Mining Representative Frequent Patterns in a Hierarchy of Contexts

Julien Rabatel, Sandra Bringay, Pascal Poncelet

Mining Representative Frequent Patterns in a Hierarchy of Contexts

Julien Rabatel[1], Sandra Bringay[12], and Pascal Poncelet[1]

[1] LIRMM (CNRS UMR 5506), Univ. Montpellier 2
161 rue Ada, 34095 Montpellier Cedex 5, France
[2] Dpt MIAp, Univ. Montpellier 3
Route de Mende, 34199 Montpellier Cedex 5, France

Abstract. More and more data come with contextual information describing the circumstances of their acquisition. While the frequent pattern mining literature offers a lot of approaches to handle and extract interesting patterns in data, little effort has been dedicated to relevantly handling such contextual information during the mining process. In this paper we propose a generic formulation of the contextual frequent pattern mining problem and provide the `CFPM` algorithm to mine frequent patterns that are representative of a context. This approach is generic w.r.t. the pattern language (e.g., itemsets, sequential patterns, subgraphs, etc.) and therefore is applicable in a wide variety of use cases. The `CFPM` method is experimented on real datasets with three different pattern languages to assess its performances and genericity.

1 Introduction

In the data mining field as well as in every data-related domain, more and more data come with contextual information detailing the circumstances under which data have been acquired. A concrete example lies in the explosion of mobile phone usage accompanied by information about the user location and user profiles. Another omnipresent example is related to the Web, where users often give some information about themselves (e.g., on forums or social media) that can be used to better understand their Web usage.

While mining patterns of very various forms and structures (itemsets, sequences, episodes, subgraphs, spatio-temporal patterns, etc.) has been studied extensively in the past two decades [6], there has been little interest in fully exploiting the surrounding data, i.e., the so-called contextual data. Many machine learning approaches can however benefit from this contextual information to finely analyze or exploit the data. In the current paper, we study and propose a solution for mining frequent patterns in the presence of contextual information. More precisely, the contributions of this paper are twofold:

- **Providing a generic theoretical framework.** We propose a formalism for defining contextual frequent patterns that does not depend on a particular pattern language. This framework has the ability to generalize the frequent pattern mining problem to consider available contextual information.

– **Contextual frequent pattern mining algorithm.** We also propose a new algorithm, so-called `CFPM`, which is generic w.r.t. the pattern language and the underlying mining algorithm. This approach is based on relevantly post-processing the output of existing algorithms, meaning that it can be applied in conjunction with any algorithm that aims at solving a transactional frequent pattern mining problem and offers a great applicability range.

While seminal work has already defined the basis of contextual frequent pattern mining in the case of sequential patterns [12, 13], the existing work has the following drawbacks: *(1)* its formulation is only dedicated to sequential patterns, while we are interested in providing a generic formulation applicable to most frequent patterns definitions; *(2)* the algorithm designed to mine contextual frequent sequential patterns uses specific techniques that make it unusable for other pattern languages. We build upon this previous work and show that the principles of mining contextual frequent patterns are not inherently associated to one pattern language, or even to one mining method, and can be used in conjunction with a lot of existing previous work for a great flexibility and applicability. Mining contextual frequent patterns only relies on pattern frequency and does not relate to how a pattern frequency contrasts with the rest of the database. Such patterns, found in the literature as *discriminative patterns*, *contrast patterns* or *correlated patterns* [4] do not fall within the scope of this study.

The remaining is organized as follows. Section 2 defines the contextual frequent pattern mining problem. Then, Section 3 describes the proposed `CFPM` algorithm. Experiments are conducted in Section 4 and some conclusions and prospects are given in Section 5.

2 Contextual Data and Frequent Patterns

This section aims at formalizing the frequent pattern mining problem as well as its extension for handling contextual information. According to [10], a large family of pattern mining problems can be specified with the following formulation: given a database D, a class of patterns $\mathcal{P}$ called a pattern language and a selection predicate q, it consists in finding the set $\{p \in \mathcal{P} \mid q(D, p) \text{ is } true\}$. This definition is refined as follows to describe the *transactional frequent pattern mining* problem addressed in this study.

