Handling Item Similarity in Behavioral Patterns through General Pattern Mining

Julie Daher, Armelle Brun

Handling Item Similarity in Behavioral Patterns through General Pattern Mining

Julie Bu Daher
Université de Lorraine, CNRS, Loria
54506 Vandoeuvre-lès-Nancy Cedex, France
julie.bu-daher@loria.fr

Armelle Brun
Université de Lorraine, CNRS, Loria
54506 Vandoeuvre-lès-Nancy Cedex, France
armelle.brun@loria.fr

***Abstract*—Modeling human behavior on the Web is often performed by sequential pattern mining (SPM). However, the similarity between data elements often results in the decrease of the number of patterns mined. This work proposes to handle this similarity by managing multiple data sources representing different views of the data. We introduce G_SPM, a behavioral pattern mining algorithm that takes advantage of multi-source data to handle the problem of data similarity. It adopts a selective mining strategy to limit the complexity and forms general patterns to limit the decrease of the patterns.**

Experimental results confirm that G_SPM succeeds in handling the problem of item similarity. In addition, G_SPM outperforms traditional approaches in terms of runtime and redundancy of the resulting set of patterns.

I. Introduction

Nowadays, a significant amount of data is available over the Web. By analyzing these data, we can discover knowledge, extract links between data elements, predict future data, etc. For example, from the customers' traces of activities on an e-commerce website, it is possible to identify frequent purchase behaviors. Obtaining such behavior-related information is the focus of Web usage mining [11] that has attracted attention during the last two decades [2]. Sequential pattern mining, is a key task in data mining. It aims at identifying frequent patterns in data, under the form of sequences of items or itemsets that occur frequently in the data. In Web usage mining, such patterns are behavioral patterns.

Some data characteristics could lead to the loss of possibly interesting patterns and thus could limit the number of frequent patterns mined. This is particularly the case of the similarity that could exist between certain data items. This similarity results in the occurrence of several patterns representing the same behavior. The behavior could be frequent; however, as its frequency is distributed among several patterns, this could result in the infrequency of these patterns and thus their loss.

Let us consider an example of two sequential patterns, p_1 and p_2, representing customers' purchase patterns mined from the log file of an e-commerce website. $p_1 = \langle i_{47}, i_{25}, i_1, i_{19} \rangle$ and $p_2 = \langle i_{47}, i_{25}, i_2, i_{19} \rangle$, with supports of 7 and 8 respectively where i_v is an item id. If the minimum support to consider a pattern as frequent is set to 10, p_1 and p_2 are not frequent. However, we can see that p_1 and p_2 only differ by items i_1 and i_2, both at position 3. Supposing that i_1 and i_2 are stainless steel scissors of 8.3 inches, where i_1 is a green one and i_2 is a purple one, it is clear that these patterns represent the same general purchase behavior. We can thus consider that both patterns are similar, and they are infrequent due to the similarity between i_1 and i_2. The actual purchase pattern is $p = \langle i_{47}, i_{25}, \textit{"stainless steel scissors 8.3 inches"}, i_{19} \rangle$. p encompasses p_1 and p_2, and p is frequent as its support is equal to at least the sum of the supports of p_1 and p_2. We call p a general pattern, and p_1 and p_2 are original patterns. We define a general pattern as a frequent pattern that encompasses a set of similar but infrequent original patterns. Mining general patterns allows to limit the decrease in the number of frequent patterns. However, a general pattern is less precise than an original one. For this reason, frequent patterns will directly be part of the final output, while only infrequent patterns are considered for forming general ones.

Obviously, general patterns cannot be mined from a log file as the descriptions of the items are not part of such data. However, data on the Web can be collected from multiple sources, where each source can represent data from a specific viewpoint. Putting these data together results in a heterogeneous and multi-source dataset and we consider that it can be used to handle the problem of item similarity. An example of a 3-source dataset in relation with the previous example is as follows. A first data source contains the traces of the activities of the customers on the website. A second data source contains descriptive data of customers collected from their customer accounts on the website: age, gender, preferences, etc. A third data source contains descriptive data of products that can be purchased: brand, size, expiry date, etc. Each source thus provides data related to a specific perspective about the retail website: purchase activity, customers or items.

