



## Contextual Sequential Pattern Mining

Julien Rabatel\*<sup>†</sup>, Sandra Bringay<sup>†‡</sup>, Pascal Poncelet<sup>†</sup>  
\*Tecnalia

Cap Omega, Rd-Pt B. Franklin, 34960 Montpellier Cedex 2, France

<sup>†</sup>LIRMM (CNRS UMR 5506), Univ. Montpellier 2

161 rue Ada, 34095 Montpellier Cedex 5, France

Email: name@lirmm.fr

<sup>‡</sup>Dpt MIAp, Univ. Montpellier 3

Route de Mende, 34199 Montpellier Cedex 5, France

**Abstract**—Traditional sequential patterns do not take into account additional contextual information since patterns extracted from data are usually general. By considering the fact that a pattern is associated with one specific context the decision expert can then adapt his strategy considering the type of customers. In this paper we propose to mine more precise patterns of the form “*young users buy products A and B then product C, while old users do not follow this same behavior*”. By highlighting relevant properties of such contexts, we show how contextual sequential patterns can be extracted by mining the database in a concise manner. We conduct our experimental evaluation on real-world data and demonstrate performance issues.

**Keywords**—Sequential Patterns; Contextual Data;

### I. INTRODUCTION

Sequential pattern mining is an important problem widely addressed by the data mining community, with a very large field of applications such as user behavior, sensor data, DNA analysis, web log analysis, etc. Sequential pattern mining aims to extract sets of items commonly associated over time. For instance, when studying purchases of customers in a supermarket, a sequential pattern could be “*many customers buy products A and B, then buy later on product C*”. However, data are very often provided with additional information about purchases, such as for example customer’s age or gender. Traditional sequential patterns do not take into account this information since the patterns extracted from data are usually general. Having a better knowledge about the particularity of objects supporting a given behavior can help decision making. By considering the fact that a pattern is associated with one specific context the decision expert can then adapt his strategy considering the type of customers. For instance, it could be interesting for experts to obtain much more informative knowledge by considering contextual data, in order to answer questions such as “*What are behaviors specific to young customers?*”, “*Is there a certain behavior specific to young women?*” or “*What is the most general behavior, not depending on contextual information?*”. In this paper we propose to mine more precise patterns of the form “*young users buy products A*

*and B then product C, while old users do not follow this same behavior*”.

Mining such contextual sequential patterns is a difficult task. Different contexts should be mined independently and then tested if frequent patterns are shared with other contexts. Some contexts can be more or less general. For instance, the context corresponding to *young* customers is more general than the context corresponding to *young male* customers. Hence, all (more or less general) contexts have to be considered (taking into consideration the generalization / specialization hierarchies), and the mining process can be very time consuming.

The problem of finding more or less general contexts where one sequential pattern is specific, while handling a context hierarchy, has not been investigated yet. Indeed, multidimensional sequential pattern mining [9] only partially addresses this problem since patterns specific to a particular context will not appear if this context is not in itself frequent. Another related field, the mining of emergent patterns [6] only considers sequences that are specific to one class regardless of the level of generalization / specialization.

The contribution of the paper is twofold. We first formally describe contexts. Then, by highlighting relevant properties of such contexts, we show how contextual sequential patterns can be extracted by only mining the whole database once. We conduct experimental evaluation on real-world data and demonstrate performance issues.

The rest of the paper is organized as follows. In Section II, we define the “traditional” pattern mining problem and show why it is not sufficient for handling contextual data. Section III presents how contextual information can be considered in order to extract relevant patterns. Algorithms are described in Section IV. In Section V experiments are presented. We conclude and discuss future work perspectives in Section VII.

### II. MOTIVATION

In this section, we describe the traditional sequential pattern mining problem and highlight the need for a more specific way to handle contextual information.

### A. Traditional Sequential Pattern Mining Problem

Sequential patterns were introduced in [2] and can be considered as an extension of the concept of frequent itemset [1] by handling timestamps associated to items. Sequential pattern mining aims to extract sets of items commonly associated over time. In the “basket market” concern, a sequential pattern can be for example: “40 % of the customers buy a television, then buy later on a DVD player”. The problem of mining all sequential patterns in a sequence database is defined as follows.

Let  $\mathcal{X}$  be a set of distinct **items**. An **itemset** is a subset of items, denoted by  $I = (i_1 i_2 \dots i_n)$ , i.e., for  $1 \leq j \leq n$ ,  $i_j \in \mathcal{X}$ . A **sequence** is an ordered list of itemsets. A sequence  $s$  is denoted by  $\langle I_1 I_2 \dots I_k \rangle$ , where  $I_i \subseteq \mathcal{X}$  for  $1 \leq i \leq k$ .

Let  $s = \langle I_1 I_2 \dots I_m \rangle$  and  $s' = \langle I'_1 I'_2 \dots I'_n \rangle$  two sequences. The sequence  $s$  is a **subsequence** of  $s'$ , denoted by  $s \sqsubseteq s'$ , if  $\exists i_1, i_2, \dots, i_m$  with  $1 \leq i_1 < i_2 < \dots < i_m \leq n$  such that  $I_1 \subseteq I'_{i_1}$ ,  $I_2 \subseteq I'_{i_2}$ , ...,  $I_m \subseteq I'_{i_m}$ . If  $s \sqsubseteq s'$  we also say that  $s'$  **supports**  $s$ .