Definition 1 (Transactional frequent pattern mining) *A **transaction** is a couple $T = (tid, o_T)$, where tid is a unique transaction identifier, and o_T is the transaction object, i.e., an object provided in an arbitrary description space. A **transactional database** is a set of transactions.*

*The pattern language is associated with a **support operator** $\prec$, such that a transaction T supports a pattern p (or p is supported by T) if $p \prec T$. From the support operator, one can define the **frequency** of p in D as the fraction of transactions in D supporting p: $Freq_D(p) = \frac{|\{T \in D | p \prec T\}|}{|D|}$.*

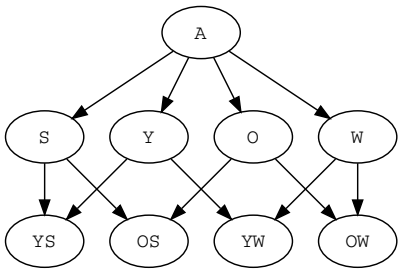
Frequent patterns are those whose frequency is above a user-specified ***minimum frequency threshold****. In other terms, given a transactional database D, a pattern language $\mathcal{P}$, a pattern support operator $\prec$, and a minimum frequency threshold σ, the* ***transactional frequent pattern mining*** *problem refers to finding the set $\mathcal{F}(\mathcal{P}, D, \prec, \sigma) = \{p \in \mathcal{P} | Freq_D(p) \geq \sigma\}$.*

As an example, the well-known *frequent itemset mining problem* [1] within this transactional setting can be: given an alphabet of items $\mathcal{I} = \{a, b, c, d, e\}$, transaction objects are itemsets, i.e., subsets of $\mathcal{I}$. The pattern language $\mathcal{P}$ is defined as $2^{\mathcal{I}}$ and the *support operator* $\prec$ as the set inclusion operator between patterns and transaction itemsets. For instance, given a pattern $p = \{a, d\}$ and a transaction $T = (t, \{a, b, d, e\})$, then $p \prec T$ because $\{a, d\} \subseteq \{a, b, d, e\}$.

Figure 1(a) provides an example of a transactional database of itemsets D using the alphabet $\mathcal{I}$. The first column gives the identifier of each of the 14 transactions, while the second one provides the corresponding transaction itemset. The third column is not used in the transactional pattern mining setting. By considering a minimum frequency threshold σ of 0.5, we notice that the pattern $p = \{a, b\}$ has a frequency $Freq_D(p) = {}^{8}/_{14}$ and is therefore frequent with $Freq_D(p) \geq \sigma$.

tid	Itemset	Context
t_1	$\{a, b, e\}$	YS
t_2	$\{a, b, d\}$	YS
t_3	$\{a, b, e\}$	YS
t_4	$\{a, b, c\}$	YS
t_5	$\{a, b\}$	YS
t_6	$\{a, c, d\}$	YW
t_7	$\{a, b, d\}$	YW
t_8	$\{a, b\}$	YW
t_9	$\{a, b, c, e\}$	OS
t_{10}	$\{b, c, d\}$	OS
t_{11}	$\{b, d\}$	OS
t_{12}	$\{b, d, e\}$	OW
t_{13}	$\{b, d\}$	OW
t_{14}	$\{a, c, d, e\}$	OW

(a) A transactional database with associated contexts.



(b) A context hierarchy.

Fig. 1: A transactional contextual database composed of *(a)* a transactional database of itemsets with the associated minimal contexts, and *(b)* a context hierarchy.

Such a theoretical framework is representative of a large fraction of frequent pattern mining approaches appeared in the literature in the past decades. These problems exploit a *transactional* view of the data, i.e., they are represented under

the form of a collection of transactions and frequent patterns are those mapped to at least a given number of transactions. Among pattern mining problems that do not enter this family, an example is the relatively recent problem of mining patterns in one unique large graph or network, addressed for instance in [3].

We are interested in enriching this transactional setting for mining frequent patterns in the presence of contextual information, i.e., data describing some circumstances regarding each transaction. We therefore introduce the *context hierarchy* to manipulate this contextual information.

Definition 2 (Context hierarchy) *A **context hierarchy** $\mathcal{H}$ is a directed acyclic graph (DAG), denoted by $\mathcal{H} = (V_{\mathcal{H}}, E_{\mathcal{H}})$, such that*

- *$V_{\mathcal{H}}$ is a set of vertices also called **contexts**,*
- *$E_{\mathcal{H}} \subseteq V_{\mathcal{H}} \times V_{\mathcal{H}}$ is a set of directed edges among contexts.*

$\mathcal{H}$ is naturally associated with a partial order $<_{\mathcal{H}}$ on its vertices, defined as follows: given $c_1, c_2 \in V_{\mathcal{H}}$, $c_1 <_{\mathcal{H}} c_2$ if there exists a directed path from c_2 to c_1 in $\mathcal{H}$. This partial order describes a specialization relationship: c_1 is said to be more *specific* than c_2 if $c_1 <_{\mathcal{H}} c_2$, and more *general* than c_2 if $c_2 <_{\mathcal{H}} c_1$.