The traditional pattern mining literature either integrates data sources together or mines sources separately. The first approach allows mining general patterns but is highly complex and mined redundant patterns, while the second approach cannot form general patterns. In this work, we propose an in-between approach that mines data sources selectively.

Referring to the previous example, the source containing customers' behavior is denoted the *main source* and is used to mine original patterns. The other sources provide data about customers or products and are denoted *additional sources* and can be used to form general patterns.

We thus propose G_SPM, a General Sequential Pattern

Mining algorithm, that manages multiple sources to cope with the problem of item similarity. It is designed to 1) limit the complexity of the mining process by relying on a selective mining, 2) mine general patterns, while controlling the level of generality of the mined patterns.

This paper is organized as follows. Section 2 reviews the literature in the domain. Section 3 introduces G_SPM. Section 4 is dedicated to the experimental evaluations of G_SPM. Section 5 concludes this work and draws future perspectives.

II. Related Work

A. Sequential Pattern Mining

Pattern mining consists of discovering interesting, useful, and unexpected patterns in large databases [7]. Pattern mining has attracted much attention and a particular emphasis has been given to sequential pattern mining that considers the ordering of elements in a sequential database [6]. A sequential database is made up of a set of sequences where a sequence is an ordered list of itemsets.

The interestingness of a pattern is usually defined as its frequency (support), i.e. the number of sequences in which it appears. The patterns with a support greater than a predefined minimum support min_sup are considered as frequent patterns. The sequential pattern mining (SPM) domain is very active where many algorithms have been proposed. They mainly differ in the way they cross the search space: depth-first or breadth-first search, and in the representation of the database: horizontal, vertical or projected. PrefixSpan [9], and SPAM [3] are popular SPM algorithms. These algorithms are designed to mine a sequential database formed by a unique database. However, data is not always that simple.

B. Multi-* Data Mining

Not only voluminous amounts of data are created and collected on a daily basis on the Web, but they also can be provided by various data sources. Each data source can represent a specific data dimension or data table. The literature uses several terms to refer to such databases We propose to refer to such data as $multi\text{-}* \ data$.

1) Multi-source data: Multi-source data represent data collected from several data sources. Each source provides one or more kinds of data with similar or different structures; The data provided by different sources can have homogeneous or heterogeneous structures.

2) Multi-dimensional data: In multi-dimensional data, each data element is made up of several fields, where one field represents a feature, an attribute, or a dimension [1]. One dimension is often either a data sequence or a set of descriptive attributes. The most popular work proposes two mining strategies [10]. They differ in the priority given to the dimensions in the mining process: sequential or descriptive dimension mined first, as well as in the way they are then attached to each other.

3) Multi-relational data: The term multi-relational data is used in the relational database domain for data provided by multiple database tables (relations) and linked through some of their attributes [4]. The algorithms mainly either combine tables and perform a global process or use id-list propagation to mine itemsets from different tables without combining them. The first approach is complex and generates a huge number of patterns, while the second approach does not support data of heterogeneous kinds nor it supports mining sequential data.

C. Typology of Relations Among Data Sources

To the best of our knowledge, no specific focus has been made on the nature of the relations between tables or data sources and on their impact on the mining process and the mined patterns. There is an exception in the domain of context-based sequential pattern mining, where [14] introduces two types of contexts: the sequence context that provides contextual data to a whole data sequence, and the element context that provides contextual data to each element in the sequence. We choose to exploit these two types of context for determining two kinds of relations between data sources in multi-source data. Figure 1 presents both types of relations for the dataset from the introduction. One data source is sequential and represents customer behavior (main source). It is related to two additional sources, which represent customer and item description, with different types of relations.