A **sequence database**  $\mathcal{B}$  is a relation  $\mathcal{R}(ID, S)$ , where an element  $id \in dom(ID)$  is a sequence identifier, and  $dom(S)$  is a set of sequences. The **size** of  $\mathcal{B}$ , denoted by  $|\mathcal{B}|$ , is the number of tuples in  $\mathcal{B}$ . A tuple  $\langle id, s \rangle$  is said to **support** a sequence  $\alpha$  if  $\alpha$  is a subsequence of  $s$ , i.e.,  $\alpha \sqsubseteq s$ . The **support** of a sequence  $\alpha$  in the sequence database  $\mathcal{B}$  is the number of tuples in  $\mathcal{B}$  supporting  $\alpha$ , i.e.,  $sup_{\mathcal{B}}(\alpha) = |\{ \langle id, s \rangle \mid \langle id, s \rangle \in \mathcal{B} \wedge (\alpha \sqsubseteq s) \}|$ . Given a real  $minSupp$  such that  $0 < minSupp \leq 1$  as the **minimum support threshold**, a sequence  $\alpha$  is a **frequent** in the sequence database  $\mathcal{B}$  if the proportion of tuples in  $\mathcal{B}$  supporting  $\alpha$  in  $\mathcal{B}$  is greater than  $minSupp$ , i.e.,  $sup_{\mathcal{B}}(\alpha) \geq minSupp \cdot |\mathcal{B}|$ . Sequence  $\alpha$  is also said to be a **sequential pattern** in  $\mathcal{B}$ .

| id       | Age   | Gender | Sequence                       |
|----------|-------|--------|--------------------------------|
| $s_1$    | young | male   | $\langle (ad)(b) \rangle$      |
| $s_2$    | young | male   | $\langle (ab)(b) \rangle$      |
| $s_3$    | young | male   | $\langle (a)(a)(b) \rangle$    |
| $s_4$    | young | male   | $\langle (c)(a)(bc) \rangle$   |
| $s_5$    | young | male   | $\langle (d)(ab)(bcd) \rangle$ |
| $s_6$    | young | female | $\langle (b)(a) \rangle$       |
| $s_7$    | young | female | $\langle (a)(b)(a) \rangle$    |
| $s_8$    | young | female | $\langle (d)(a)(bc) \rangle$   |
| $s_9$    | old   | male   | $\langle (ab)(a)(bd) \rangle$  |
| $s_{10}$ | old   | male   | $\langle (bcd) \rangle$        |
| $s_{11}$ | old   | male   | $\langle (bd)(a) \rangle$      |
| $s_{12}$ | old   | female | $\langle (e)(bcd)(a) \rangle$  |
| $s_{13}$ | old   | female | $\langle (bde) \rangle$        |
| $s_{14}$ | old   | female | $\langle (b)(a)(e) \rangle$    |

Table 1  
A CONTEXTUAL SEQUENCE DATABASE.

**Example 1:** Table 1 shows a sequence database describing the purchases of customers in a shop. The first column stands for the identifier of each sequence given in the last column.  $a, b, c, d, e$  are products. Column Gender and Age are extra

information about sequences. They are not considered in traditional sequential pattern mining. The size of  $\mathcal{B}$  is  $|\mathcal{B}| = 14$ .

The first sequence in Table 1 describes the sequence of purchases made by a customer identified by  $s_1$ : he has purchased products  $a$  and  $d$ , then product  $b$  later on.

In the following, we set the minimum support  $minSupp$  to 0.5. Let consider the sequence  $s = \langle (a)(b) \rangle$ . Its support in  $\mathcal{B}$  is  $sup_{\mathcal{B}}(s) = 8$ . So,  $sup_{\mathcal{B}}(s) \geq minSupp \cdot |\mathcal{B}|$ , and  $s$  is a sequential pattern in  $\mathcal{B}$ .

### B. Why Taking into Account Additional Information?

Considering the previous example, available contextual information are the age and the gender of customers, and a context could be *young female* or *old customer* (for any gender).

Traditional sequential pattern mining (SPM) within such data comes with the following drawbacks.

1. **Some context-dependent behaviors are wrongly considered as general, although they are frequent in only one subcategory of customers.** For instance,  $s = \langle (a)(b) \rangle$  is a sequential pattern in  $\mathcal{B}$ . However, by studying carefully the sequence database, we can easily see that it is much more specific to *young* persons. Indeed, 7 of 8 *young* customers support this sequence, while only 1 of 6 *old* customers follow this pattern.
2. **Some context-dependent behaviors are not considered, although they are frequent in a subcategory of customers.** For instance,  $s' = \langle (bd) \rangle$  which is not a sequential pattern in  $\mathcal{B}$  (6 customers of 14 support it) is however supported by 5 of 6 *old* customers.

Previous cases show that traditional SPM is very sensitive to the repartition of data sequences according to contexts. Sequences  $s$  and  $s'$  are both highly related to the *age* of customers. However,  $s$  is frequent and  $s'$  is not, because of the larger proportion of *young* customers in the whole sequence database.

By studying the sequence database more carefully, we can highlight several interesting cases:

- $\langle (b) \rangle$  is a sequential pattern in  $\mathcal{B}$  and does not depend on contextual information, as it occurs in all data sequences.
- $\langle (bd) \rangle$  is not a sequential pattern in  $\mathcal{B}$  but is specific to *old* customers, without considering their *gender*. This sequence depends only on the *age* of customers.
- $\langle (e) \rangle$  is not a sequential pattern in  $\mathcal{B}$  but is specific to *old female* customers, as it occurs in 3 of 3 such customers, and never occurs in the rest of the sequence database. It depends both on the *age* and the *gender* of customers.

These examples show that traditional SPM is not relevant when behaviors depend on contextual information associated

with data sequences. We describe in the following how contextual information are formally handled to extract much more informative knowledge through contextual sequential pattern mining.

### III. CONTEXTUAL SEQUENTIAL PATTERNS

Here we propose a formal description of contextual data, and define the notion of contextual sequential pattern.

#### A. Contextual Sequence Database

We define a **contextual sequence database**  $\mathcal{CB}$  as a relation  $\mathcal{R}(ID, S, D_1, \dots, D_n)$ , where  $dom(S)$  is a set of sequences and  $dom(D_i)$  for  $1 \leq i \leq n$  is the set of all possible values for  $D_i$ .  $D_1, D_2, \dots, D_n$  are called the **contextual dimensions** in  $\mathcal{CB}$ . A tuple  $u \in \mathcal{CB}$  is denoted by  $\langle id, s, d_1, \dots, d_n \rangle$ .

