmes ne sont pas les plus adaptés pour la prédiction d'événements distants.

La fouille de règles distantes durant le processus de fouille Récemment, certains algorithmes de l'état de l'art ont été proposés pour fouiller directement les règles sans se baser sur une étape de fouille de motifs. Ces algorithmes commencent par combiner des événements afin de construire directement une règle de taille 1, à savoir une règle avec un seul événement dans l'antécédent et un seul événement dans le conséquent : $R : P \rightarrow Q = p_1 \rightarrow q_1 : p \in P \wedge q \in Q$. Si cette règle est fréquente et confiante, elle est étendue par la suite en ajoutant un événement à l'antécédent ou au conséquent, afin de construire une règle plus longue. Dans cette version de l'algorithme *D-SR*, nous proposons d'intégrer la contrainte du gap durant la fouille de règles comme suit :

1. Dès le début du processus de la fouille, lorsque la règle $R : P \rightarrow Q$ est construite, nous proposons de tracer ses occurrences (à la manière de la définition 1). Ces occurrences sont directement filtrées afin de ne garder que celles qui respectent le gap minimal (définition 2). Suite à cette étape, seules les règles fréquentes et confiantes sont retenues.
2. Lorsque le processus de fouille continue pour obtenir des règles plus longues, l'étape 1 est réitérée. Il est important de mentionner que l'évaluation de la contrainte de gap est réalisée uniquement lorsque l'antécédent de la règle est étendu (en y ajoutant un événement). Il n'est pas nécessaire de faire cette vérification au cas où le conséquent est étendu, car la distance entre le suffixe de l'antécédent p_k et le préfixe du conséquent q_1 reste inchangée.

3 Expérimentations

Cette section est dédiée à l'évaluation de l'algorithme *D-SR*. Le tableau 1 montre les caractéristiques des deux corpus de données utilisés pour valider l'algorithme. Le corpus *Kosarak* [2] est utilisé pour l'évaluation de la scalabilité des algorithmes. Le *corpus bancaire* est un corpus de données réelles anonymisées fourni par le groupe *Crédit Mutuel ARKEA* [3]. Ce corpus représente l'historique des actions (événements) de nombreux clients anonymisés.

Corpus	# Séquences	# Événements	Nombre moyen d'événements par séquence (taille moyenne des séquences)
Kosarak	990.000 (70.000)	21.144	7,97 (max= 796)
Corpus Bancaire	825.741	71	25 (max= 80)

TAB. 1: Caractéristiques des corpus de données.

Performance de *D-SR* durant une étape de post-traitement Nous évaluons, sur le corpus *Kosarak*, la performance de *D-SR* lorsqu'il est intégré en tant qu'étape de post-traitement dans l'algorithme traditionnel *MINEPI+* (Huang et Chang, 2008). Les valeurs des paramètres d'entrée sont : $\mathrm{minsupp} = 0{,}003$, $\mathrm{minconf} = 0{,}5$, $w = 10$ et $\mathrm{gap} = 10$ (le gap est utilisé uniquement pour *D-SR*). Comme prévu, l'algorithme *MINEPI+* consomme beaucoup de temps (30 secondes) pour fouiller toutes les règles (1 900 règles), qui sont filtrées par *D-SR* : 1 seconde pour 520 règles retenues. Nous concluons que les algorithmes de fouille de motifs puis de règles ne sont pas optimaux pour la fouille de règles distantes.

Évaluation de la scalabilité Nous évaluons, sur le corpus *Kosarak*, la scalabilité de *D-SR* lorsqu'il est intégré durant le processus de la fouille de l'algorithme traditionnel *TRuleGrowth* (Fournier-Viger et al., 2015), comparée à celle de l'algorithme de base *TRuleGrowth*, pour $\mathrm{minsupp} = 0{,}003$, $\mathrm{minconf} = 0{,}5$, $w = 10$ et $\mathrm{gap} = 0$. Il est important de préciser que les deux algorithmes ne sont comparables que lorsque $\mathrm{gap} = 0$. Les figures 1 (a) et 1 (b) montrent que le temps d'exécution des deux algorithmes augmente de manière linéaire avec le nombre de séquences et de manière significative pour *TRuleGrowth* avec la taille de fenêtre w. Cela peut être expliqué via les figures 1 (a′) et 1 (b′) qui montrent que *D-SR* fouille significativement beaucoup moins de règles. Aussi, la figure 1 (c) montre que le temps d'exécution diminue considérablement avec la taille du gap, ce qui s'explique par le fait que moins de règles sont obtenues quand on dispose d'un plus grand gap (figure 1 (c′)). Nous concluons que *D-SR* est plus évolutif que *TRuleGrowth*, et qu'en outre, il est plus évolutif en augmentant w qu'en augmentant le nombre de séquences.