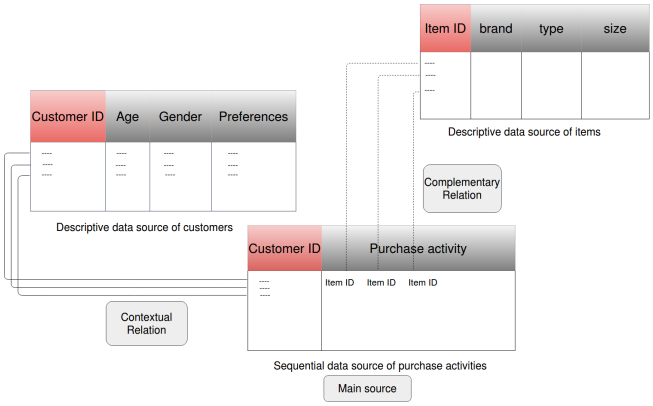

Fig. 1. A 3-source database with contextual and complementary relations

1) The Complementary Relation: A complementary relation connects an element of a sequence of the main source with an element of an additional source which is close to the element context from [14]. As displayed in Figure 1, the elements in both sources represent an item id where it is a primary key in the descriptive source of items and foreign key in the sequential source. We refer to this relation as a complementary relation as it complements elements of the sequence. This additional source is thus called a complementary source. Notice that if both sources are integrated, it will result in a traditional sequential database, made up of sequences of itemsets (in this case item ids with the description of items).

2) The Contextual Relation: A contextual relation connects the id of a sequence of the main source with the same id in the additional source which is close to the sequence context from [14]. In Figure 1, the element of the additional source is the customer id. If sources are integrated, it results in a

multi-dimensional source containing descriptive and sequential dimensions. We refer to this relation as a contextual relation as the additional source can be considered as providing contextual information to the sequence. This additional source is thus called a contextual source.

D. Item Similarity Mining

In the data mining literature, pattern mining in the context of similar items is a problem that has received little interest. We found two frames in which this problem has been considered: text mining and multi-database mining. In text mining, [12] aims to extract synonym terms from a text collection. The proposed approach relies on the identification of common synonym patterns, i.e. word sequences used to connect a phrase and its synonym. In multi-database mining, [13] proposes to identify synonym customers, where the same customer is represented by different ids in different databases, by comparing the elements that describe them in each database. However, to the best of our knowledge, the specific problem of mining similarities in pattern mining has not been addressed.

III. G_SPM: A General Sequential Pattern Mining Algorithm that Handles Item Similarity

The similarity between items in the data leads to a decrease in the support of some patterns and thus to a limited number of mined patterns. In order to handle this problem, G_SPM is designed to mine general sequential patterns by relying on the item-related complementary data source.

Before presenting G_SPM, we first introduce definitions and notations that will be used in the algorithm, and we detail the traditional naive approach.

A. Definitions and Notation

Let $MSDB$ be the multi-source database made up of two sources connected with a complementary relation. S is the sequential source, referred to as the main source. Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of distinct items, concretely item ids. S is made up of a set of sequences $S = \{S_1, \ldots, S_t\}$, with $S_i = \langle s_{i1}, \ldots s_{it_i} \rangle$, with $s_{ij} \in I$.

C is the complementary source. C describes a set of items ($I(C)$). Each item i, $i \in I(C)$, is associated with a set of attributes values, $att(i) = \{a_1, \ldots, a_d\}$ and one of these attributes represents the item id.

1) Original Frequent Pattern: In Web usage mining, a frequent sequential pattern p is an ordered list of items, denoted by $p = \langle e_1, e_2, \ldots, e_n \rangle$ where $e_k \in I$ with $1 \leq k \leq n$. Given a minimum support threshold min_sup, a sequential pattern p is frequent in S if $sup(p) \geq min\_sup$. $sup(p)$ is the support of p and represents the number of sequences in S where p occurs. In this work, such a frequent pattern is referred to as an original frequent pattern.