Values on contextual dimensions can be organized as hierarchies. For  $1 \leq i \leq n$ ,  $dom(D_i)$  can be extended to  $\mathcal{H}(D_i)$  where  $\subseteq_{D_i}$  is a partial order on  $\mathcal{H}(D_i)$  such that  $dom(D_i)$  is the set of minimal elements of  $\mathcal{H}(D_i)$ .

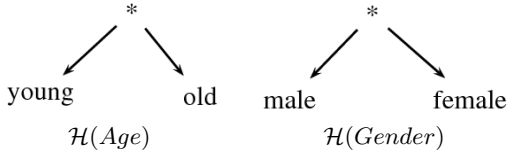


Figure 1. Hierarchies on Age and Gender dimensions.

**Example 2:** We consider hierarchies on dimensions Age and Gender given in Figure 1. In this example,  $\mathcal{H}(Age) = dom(Age) \cup \{*\}$ , where  $young \subseteq_{Age} *$  and  $old \subseteq_{Age} *$ . Similarly,  $\mathcal{H}(Gender) = dom(Gender) \cup \{*\}$ , where  $male \subseteq_{Gender} *$  and  $female \subseteq_{Gender} *$ .

A **context**  $c$  in  $\mathcal{CB}$  is denoted by  $[d_1, \dots, d_n]$  where  $d_i \in \mathcal{H}(D_i)$ . If for  $1 \leq i \leq n$ ,  $d_i \in dom(D_i)$ , then  $c$  is called a **minimal context**.

Let  $c_1$  and  $c_2$  be two contexts in  $\mathcal{CB}$ , such that  $c_1 = [d_1^1, \dots, d_n^1]$  and  $c_2 = [d_1^2, \dots, d_n^2]$ . Then  $c_1 \geq c_2$  iff  $\forall i$  with  $1 \leq i \leq n$ ,  $d_i^1 \supseteq_i d_i^2$ . Moreover, if  $\exists i$  with  $1 \leq i \leq n$  such that  $d_i^1 \supseteq_i d_i^2$ , then  $c_1 > c_2$ . In this case,  $c_1$  is said then to be **more general** than  $c_2$ , and  $c_2$  is **more specific** than  $c_1$ .

**Example 3:** In data from Table I, there are four minimal contexts:  $[y, m]$ ,  $[y, f]$ ,  $[o, m]$ , and  $[o, f]$ , where  $y$  and  $o$  respectively stand for young and old, and  $m$  and  $f$  respectively stand for male and female. In addition, context  $[*, *]$  is more general than  $[y, *]$ . On the other hand,  $[y, *]$  and  $[*, m]$  are incomparable.

The set of all contexts associated with the partial order  $>$  is called the context hierarchy and denoted by  $\mathcal{H}$ . Given two contexts  $c_1$  and  $c_2$  such that  $c_1 > c_2$ ,  $c_1$  is called an **ancestor** of  $c_2$ , and  $c_2$  is a **descendant** of  $c_1$ .

For instance, Figure 2 shows a visual representation of  $\mathcal{H}$  for data provided in Table I and hierarchies previously given for dimensions Age and Gender.

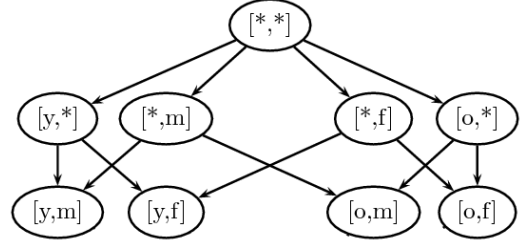


Figure 2. The context hierarchy.

We can now consider the tuples of  $\mathcal{CB}$  according to contexts defined above. Let  $u = \langle id, s, d_1, \dots, d_n \rangle$  be a tuple in  $\mathcal{CB}$ . The context  $c$  such that  $c = [d_1, \dots, d_n]$  is called the **context of  $u$** . Note that the context of  $u$  is minimal, since  $\forall i$  with  $1 \leq i \leq n$ ,  $d_i \in dom(D_i)$ .

Let  $u$  be a tuple in  $\mathcal{CB}$  and  $c$  the context of  $u$ . For all contexts  $c'$  such that  $c' \geq c$  we say that  $c'$  **contains**  $u$  (and  $u$  is contained by  $c'$ ).

Let  $c = [d_1, \dots, d_n]$  be a context (not necessarily minimal) in  $\mathcal{CB}$ , and  $\mathcal{U}$  the set of tuples contained by  $c$ . The **sequence database of  $c$** , denoted by  $\mathcal{B}(c)$ , is the set of tuples  $\langle id, s \rangle$ , such that  $\exists u \in \mathcal{U}$  with  $u = \langle id, s, d_1, \dots, d_n \rangle$ . We define the **size of a context  $c$** , denoted by  $|c|$ , as the size of its sequence database, i.e.,  $|c| = |\mathcal{B}(c)|$ .

**Example 4:** In Table I, let consider contexts  $[o, m]$  and  $[o, *]$ . Then  $\mathcal{B}([o, m]) = \{s_9, s_{10}, s_{11}\}$  and  $\mathcal{B}([o, *]) = \{s_9, s_{10}, s_{11}, s_{12}, s_{13}, s_{14}\}$ .

Let  $c$  be a context in  $\mathcal{CB}$ . The **decomposition** of  $c$  in  $\mathcal{CB}$ , denoted by  $decomp(c)$ , is the nonempty set of minimal contexts  $c'$  such that  $c \geq c'$ .