A *minimal context* from $\mathcal{H}$ is a context such that no more specific context exists in $\mathcal{H}$, i.e., $c \in V_{\mathcal{H}}$ is minimal iff $\nexists c' \in V_{\mathcal{H}} \mid c' <_{\mathcal{H}} c$. The set of minimal contexts in $\mathcal{H}$ is denoted as $V_{\mathcal{H}}^{-}$.

A context hierarchy aims at offering more information about the elements of a transactional database D when mining frequent patterns. A transactional database and a context hierarchy are combined to produce a *contextual transactional database* $\mathcal{D}$, i.e., a triple $(D, \mathcal{H}, \delta)$ such that:

- D is a transactional database,
- $\mathcal{H}$ is a context hierarchy,
- δ is a function $\delta : D \mapsto V_{\mathcal{H}}^{-}$ mapping each transaction from D to a minimal context in $\mathcal{H}$.

In a contextual transactional database $\mathcal{D}$, a transaction T is explicitly mapped to a minimal context given by $c = \delta(T)$. By following the intuition that a transaction associated to a very specific context also is part of the more general contexts, we define the *database induced by* c, denoted by $D(c)$, as the subset of D which is associated with c. More formally, $D(c) = \{T \in D | (\delta(T) <_{\mathcal{H}} c) \vee (\delta(T) = c)\}$.

Example. To illustrate the contextual database notions, let first consider Figure 1(b) which provides a visual representation of a context hierarchy. Contexts are the labels of vertices, such as contextual information is given by the age (*young* or *old*, respectively shorten to `Y` and `O`) and the season (*summer* or *winter*, respectively shorten to `S` and `W`). Some examples of contexts provided by this context hierarchy shown in Figure 1(b) are (from the more specific to the more general) `YS`, `S`, or `A`, respectively corresponding to "*transactions associated to young people in summer*", "*transactions associated to summer (regardless of*

the age of people)" and "*all (A) the transactions regardless of the age and season*". The third column of Figure 1(a) describes the δ function by mapping each transaction to a minimal context, such as the first transaction identified by t_1 associated with the context YS, i.e., $\delta(t_1) = YS$. Figure 1 hence provides a contextual transactional database $\mathcal{D}$. From this δ mapping, we notice for instance that the database induced by YS (i.e., $D(YS)$) is the set of transactions of identifiers $t_1, \ldots, t_5$, while $D(A)$ is the set of all transactions.

2.1 When should a pattern be associated with a context?

The contextual frequent pattern mining problem aims at discovering patterns whose the property of being frequent is context-dependent. In order to study and highlight the interest of exploiting contextual information within the frequent pattern mining process, we below isolate two patterns from our running example.

Case 1. Itemset $\{a, b\}$ is frequent in D ($Freq_D(\{a, b\}) = 8/14$). However, considering the database and its fragmentation given by contextual information, one can notice that $\{a, b\}$ is frequent in the fraction of D corresponding to young people (Y) with a frequency of $7/8$ while it is infrequent in the rest of D, i.e., old people. In the following, we will state that this pattern, while *frequent in A*, is not *general in A* because it is not frequent in every context contained in A. On the other hand, this pattern is *general in Y* because, in addition to be frequent in Y, it is also frequent in YS ($Freq_{YS}(p) = 5/5$) and YW ($Freq_{YW}(p) = 2/3$), i.e., all contexts contained in Y.

Case 2. $p' = \{b, d\}$ is not frequent in D ($Freq_D(p') = 6/14$), while it however is frequent in O ($Freq_O(p) = 5/6$) as well as in the contexts contained in O: its frequency is $2/3$ both in OS and OW.

Case 1 shows that simply mining frequent patterns within a context does not necessarily provide representative patterns. In addition, Case 2 shows a pattern that is representative of a given context, but mining frequent patterns in the whole database could not make such patterns emerge as the context they represent is not large enough relatively to the whole database.

The next section exploits these intuitions to formally define what types of frequent patterns are mined in a contextual database and how they relate to the context hierarchy.

2.2 Contextual Frequent Patterns: a Formal Definition

The current section applies the contextual transactional setting defined above to first reformulate the notion of frequent pattern within a context and then introduce the notions related to the contextual frequent patterns (CFPs).