2) Promising Pattern: Let min_sup_prom be a support threshold, where $min\_sup\_prom < min\_sup$. The original pattern p is a promising pattern if $min\_sup\_prom \leq sup(p) < min\_sup$ i.e. p is not frequent but is almost frequent.

3) Similar Items: Let i_j and i_k be two distinct items. The similarity between these items ($sim(i_j, i_k)$) is evaluated as the ratio of common attributes values between both items ($att(i_j)$ and $att(i_k)$). i_j and i_k are considered as similar if $sim(i_j, i_k) \geq min\_simitem$ where $min\_simitem$ is a threshold representing the minimum ratio of common attributes values between two items.

4) Similar Patterns: Let p_1 and p_2 be two distinct patterns. p_1 and p_2 are similar if 1) they have the same length, 2) they are made up of the same sequence of items except at one position, 3) the items at this position are similar. Given $p_1 = \langle e_{11}, e_{12}, \ldots, e_{1l} \rangle$ and $p_2 = \langle e_{21}, e_{22}, \ldots, e_{2l} \rangle$, p_1 and p_2 are similar if $\exists i, sim(e_{1i}, e_{2i}) \geq min\_simitem \wedge \forall j \in [1..l] \wedge j \neq i, e_{1j} = e_{2j}$.

5) General Pattern: A pattern is a general pattern if at least one of its elements is a description of an item (a set of attributes values). $p = \langle e_1, e_2, \ldots, e_l \rangle$ is a general pattern if $\exists j, k, e_j \subseteq att(i_k)$.

B. The Naive Approach

A study of the literature has highlighted that the traditional approach, that we refer to as the naive approach, first forms a unique dataset by integrating sources and then performs pattern mining on the integrated dataset (SC). The dataset consists of the item ids and their descriptive attributes. As SC is made up of sequences of itemsets, it can be mined by traditional SPM algorithms (see Section II). Although these algorithms are not specifically designed to tackle the item similarity problem, the data contained in SC make these algorithms naturally able to mine general patterns. Referring to the example in the introduction, a pattern mined from SC can be: $p' = \langle \{i_{47}$, *eraser, pink, large*$\}$, $\{i_{25}$, *pencils, wood-cased, pre-sharpened*$\}$,$\{$*stainless, steel scissors, 8.3 inches*$\}$, $\{i_{19}$, *post-it, 3"x3", super-sticky*$\}\rangle$. We can see that the first, second and fourth elements of p' contain redundant information as they contain both the item ids as well as the descriptive attributes. In line with our goal, having only the item id is sufficient as it is frequent. However, such an approach is interesting when considering the third element that is only made of attributes as the item id is not frequent; therefore, a "generalization" has been automatically made. A post-processing step can be made for the three other elements to discard the attributes when the item id is part of an element of a pattern. The resulting pattern will thus be: $p = \langle i_{47}, i_{25}$, *"stainless steel scissors 8.3 inches",* $i_{19} \rangle$, which corresponds to the general pattern. We conclude that this approach allows mining the patterns of interest. However, the resulting set of patterns is huge and redundant; and the mining process is highly complex, not only due to the mining of a complex dataset, but also due to the post-processing.

C. The G_SPM Algorithm

To overcome the limits of the naive approach, we propose G_SPM, a General Sequential Pattern Mining algorithm, that adopts a "mine and select" approach. G_SPM relies on a double assumption. First, due to the similarity between items,

some behaviors may occur under the form of several patterns. The support of this behavior is thus distributed over several patterns with lower supports that could be infrequent. Second, the id of an item is more precise than its attributes as an item id represents a unique item, while a set of attributes may represent several items. Thus, a pattern containing item ids is more precise than a one containing sets of attributes.