**Example 5:** The decomposition of context  $[y, *]$  is  $\{[y, m], [y, f]\}$ .

**Lemma 1:** Let  $c$  be a context,  $decomp(c) = \{c_1, c_2, \dots, c_n\}$  its decomposition and  $s$  a sequence. Given the definition of the sequence database of  $c$ , the decomposition of  $c$  has the following properties:

- 1)  $\bigcap_{i=1}^n \mathcal{B}(c_i) = \emptyset$ ;
- 2)  $\bigcup_{i=1}^n \mathcal{B}(c_i) = \mathcal{B}(c)$ ;
- 3)  $|c| = |\mathcal{B}(c)| = \sum_{i=1}^n |c_i|$ .

#### B. Contextual Sequential Patterns

We saw in the previous section how a contextual sequence database can be divided into several sequence databases according to contexts. Now, let us consider the definition

of frequent sequences in such contexts. Let  $c$  be a context, and  $s$  a sequence.

**Definition 1:**  $s$  is a **frequent in  $c$**  iff  $s$  is frequent in  $\mathcal{B}(c)$ , i.e., if  $\text{sup}_{\mathcal{B}(c)}(s) \geq \text{minSup} \cdot |c|$ . We also say that  $s$  is a **sequential pattern** in  $c$ . In the following, we will denote  $\text{sup}_{\mathcal{B}(c)}(s)$  as  $\text{sup}_c(s)$ .

As seen in Section II, we are interested in mining sequential patterns being specific to a particular context, according to the following definition.

**Definition 2:** Sequence  $s$  is **specific to  $c$**  ( $c$ -specific) iff:

- 1)  $s$  is frequent in  $c$ .
- 2)  $s$  is **general in  $c$** , i.e.,  $s$  is frequent in all  $c$ 's descendants in the context hierarchy. In this case,  $s$  is said to be  $c$ -general.
- 3) there does not exist a context  $c'$ , such that  $c' > c$  and  $s$  is  $c'$ -general.

**Definition 3:** A **contextual sequential pattern** is a couple  $(c, s)$ , such that  $s$  is  $c$ -specific. Such a contextual sequential pattern is said to be **generated by  $s$** .

| sequence   | $[y, m]$   | $[y, f]$   | $[o, m]$   | $[o, f]$   |
|--|------------|------------|------------|------------|
| $\langle\langle a \rangle\rangle$                                | <b>5/5</b> | <b>3/3</b> | <b>2/3</b> | <b>2/3</b> |
| $\langle\langle b \rangle\rangle$                                | <b>5/5</b> | <b>3/3</b> | <b>3/3</b> | <b>3/3</b> |
| $\langle\langle d \rangle\rangle$                                | 2/5        | 1/3        | <b>3/3</b> | <b>2/3</b> |
| $\langle\langle e \rangle\rangle$                                | 0/5        | 0/3        | 0/3        | <b>3/3</b> |
| $\langle\langle a \rangle\rangle\langle\langle b \rangle\rangle$ | <b>5/5</b> | <b>2/3</b> | 1/3        | 0/3        |
| $\langle\langle b \rangle\rangle\langle\langle a \rangle\rangle$ | 0/5        | <b>2/3</b> | <b>2/3</b> | <b>2/3</b> |
| $\langle\langle bd \rangle\rangle$                               | 1/5        | 0/3        | <b>3/3</b> | <b>2/3</b> |

Table II  
SEQUENTIAL PATTERNS IN MINIMAL CONTEXTS OF  $\mathcal{CB}$ .

**Example 6:** Table II shows, from the contextual sequence database provided in Table I, the sequences being frequent in at least one minimal context as well as their support for each minimal context (of the form  $\text{sup}_c(s)/|\mathcal{B}(c)|$ ). When the support is displayed in bold, then the sequence is frequent in the corresponding minimal context.

For instance,  $s = \langle\langle a \rangle\rangle\langle\langle b \rangle\rangle$  is frequent in  $[y, *]$  (it is supported by 7 young customers of 8) and in its descendants  $[y, m]$  and  $[y, f]$ , i.e.,  $s$  is  $[y, *]$ -general. Moreover,  $s$  is not  $[*, *]$ -general, as there exist descendants of  $[*, *]$  where  $s$  is not frequent. In consequence,  $s$  is  $[y, *]$ -specific, and  $([y, *], \langle\langle a \rangle\rangle\langle\langle b \rangle\rangle)$  is a contextual sequential pattern.

### C. Contextual Sequential Pattern Mining

The problem of mining contextual sequential patterns can be defined as extracting all contextual sequential patterns from a contextual sequence database. Relying on previous definitions, a naive idea would be to mine independently each context, in order to obtain corresponding sequential patterns, and then filter those patterns being specific to this context. However, this would imply the two following issues.

- **Contexts can be very numerous.** Indeed, the number of contexts is  $\prod_{i=1}^n |\mathcal{H}(D_i)|$ , where  $D_1, \dots, D_n$  are the contextual dimensions. In comparison, the number of minimal contexts is  $\prod_{i=1}^n |\text{dom}(D_i)|$ .
- **Filtering sequential patterns having the right properties is very costly** as it requires to test for each mined sequential pattern its presence in a large number of contexts, in order to verify whether it is specific to one context or not.

In order to overcome these drawbacks, we propose to mine the whole sequence database in a single process to extract contextual sequential patterns. To this end, we exploit the properties of the context hierarchy in order to show that contextual sequential patterns can be extracted by mining only frequent sequences in minimal contexts, and generating the corresponding contextual sequential patterns.

According to the properties of the decomposition of a context, we can deduce the following lemma.

**Lemma 2:** Let  $c$  be a context, and  $\text{decomp}(c) = \{c_1, c_2, \dots, c_n\}$ . If  $\forall i \in 1, \dots, n$ ,  $s$  is frequent in  $c_i$  (resp. is not frequent), then  $s$  is frequent in  $c$  (resp. is not frequent).

In addition,  $s$  is frequent (resp. is not frequent) in all the descendants of  $c$ .