Performance dans une tâche de prédiction Nous évaluons la performance en prédiction sur le corpus *Kosarak* (pour $\mathrm{minsupp} = 0{,}002$) en le divisant en deux sous-corpus : 50% pour l'apprentissage des règles et 50% pour le test (prédiction par la règle la plus confiante). Les figures 2 (a) et (a′), montrent qu'en fonction de w, *D-SR* donne toujours une meilleure précision : on obtient jusqu'à 30% de précision supplémentaire malgré une couverture inférieure à celle de *TRuleGrowth*. Nous concluons que l'utilisation de fenêtres est bénéfique, car en augmentant w, plus de règles sont potentiellement plus fréquentes et donc retenues, ce qui améliore leur couverture. Dans les figures 2 (b) et (b′), nous remarquons que la précision diminue en

2. Kosarak dataset : `http://goo.gl/4B6ve5`.
3. Pour des raisons de confidentialité, ce corpus a été anonymisé au sein du groupe Crédit Mutuel ARKEA.

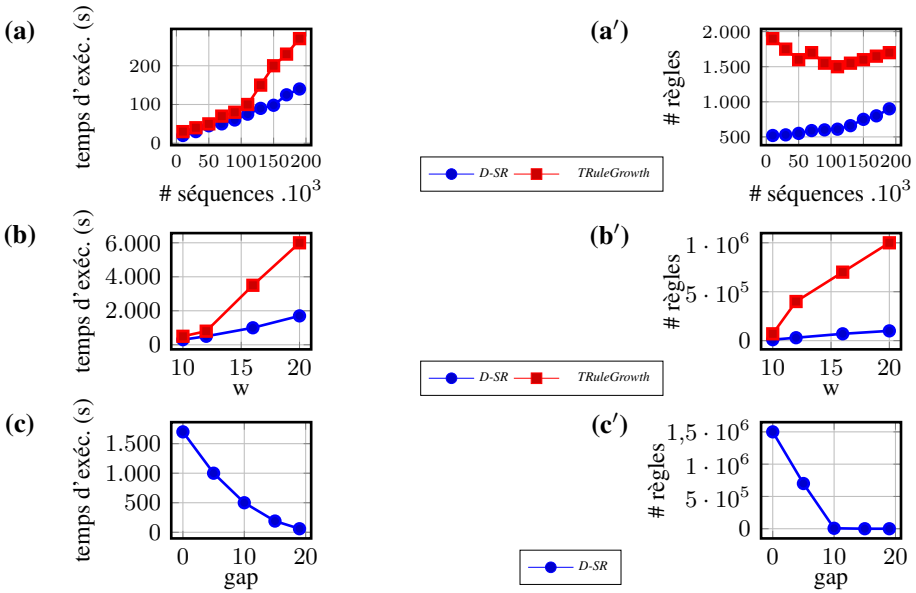

FIG. 1: Scalabilité sur Kosarak : impact du nombre de séquences, w, gap.

augmentant le gap, ce qui est prévisible, car plus le gap est grand, plus la prédiction devient difficile puisque l'objectif est de prédire des événements loin dans le futur. Nous concluons que *D-SR* réussit à prédire des événements distants et surpasse *TRuleGrowth* dans cette tâche.