G_SPM is designed to mine frequent original patterns as well as frequent general patterns that encompass similar infrequent patterns. It is presented in Algorithm 1 and proceeds as follows. G_SPM starts the mining process by one source, namely the main source, and it first mines original patterns from this source: frequent patterns (F_f, with a support $\geq$ min_sup) and promising patterns (F_p, with $min\_sup\_prom \leq support(F_p) < min\_sup$) (line 2). Any traditional SPM algorithm can be used for this step. Each frequent original pattern is part of the final output set of the algorithm. A complexity gain is thus obtained on these patterns, in comparison to the naive approach of the literature. For each *promising pattern*, G_SPM checks if its relatively low support is due to the similarity between items. So, for each pair of *promising patterns* (lines 3 and 4), G_SPM checks if these patterns are pseudo-similar (line 5), i.e. they have the same length and differ at only one position. If yes (line 6), G_SPM attempts to generalize them and form a general pattern (line 7) by mining the complementary source (C) that contains descriptive attributes of the items. The generalization identifies if the items that differ in both patterns are similar (line 26). Different items are considered similar if they share a minimum number of common attributes ($min\_simitem$). If items are similar, the general pattern is formed by replacing the items with their set of common attributes (line 30). The support of this general pattern G is equal to the sum of the supports of the two promising patterns. If G is frequent, it is added to F_g, the set of general patterns, by using *add_general_pattern* (line 8). If G is already part of F_g, *add_general_pattern* simply updates the support of G. The algorithm returns both sets of original frequent patterns and general frequent patterns.

Algorithm 1 G_SPM($MSDB$, min_sup, min_sup_prom, com_a)

1: $F_g \leftarrow \emptyset$
2: $F_f, F_p \leftarrow SPM(S, min\_sup, min\_sup\_prom)$
3: **for** ($i = 1$; $i \leq |F_p|$; $i++$) **do**
4: **for** ($j = i + 1$; $j \leq |F_p|$; $j++$) **do**
5: $pos \leftarrow are\_pseudo\_sim\_patt(p_i, p_j)$
6: **if** pos $\neq$ -1 **then**
7: $P_g \leftarrow general(C, p_i, p_j, pos, min\_sup, com\_a)$
8: $F_g \leftarrow add\_general\_pattern(F_g, P_g)$
9: **end if**
10: **end for**
11: **end for**
12: **return** F_f, F_g

Function are_pseudo_sim_patt(p_i, p_j)
13: $pos \leftarrow -1$
14: **if** ($length(p_i) = length(p_j)$) **then**
15: $nb\_diff \leftarrow 0$ //number of differences
16: **for** ($k = 1$; $k \leq length(p_i)$; $k++$) **do**
17: **if** ($p_i[k]! = p_j[k]$) **then**
18: $nb\_diff \leftarrow nb\_diff + 1$
19: $pos \leftarrow k$
20: **end if**
21: **end for**
22: **if** $nb\_diff > 1$ **then**
23: $pos \leftarrow -1$
24: **end if**
25: **end if**
26: **return** pos

Function general(C,p_i,p_j,pos,min_sup,$min\_simitem$)
27: $G \leftarrow \emptyset$
28: $r \leftarrow sim(att(p_i[pos]), att(p_j[pos]))$
29: **if** ($r \geq min\_simitem$) **then**
30: $s \leftarrow set\_comm\_att(att(p_i[pos]), att(p_j[pos]))$
31: $G \leftarrow p_i$
32: $Replace(G, pos, s)$
33: **if** ($sup(p_i) + sup(p_j) \geq min\_sup$) **then**
34: **return** G
35: **else**
36: **return** $\emptyset$
37: **end if**
38: **end if**

IV. Experiments

In this section, we discuss the experimental evaluations of G_SPM. The experiments are run on a 64-bit Intel Core 2.60GHz CPU laptop, 7.7 GB RAM and Ubuntu 16.04 LTS operating system.

A. Data

The dataset used is a subset of the music corpus from [8] and proposed by the *Deezer* company[1]. It consists of two data sources. The first source provides behavioral data representing users' consultations of soundtracks on *Deezer* website. It consists of sequences of soundtrack ids. This source is considered as the main source. The second source contains descriptive data about the soundtracks consisting of six descriptive attributes for each soundtrack. It is considered as the complementary source.