**Proof:** For each  $c_i$  such that  $i \in \{1, \dots, n\}$ ,  $\text{sup}_{c_i}(s) \geq \text{minSup} \cdot |c_i|$ . This means  $\sum_{i=1}^k \text{sup}_{c_i}(s) \geq \sum_{i=1}^n \text{minSup} \cdot |c_i|$ .

However,  $\sum_{i=1}^n \text{minSup} \cdot |c_i| = \text{minSup} \cdot \sum_{i=1}^n |c_i| = \text{minSup} \cdot |c|$ . Since  $\sum_{i=1}^k \text{sup}_{c_i}(s) = \text{sup}_c(s)$  it follows  $\text{sup}_c(s) \geq \text{minSup} \cdot |c|$ .

Let  $c'$  be a context such that  $c > c'$ . Then  $\text{decomp}(c') \subseteq \text{decomp}(c)$ , i.e.,  $s$  is a frequent in all contexts in  $\text{decomp}(c')$ . Applying the previous result we obtain  $s$  a frequent sequence in  $c'$ .

A similar reasoning is applied if  $s$  is unfrequent in all contexts of the decomposition of  $c$ . ■

Lemma 2 is a key result as it allows us to redefine the notion of  $c$ -specificity, using only the decomposition of  $c$  and  $\mathcal{F}$  the set of minimal contexts where  $s$  is frequent.

**Lemma 3:** The sequence  $s$  is  $c$ -specific iff:

- $\text{decomp}(c) \subseteq \mathcal{F}$ ,
- there does not exist a context  $c'$  such that  $c' > c$  and  $\text{decomp}(c') \subseteq \mathcal{F}$ .

**Proof:** If  $\text{decomp}(c) \subseteq \mathcal{F}$ , then  $s$  is frequent in each element of  $\text{decomp}(c)$ . According to lemma 2, it follows that  $s$  is frequent in  $c$  and its descendants, i.e.,  $s$  is  $c$ -general. Moreover, there does not exist a context  $c'$  such that  $c' > c$  and  $\text{decomp}(c') \subseteq \mathcal{F}$ . In consequence  $\forall c'$  such that  $c' > c$ ,  $s$  is not  $c'$ -general. In conclusion,  $s$  is  $c$ -specific. ■

The set of contexts that meet the properties given in lemma 3 is called the **coverage** of  $\mathcal{F}$  in the context hierarchy, and denoted by  $cov(\mathcal{F})$ . We show in Section IV how the coverage of  $\mathcal{F}$  can be easily computed to generate the contextual sequential patterns.

**Example 7:** Let  $s = \langle(b)(a)\rangle$  be a sequence and  $\mathcal{F}_s = \{[y, f], [o, m], [o, f]\}$  the set of minimal contexts where  $s$  is frequent. Then,  $cov(\mathcal{F}_s) = \{[*], [f], [o, *]\}$ , i.e.,  $([*], [f], \langle(b)(a)\rangle)$  and  $([o, *], \langle(b)(a)\rangle)$  are the contextual sequential patterns generated by  $\langle(b)(a)\rangle$ .

**Theorem 1:** Let  $S$  be the set of sequences that are frequent in at least one minimal context. The set of contextual sequential patterns in a contextual sequence database  $\mathcal{CB}$  is the set of all couples  $(c, s)$ , where  $s \in S$ , and  $(c, s)$  is generated by  $s$ .

**Proof:** This result is an immediate consequence of the definition of a  $c$ -specific sequence. Indeed, if  $s$  is unfrequent in each minimal context, i.e.,  $\mathcal{F} = \emptyset$ , then it is unfrequent in each element of the context hierarchy (see Lemma 2 and there does not exist a context  $c$  such that  $s$  is  $c$ -specific. In conclusion, a contextual sequential pattern is necessarily generated by a sequence being frequent in at least one minimal context. ■

Theorem 1 is essential in the contextual SPM problem. Indeed, they ensure that contextual sequential patterns in a contextual sequence database can be deduced from the set of sequences that are frequent in at least one minimal context. Hence, it is not required to mine sequential patterns in each context of the hierarchy. In addition, we show in Section IV that extracting sequences that are sequential patterns in at least one minimal context can be performed in one single process, instead of mining each minimal context separately.

#### IV. ALGORITHM

In this section, we describe the Contextual Sequential Pattern Mining algorithm. This algorithm is based on PrefixSpan [8], an efficient method to solve the traditional sequential pattern mining problem.

##### A. PrefixSpan

The principle of PrefixSpan is explained in the following example, by describing the process of mining sequential patterns in the sequence database  $\mathcal{B}$  from Table I, with a minimum support threshold set to 0.5.

**Example 8:** A scan of the sequence database extracts all the sequential patterns of the form  $\langle(i)\rangle$ , where  $i$  is an item. Hence, PrefixSpan finds  $\langle(a)\rangle$ ,  $\langle(b)\rangle$ ,  $\langle(d)\rangle$ , since  $\langle(c)\rangle$  and  $\langle(e)\rangle$  are not frequent.

In consequence, the whole set of sequential patterns in  $\mathcal{B}$  can be partitioned into subsets, each subset being the set of sequential patterns having  $\langle(i)\rangle$  as a prefix. These subsets

can be extracted by mining the **projected databases** for each prefix, i.e., for each  $\langle(i)\rangle$ . A projected database contains, for each data sequence, its subsequence containing all frequent items following the first occurrence of the given prefix. Such a subsequence is called a **postfix**. If the first item  $x$  of the postfix is in the same itemset as the last item of the prefix, the postfix is denoted by  $\langle(\_x\_\dots)\_$ .