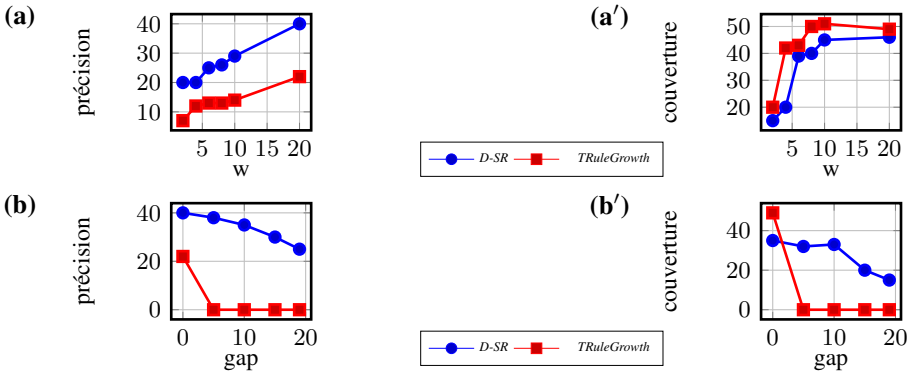

FIG. 2: Performance en prédiction sur Kosarak : impact de w, gap.

Évaluation sur un corpus de données bancaires Sur le corpus bancaire (pour $\text{minsupp} = 0{,}109$, $\text{minconf} = 0{,}8$, $\text{w} = 40$, $\text{gap} = 180$), *D-SR* fouille 200 règles alors que *TRuleGrowth* en fouille 1000, ce qui représente une diminution de 80% du nombre de règles, due au fait que *D-SR* impose plus de contraintes durant la fouille. Un exemple de règle obtenue par *D-SR*

est le suivant : *R : (simulation crédit habitat, annulation contrat assurance habitat, virement extérieur important)* $\rightarrow$ *(départ client)*, avec $\mathrm{gap} = 180$. Cette règle signifie que lorsqu'on détecte un profil client qui effectue une simulation de crédit habitat, puis annule sont contrat d'assurance habitation, et enfin effectue un virement extérieur d'un montant important, cela permet de prédire que ce profil de client partirait probablement après 180 jours. La banque pourrait profiter de ce temps pour contacter le client et lui proposer des offres adaptées.

4 Conclusion et perspectives

Dans cet article, nous proposons l'algorithme *D-SR* pour la fouille de règles séquentielles avec un conséquent distant. Les expérimentations sur deux corpus de données montrent que *D-SR* est plus performant lorsqu'il est intégré durant le processus de la fouille et que ses performances dépassent celles d'autres algorithmes de l'état de l'art. Dans un futur travail, nous envisageons de proposer une nouvelle mesure d'intérêt afin d'apporter plus de flexibilité à la contrainte de gap.

Références

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

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

Ao, X., P. Luo, J. Wang, F. Zhuang, et Q. He (2017). Mining precise-positioning episode rules from event sequences. In *IEEE 33rd Int. Conf. on Data Engineering (ICDE)*, pp. 83–86.

Boudane, A., S. Jabbour, L. Sais, et Y. Salhi (2017). Enumerating non-redundant association rules using satisfiability. In *Conf. on Knowledge Discovery & Data Mining*, pp. 824–836.

Fahed, L., A. Brun, et A. Boyer (2018). Deer : Distant and essential episode rules for early prediction. *Expert Systems with Applications 93*, 283–298.

Fournier-Viger, P., C.-W. Wu, V. S. Tseng, L. Cao, et R. Nkambou (2015). Mining partially-ordered sequential rules common to multiple sequences. *IEEE Transactions on Knowledge and Data Engineering 27*(8), 2203–2216.

Huang, K.-Y. et C.-H. Chang (2008). Efficient mining of frequent episodes from complex sequences. In *Information Systems*, Volume 33, pp. 96–114. Elsevier.

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

Summary

In this paper, we focus on the prediction of distant events by mining sequential rules and propose the algorithm *D-SR* (Distant Sequential Rules). The originality of *D-SR* is that it mines rules with a consequent temporally distant from the antecedent by applying a minimal gap constraint between the two elements. *D-SR* can be integrated into traditional algorithms as a post-processing step or during the mining process. Experiments show the efficiency of *D-SR* in terms of running time, scalability and in a prediction task.