[1]https://www.deezer.com/

# Sequences	2,220
# Avg. sequence length	49
# Consultations	108,596
# Artists	8,959
# Soundtracks	34,080
Period	5th-8th Dec. 2016

TABLE I
CHARACTERISTICS OF THE MAIN SOURCE

The characteristics of the main source are displayed in Table I. The six descriptive attributes from the complementary

source are the following. *Duration* represents the soundtrack duration in seconds. *Artist ranking* represents the ranking of the soundtrack's artist on Deezer (from 0 to 1,000,000). *Acousticness* is the soundtrack's acoustic level, *energy* is the soundtrack's activity level, *danceability* represents how much the soundtrack is suitable for dancing and *speechiness* represents the importance of the soundtrack's lyrics. These last four attributes take the form of a number ranging in the interval $[0, 1]$.

The mining of original patterns in G_SPM, as well as in the naive approach are performed by SPAM algorithm [3], with its implemented version in SPMF library [5]. SPAM is proved in the literature to be efficient and faster than other algorithms on datasets with long sequences.

In line with the literature, the minimum support threshold used (min_sup) is determined according to the characteristics of the dataset: number of sequences, number of transactions and average length of the sequences. We set the value of min_sup to 40 which represents 1.8% of the total number of sequences.

B. The Naive Approach

The previously discussed naive approach is considered as the baseline and is compared to G_SPM. Recall that this approach integrates both data sources in one dataset made up of sequences of itemsets. Each itemset contains an item id and its set of descriptive attributes. The length of the sequences is noticeably increased compared to that of the sequences of the main source. Consequently, it leads to an increase in the complexity of the mining process especially as the number of descriptive attributes is significant. When running SPAM with min_sup = 40 on this dataset, the number of frequent patterns is too huge to be stored (more than 15GB), the runtime of the mining process takes 3 days and the number of generated frequent patterns is greater than 1 billion. When running SPAM algorithm on a sample of this dataset made up of 3 attributes, the number of frequent patterns reaches 55 million patterns, with a runtime of 1.5 hours. These results confirm the complexity of this approach and thus its inadequacy on such a dataset. The set of patterns is also huge and redundant. Not only many patterns contain both the item id and the item descriptions, but also many patterns are subsets of others. For example, both patterns $p' = \langle i_{20}, i_{98}, \{$*kettle, steel, 1 liter*$\}\rangle$ and $p'' = \langle i_{20}, i_{98}, \{$*kettle, steel*$\}\rangle$ can be mined, and $p'' \subset p'$. Therefore, we consider that p'' is redundant with p' as it provides no additional information, and it is more general. Therefore, an additional complexity due to the post-processing is required to obtain non-redundant patterns.

C. The G_SPM Algorithm

1) Mining Promising Patterns: G_SPM starts by mining original patterns: frequent and promising ones, from the main data source, with a traditional SPM algorithm. min_sup_prom and min_sup are used as the minimum support thresholds.

Recall that min_sup is set to 40. With SPAM, the number of mined frequent patterns is $1,280$ and the runtime is 0.368 seconds. As for the value of min_sup_prom, it impacts the number of promising patterns and the runtime. In this concern, we experimentally evaluate the impact of min_sup_prom on G_SPM. Recall that the promising patterns are those that are considered for generalization. We choose to represent min_sup_prom as a percentage of min_sup and to vary min_sup_prom from 90% to 75% of min_sup. Table II displays the number of promising patterns and the runtime of G_SPM over different values of min_sup_prom. Recall that the number of frequent patterns remains fixed. Table II confirms that the value of min_sup_prom greatly impacts the number of promising patterns, in line with the literature stating that the number of frequent patterns increases exponentially with the decrease of the minimum support. For the highest value of min_sup_prom (90% of min_sup), the number of promising patterns is almost equivalent to that of frequent patterns (90% of frequent patterns). For each decrease by 5% of min_sup_prom, the number of promising patterns increases within a range of 70% to 85%. For example, when min_sup_prom is equal to 80% of min_sup, the number of promising patterns is almost 3 times larger than that of frequent patterns. The runtime of the algorithm also increases as min_sup_prom decreases, but with a smaller degree than the number of promising patterns. For the highest value of min_sup_prom, the increase in the runtime is only 31%, whereas the number of promising patterns is almost equal to that of frequent patterns. When min_sup_prom equals 80% of min_sup, the runtime increases by only 72% (the number of promising patterns is increased by 293%).