Then,  $\langle(a)\rangle$  is outputted, and the  $\langle(a)\rangle$ -projected database is built, containing 11 postfixes:  $\langle(\_d)(b)\rangle$ ,  $\langle(\_b)(b)\rangle$ ,  $\langle(a)(b)\rangle$ ,  $\langle(bc)\rangle$ , etc. Then items  $i$ , such that either  $\langle(bi)\rangle$  or  $\langle(b)(i)\rangle$  is frequent, are extracted from the  $\langle(a)\rangle$ -projected database.  $b$  is such an item, as  $\langle(a)(b)\rangle$  is a sequential pattern. So, the process continues by outputting  $\langle(a)(b)\rangle$ , and using it as a new prefix.

##### B. Contextual Sequential Pattern Mining

In this section, we present the algorithm for extracting contextual sequential patterns. Our approach can be decomposed into two main steps.

- 1) From a contextual sequence database, we extract all sequences that are frequent in at least one minimal context. As explained in Section III, all contextual sequential patterns can be generated from this set of sequences;
- 2) From the set of sequences obtained in Step 1, we generate the set of contextual sequential patterns.

The above steps are explained in Algorithm 1, which makes use of Algorithm 2. From a contextual sequence database  $\mathcal{CB}$ , a minimum support threshold  $minSup$  and a context hierarchy  $\mathcal{H}$ , the algorithm outputs the contextual sequential patterns from  $\mathcal{CB}$ .

##### 1) Extracting sequential patterns in minimal contexts:

The first objective of the algorithm is to extract frequent sequences in minimal contexts, and for each of them, the corresponding set of minimal contexts where it is frequent. This part is performed using the principle of PrefixSpan: from a prefix sequence  $s$ , the algorithm builds the  $s$ -projected database (method *BuildProjectedDatabase*), and scans the projected database (method *ScanDB*) to find items  $i$  that can be assembled to form a new sequential pattern  $s'$ . Then, the  $s'$ -projected database is built and the process continues. Since the general idea is close to PrefixSpan, we do not detail the *ScanDB* and *BuildProjectedDatabase* methods, but only focus on the differences with the original ones.

- *ScanDB*( $\mathcal{B}$ ): the main difference consists of the fact that the support of  $i$  is computed in each minimal context of the  $s$ -projected database. Hence, this method returns the set of couples  $(i, \mathcal{F}_i)$ , where  $i$  is an item and  $\mathcal{F}_i$  is the nonempty set of minimal contexts where  $i$  is frequent.
- *BuildProjectedDatabase*( $s, \mathcal{F}_i, \mathcal{CB}$ ): the construction of the  $s$ -projected database differs from PrefixSpan since only sequences contained in contexts of  $\mathcal{F}_i$  are considered for constructing the new sequence database.

In other words, if  $s$  is not frequent in a context  $c$ , then sequences contained in  $c$  will not be considered for generating new sequential patterns.

---

**Algorithm 1** Contextual SPM
 

---

**Require:**  $\mathcal{CB}$  a contextual sequence database,  $minSup$  a minimum support threshold,  $\mathcal{H}$  a context hierarchy.  
Call  $subContextualSPM(\langle \rangle, \mathcal{CB}, \mathcal{H})$ ;

**Subroutine**  $subContextualSPM(s, \mathcal{CB}, \mathcal{H})$

**Require:**  $s$  a sequence;  $\mathcal{CB}$  the  $s$ -projected database,  $\mathcal{H}$  a context hierarchy.

$I = ScanDB(\mathcal{CB})$ ;

**for all** couple  $(i, \mathcal{F}_i)$  in  $I$  **do**

$s'$  is the sequence such that  $i$  is appended to  $s$ ;

*/\* Contextual sequential pattern generation\*/*

$C = Coverage(\mathcal{F}_i, \mathcal{H})$

**for all**  $c$  in  $C$  **do**

output  $(c, s')$ ;

**end for**

$\mathcal{CB}' = BuildProjectedDatabase(s', \mathcal{F}_i, \mathcal{CB})$ ;

call  $subContextualSPM(s', \mathcal{CB}', \mathcal{H})$ ;

**end for**

---

2) *Generating contextual sequential patterns:* A sequential pattern  $s$  is extracted with the set of minimal contexts where it is frequent. From this set, we generate the contextual sequential patterns generated by  $s$ , i.e., the set of  $(c, s)$  where  $c$  is a context such that  $s$  is  $c$ -specific. This is performed by  $Coverage(\mathcal{F}, \mathcal{H})$  described in Algorithm 2. This algorithm is based on a bottom-up scan of the context hierarchy (i.e., from the leaves to the root), in order to collect the most general contexts being subsets of  $\mathcal{F}$ , i.e., where a sequence is specific (see Section III).

---

**Algorithm 2** Coverage( $\mathcal{F}, \mathcal{H}$ )
 

---

**Require:**  $\mathcal{F}$  a set of minimal contexts,  $\mathcal{H}$  a context hierarchy.

Let  $C = \emptyset$

Let  $\mathcal{L}$  be the set of leaves in  $\mathcal{H}$

**for all**  $l \in \mathcal{L}$  **do**

$C = C \cup subCoverage(l, \mathcal{F}, \mathcal{H})$ ;

**end for**

**return**  $C$  the coverage of  $\mathcal{F}$  in  $\mathcal{H}$

**Subroutine**  $subCoverage(c, \mathcal{F}, \mathcal{H})$

**Require:**  $c$  a context,  $\mathcal{F}$  a set of minimal contexts,  $\mathcal{H}$  a context hierarchy.

Let  $C = \emptyset$

**if**  $decomp(c) \subseteq \mathcal{F}$  **then**

**for all**  $p$  parent of  $c$  in  $\mathcal{H}$  **do**

$C = C \cup subCoverage(p, \mathcal{F}, \mathcal{H})$

**end for**

**if**  $C = \emptyset$  **then**

$C = \{c\}$

**end if**

**end if**

**return**  $C$

---

## V. EXPERIMENTS

### A. Data Description

The experiments were conducted on about 100000 product reviews from *amazon.com*, in order to study the vocabulary used according to reviews. This dataset is a subset of the one used in [7]. Reviews have been lemmatized<sup>1</sup> and grammatically filtered in order to remove uninteresting terms, by using the *tree tagger* tool [11]. Preserved terms are verbs (apart from modal verbs and the verb “*be*”), nouns, adjectives and adverbs. Then the sequence database is constructed using the following principles:

- each review is a data sequence,
- each sentence is an itemset (i.e., the order of the words in a sentence is not considered),
- each word is an item.