min_sup_prom (relative to min_sup)	90% (36)	85% (34)	80% (32)	75% (30)
# promising patterns (% relative to freq. patt.)	1,135 (90%)	2,129 (166%)	3,745 (293%)	6,399 (500%)
additional runtime (s) (% add. runtime)	0.115 (+31%)	0.231 +(63%)	0.266 (+72%)	0.322 (+87%)

TABLE II
IMPACT OF min_sup_prom ON THE PROMISING PATTERNS AND RUNTIME

2) Forming General Patterns: The number of general patterns formed by G_SPM and the runtime of the algorithm are of interest in this section. They are both presented in table III. In these experiments, we set $min\_simitem$, defined in III-A3, to 0.8.

When min_sup_prom is set to 90% of min_sup, the number of general patterns (340) represents about 30% of the number of promising patterns. Let us study in details the way this set is obtained. First, we focus on the number of promising patterns that have at least one pseudo-similar pattern in the set of promising patterns, (named # pseudo-similar patterns in row 3 of table III for the sake of simplicity). 94.5% of the promising patterns have pseudo-similar patterns, which is quite high. The effective pruning occurs with the similarity between items: only 44.8% of the promising patterns have at least one similar pattern (named similar patterns in row 4). Notice that

min_sup_prom (relative to min_sup)	90% (36)	85% (34)	80% (32)	75% (30)
# promising patterns	1,135	2,129	3,745	6,399
# pseudo-similar patterns (% prom. patt.)	1,073 (94.5%)	2,026 (95.2%)	3,596 (96%)	6,183 (96.6%)
# similar patterns (% prom. patt.)	508 (44.8%)	1,189 (55.8%)	2,432 (64.9%)	4,433 (61.4%)
# general patterns (% prom. patt.)	340 (30%)	873 (41%)	1,968 (52%)	3,930 (61%)
# frequent patterns	1,620	2,153	3,248	5,210
Runtime (s) of G_SPM	124	281	769	1,508

TABLE III
IMPACT OF min_sup_prom ON THE NUMBER OF GENERAL PATTERNS AND RUNTIME

there is no pruning due to the support of the general pattern (line 31, Algorithm 1) as $min\_sup\_prom > 50\%$, so if two patterns are similar, the resulting general pattern is frequent.

Let us now focus on the impact of min_sup_prom on these values. As expected, the number of general patterns increases with the decrease of min_sup_prom; it is even more than doubled for each decrease of 5% of min_sup_prom. With $min\_sup\_prom = 75\%$, it is 3,930, three times larger than that of original frequent patterns. In addition, the number of general patterns represents from 30% to 61% of the promising patterns as min_sup_prom decreases. When focusing on the number of pseudo-similar patterns, it slightly increases. The main difference occurs here again in the identification of similar patterns. The increase in the ratio can be explained by the fact that the initial set of promising patterns is larger for lower values of min_sup_prom, so the chances of having a similar pattern among the promising ones are higher. When $min\_sup\_prom = 85\%$, the number of general patterns is close to the number of original frequent patterns. In this case, the total number of frequent patterns is almost doubled (2,153 patterns). Let us now consider the runtime of G_SPM according to min_sup_prom. As expected, it is longer than the runtime of mining the main source only (see Table II). More importantly, it is significantly lower that of the naive approach (see section IV-B). This confirms that one of the goals of G_SPM is achieved: reducing the complexity of the mining process. When considering the impact of min_sup_prom on the runtime, we can see that it increases linearly with the number of promising patterns. Although the generalization step is complex , the increase in the runtime of the algorithm remains limited. The total number of frequent original patterns and frequent general patterns is: 1,620 for $min\_sup\_prom = 90\%$ and 5,210 for $min\_sup\_prom = 75\%$. Notice that this number can be significantly increased by decreasing the value of min_sup_prom.