In such sequence database, a sequential pattern could be  $p = \langle (eat\ mushroom)(hospital) \rangle$ , which means that frequently a sentence containing *eat* and *mushroom* and one of the following sentences containing *hospital* co-occur.

*Contextual dimensions:* Each review is associated with contextual dimensions:

- the *product* type (*Books, DVD, Music* or *Video*)
- the *rating* (originally a numeric value  $r$  between 0 and 5). For these experiments,  $r$  has been translated into qualitative values: *bad* (if  $0 \leq r < 2$ ), *neutral* (if  $2 \leq r \leq 3$ ), and *good* (if  $3 < r \leq 5$ )
- the proportion of helpful *feedbacks*<sup>2</sup>, i.e., 0-25%, 25-50%, 50-75% or 75-100%.

We define hierarchies on contextual dimensions as described in Figure 3. The number of contexts in the context hierarchy is  $|\mathcal{H}(product)| \times |\mathcal{H}(rating)| \times |\mathcal{H}(feedbacks)| = 6 \times 5 \times 7 = 210$ , the number of minimal contexts is  $dom(product) \times dom(rating) \times dom(feedbacks) = 4 \times 3 \times 4 = 48$ .

Note that the domain of values of contextual dimensions has been enriched with new values. For instance, hierarchy  $\mathcal{H}(rating)$  contains an element *Extreme*, that will allow us to extract patterns being specific to extreme opinions (positive or negative).

### B. Results and Discussion

All experiments have been performed on a system equipped with a 3GHz CPU and 16GB of main memory. The methods are implemented by extending a Java implementation of PrefixSpan-based algorithm described in [5].

Figure 4-A shows the scalability of our approach as the runtime increases almost linearly with the database size evolving from 12400 tuples to 99834 (i.e., the whole

<sup>1</sup>i.e., the different forms of a word have been grouped together as a single item. For instance, the different forms of the verb *to be* (is, are, was, being, etc.) are all returned as *be*.

<sup>2</sup>On *amazon.com* each reader can post a feedback on a review.

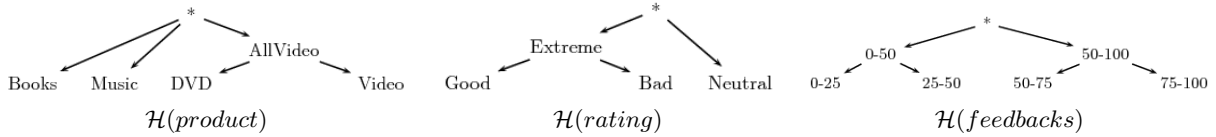


Figure 3. Hierarchies on contextual dimensions.

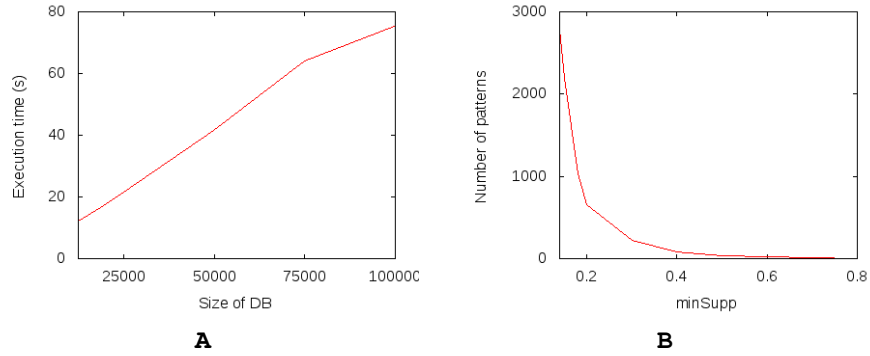


Figure 4. **A** - Execution time according to the size of  $\mathcal{CB}$  (with  $minSupp = 0.3$ ). **B** - Number of contextual sequential patterns according to  $minSupp$ .

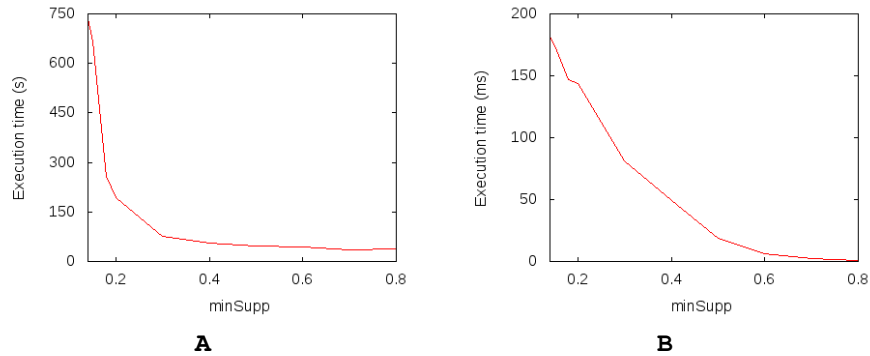


Figure 5. **A** - Global execution time according to  $minSupp$  (in seconds). **B** - Execution time for generating contextual sequential patterns according to  $minSupp$  (in milliseconds).

dataset). To construct the sampled databases, a certain number of tuples have been selected within each minimal context from the original contextual sequence database, in order to keep the same repartition of tuples over the contexts.