From the previous experiments, we can consider that the most adequate value of min_sup_prom for this dataset is $min\_sup\_prom = 85\%$ of min_sup as it results in a significant but limited number of frequent general patterns relative to the frequent patterns generated by SPM process (about 70% of additional patterns), and it has a limited runtime (around 4 minutes). At last, we can see that the set of patterns mined by G_SPM is much smaller than that mined by the naive approach as it is not redundant. This confirms that the second goal of G_SPM is achieved: limiting the number of mined patterns.

V. CONCLUSION AND FUTURE WORK

This paper aimed at handling the problem of item similarity in behavioral pattern mining. We proposed G_SPM, an algorithm that takes advantage of multi-source data to generate frequent general patterns by adopting a strategy of selective mining of sources. This strategy limits the complexity of the mining process and the redundancy among the mined patterns. The conducted experiments confirm that this strategy significantly decreases the runtime compared to the naive approach and that general patterns are actually formed.

Initial choices have been made, and they will be questioned in future works. For example, the similarity between two patterns can only be performed on patterns of the same length, and two candidate patterns can only differ by one item. Obviously, we can identify more candidates by adopting more advanced similarity measures and by considering more item differences between patterns, especially for long ones.

ACKNOWLEDGMENT

This work has been funded by the PIA2 e-FRAN METAL Project (2016-2021).

REFERENCES

[1] C. C. Aggarwal. *Data mining: the textbook*. Springer, 2015.
[2] M Aldekhail. Application and significance of web usage mining in the 21st century: a literature review. *Int. Journal of Computer Theory and Engineering*, 8(1):41, 2016.
[3] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using a bitmap representation. In *8th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining*, pages 429–435, 2002.
[4] S. Džeroski. Multi-relational data mining: an introduction. *ACM SIGKDD Explorations Newsletter*, 5(1):1–16, 2003.
[5] P. Fournier-Viger, J. C. Lin, A. Gomariz, T. Gueniche, A. Soltani, Z. Deng, and H.T. Lam. The spmf open-source data mining library version 2. In *ECML-PKDD*, pages 36–40. Springer, 2016.
[6] P. Fournier-Viger, J.C. Lin, R.U. Kiran, Y.S. Koh, and R. Thomas. A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1(1):54–77, 2017.
[7] J. Han, J. Pei, and M. Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.
[8] A. l'Huillier. *Modéliser la diversité au cours du temps pour comprendre le contexte de l'utilisateur dans les systèmes de recommandation*. PhD thesis, Université de Lorraine, 2018.
[9] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *17th ICDE*, pages 215–224. IEEE, 2001.
[10] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, and U. Dayal. Multi-dimensional sequential pattern mining. In *CIKM*, pages 81–88, 2001.
[11] O. Raphaeli, A. Goldstein, and L. Fink. Analyzing online consumer behavior in mobile and pc devices: A novel web usage mining approach. *Electronic commerce research and applications*, 26:1–12, 2017.
[12] A. Simanovsky and A. Ulanov. Mining text patterns for synonyms extraction. In *2011 22nd Int. Workshop on Database and Expert Systems Applications*, pages 473–477. IEEE, 2011.
[13] S. Zhang, X. You, Z. Jin, and X. Wu. Mining globally interesting patterns from multiple databases using kernel estimation. *Expert Systems with Applications*, 36(8):10863–10869, 2009.
[14] Radosław Ziembiński. Algorithms for context based sequential pattern mining. *Fundamenta Informaticae*, 76(4):495–510, 2007.