Figure 5-A shows the runtime in seconds for the whole process (i.e., mine the sequential patterns in 48 minimal contexts and generate the contextual sequential patterns in the whole hierarchy) while Figure 5-B shows the runtime in milliseconds for only generating the contextual sequential patterns. Once sequential patterns in minimal contexts are extracted, the time for the generation of contextual sequential patterns is negligible. For instance, it stands for only 0.026% of the global runtime when  $minSupp = 0.15$ . This shows the efficiency of our approach since, by comparison, the naive approach described in Section III would require to mine 210 contexts (instead of 48) to obtain the contextual sequential patterns.

Let us consider again the values enriching the hierarchies on contextual dimensions (e.g., value *Extreme* in

$\mathcal{H}(rating)$ ). Such values do not change the number of minimal contexts in the hierarchy, and therefore only affect the generation of contextual sequential patterns. However, as shown in Figure 5-B, the generation time is negligible in the global execution time. This means that adding such elements in the context hierarchy has a negligible impact on the execution time, despite the fact that it provides much more informative patterns.

Amongst 2193 contextual sequential patterns extracted with  $minSupp = 0.15$  (see Figure 4-B) only 13 are  $[*, *, *]$ -specific. This means that only 0.6% of extracted patterns do not depend on the contextual dimensions hence highlighting the need of taking into account contextual data for mining.

## VI. RELATED WORK

Mining frequent sequences over contextual information can be related to two important data mining problems. Let us consider first the problem of mining multidimensional sequential patterns, i.e., extracting sequential patterns dealing

with several dimensions. The first proposition is described in [9], where authors are handling data which are equivalent to contextual data in our approach. Indeed, each sequence is associated with some contextual dimensions. However, while multidimensional sequential patterns take into account contextual dimensions, they have the same drawbacks than traditional sequential pattern mining, described in Section III. For instance, a sequential pattern would be  $([y, *], \langle(ab)\rangle)$  standing for *sequence  $\langle(ab)\rangle$  associated with context  $[y, *]$  is frequent in the whole sequence database*. Hence, sequential patterns can be extracted only if they are frequent in the whole database, and patterns specific to a particular context will not appear if this context is not frequent. Newest approaches have considered more complex multidimensional sequences databases, e.g., where items are also described over several dimensions [10], and allowing to take into consideration taxonomies for those dimensions, but the previous about taking into account only frequent contexts still hold.

Second, the problem of mining emergent patterns can also be seen as related to contextual sequential pattern mining. Introduced in [4], the mining of emerging patterns aims to extract the itemsets that are discriminant to one class in a set of itemset databases. An emerging pattern is then a pattern whose support is significantly higher in a class than in others. Such patterns can then for instance be exploited to build classifiers. For instance, given two data classes  $A$  and  $B$ , emergent patterns in  $B$  would be itemsets whose support is significantly higher in  $B$  than in  $A$ . Few approaches are dealing with sequential data to find emerging sequences. [3] introduces the concept of emergent substrings, which is a very specific type of sequences. [6] proposes to mine emergent subsequences, but here sequences are sequences of items (i.e., itemsets can not be considered). Globally, although emergent sequences and contextual sequential patterns have interests in mining sequences that are more frequent in one sequence database than in others, there are important differences. The search space for emerging sequences is much smaller than contextual sequential patterns. Indeed, contextual sequential pattern mining does not only aim to mine sequences that are specific to one class, but to extract all sequential patterns in a whole sequence database, and describe if there exist some more or less general contexts where one sequential pattern is specific by handling a context hierarchy.

## VII. CONCLUSION

In this paper we have motivated and introduced contextual sequential pattern mining. We formally defined the problem and unveiled set-theoretical properties that allow database mining in a concise manner. Immediate future work include testing other real world data sets and an efficiency comparison with a naive approach not benefiting from above mentioned properties.

Our work can be extended in a number of ways. First we want to explore how using our results can be applied for classification. The context hierarchy can serve to provide a confidence score depending on the generalization / specialization level of the context hierarchy. For instance, if the confidence score to belong to a given context is below a certain threshold, then a more general context could be more appropriate. Second, we plan to use our results for prediction purposes.

## REFERENCES

- [1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2), 1993.
- [2] R. Agrawal and R. Srikant. Mining sequential patterns. In P. S. Yu and A. S. P. Chen, editors, *Eleventh International Conference on Data Engineering*. IEEE Computer Society Press, 1995.
- [3] S. Chan, B. Kao, C. Yip, and M. Tang. Mining emerging substrings. In *Database Systems for Advanced Applications, 2003.(DASFAA 2003). Proceedings. Eighth International Conference on*, pages 119–126, 2003.
- [4] G. Dong and J. Li. Efficient mining of emerging patterns: discovering trends and differences. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 43–52, New York, NY, USA, 1999. ACM.
- [5] P. Fournier-Viger, R. Nkambou, and E. Nguifo. A knowledge discovery framework for learning task models from user interactions in intelligent tutoring systems. *MICAI 2008: Advances in Artificial Intelligence*, pages 765–778, 2008.
- [6] X. Ji, J. Bailey, and G. Dong. Mining minimal distinguishing subsequence patterns with gap constraints. *Knowledge and Information Systems*, 11(3):259–286, 2007.
- [7] N. Jindal and B. Liu. Opinion spam and analysis. In *Proceedings of the international conference on Web search and web data mining*, pages 219–230. ACM, 2008.
- [8] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Mining sequential patterns by pattern-growth: the PrefixSpan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1424–1440, 2004.
- [9] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, and U. Dayal. Multi-dimensional sequential pattern mining. In *Proceedings of the tenth international conference on Information and knowledge management*, page 88. ACM, 2001.
- [10] M. Plantevit, Y. W. Choong, A. Laurent, D. Laurent, and M. Teisseire. M<sup>2</sup>SP: Mining sequential patterns among several dimensions. In A. Jorge, L. Torgo, P. Brazdil, R. Camacho, and J. Gama, editors, *PKDD*, volume 3721 of *Lecture Notes in Computer Science*, pages 205–216. Springer, 2005.
- [11] H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, volume 12. Citeseer, 1994.