Definition 3 (c-frequent pattern) *A pattern p is frequent in c, or c-frequent, iff p is frequent in $D(c)$, i.e., if $Freq_{D(c)}(p) \geq \sigma$. For the sake of readability, we denote $Freq_{D(c)}(p)$ with $Freq_c(p)$.*

As discussed in the previous section, we are interested in patterns being representative of a context, i.e., such that their frequency property holds for all the descendants of this context. Such patterns are called *general patterns* and are used to define CFPs.

Definition 4 (c-general pattern) *A pattern p is general in c, or c-general, iff p is c-frequent and p is c'-frequent $\forall c' \mid c' <_{\mathcal{H}} c$.*

Definition 5 (Contextual frequent pattern) *A contextual frequent pattern is a couple $\alpha = (c, p)$, such that p is c-general. α is said to be **generated by** p. (c, p) is **context-maximal** if there does not exist another context c' more general than c and such that (c', p) is a CFP, i.e., $\nexists c' \in V_{\mathcal{H}} | (c <_{\mathcal{H}} c')$ and p is c'-general.*

*The **CFP mining problem** consists in enumerating all the context-maximal CFPs given a contextual database $\mathcal{D}$ and a minimum frequency threshold σ.*

The set of CFPs that are context-maximal constitutes an exact condensed representation of the set of CFPs, as no CFP cannot be derived from a *context-maximal* one. Indeed, following Definition 4, one may notice that if a pattern is c-general, then it is also general in all descendants of c. Therefore, mining all CFPs in $\mathcal{D}$ is equivalent to mining context-maximal CFPs only.

The CFP mining framework also has the advantage of associating to each context in $\mathcal{H}$ less patterns than what a typical transactional frequent pattern miner would provide (as being frequent in a context is only one of the requirements for a pattern to generate a CFP). To some extent,

3 Mining Contextual Frequent Patterns

This section describes how CFPs are mined given a contextual transactional database, by defining two approaches: *(1)* a baseline approach that makes direct use of the definitions given in Section 2, and *(2)* a more efficient approach that relies on theoretical properties emerging from the CFP mining framework.

3.1 A Baseline Approach

By relying on the requirements listed in Definition 4, Algorithm 1 provides a baseline approach to extract context-maximal CFPs. This approach relies on the following steps: *(1)* extracting frequent patterns from every possible context (lines 2-4), *(2)* for each context c and each pattern p frequent in c, checking the c-generality and context-maximality of (c, p) (lines 5-11).

Mining all the contexts in order to enumerate all their frequent patterns obviously is very time-consuming, as it requires to run an external pattern miner for each context separately. We therefore study in the following some theoretical properties in order to allow a more efficient extraction of CFPs.

Algorithm 1 A Baseline Approach

Require: A contextual database $\mathcal{D}$, a minimum frequency threshold σ
Ensure: Set of contextual frequent patterns in $\mathcal{D}$
1: $\mathcal{C} \leftarrow \emptyset$ `// initialize the set of discovered CFPs`
2: **for** $c \in \mathcal{D}$ **do**
3: $\quad \mathcal{F}_c \leftarrow$ frequent patterns in c
4: **end for**
5: **for** $c \in \mathcal{D}$ **do**
6: $\quad$ **for** $p \in \mathcal{F}_c$ **do**
7: $\quad\quad$ **if** p is c-general and context-maximal **then**
8: $\quad\quad\quad \mathcal{C} \leftarrow \mathcal{C} \cup \{(p, c)\}$ `// generate and store the contextual pattern`
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: **end for**

3.2 `CFPM`: A More Efficient Post-Processing Approach

The approach described in Algorithm 1 trivially exploits the definition of contextual data and CFPs and leads to costly calculation, in particular by first extracting frequent patterns from each possible context. We highlight some interesting properties to reduce redundant calculations, in particular by reducing the executions of the frequent pattern miner. As opposed to [12, 13], we focus in this paper on providing a generic post-processing algorithm that, from the output of existing frequent pattern miners, generates the CFPs.

Additional properties of contextual general patterns. A context can be uniquely described by its minimal descendants in $\mathcal{H}$. To this end, we consider the *decomposition* of a context c in $\mathcal{H}$ as the set of minimal contexts in $\mathcal{H}$ being more specific than c, i.e., $decomp(c, \mathcal{H}) = \{c' \in V_{\mathcal{H}}^{-} | (c' <_{\mathcal{H}} c) \vee (c' = c)\}$. For instance, $decomp(Y) = \{YS, YW\}$ and $decomp(YS) = \{YS\}$.

Property 1 *p is c-general iff p is c'-frequent $\forall c' \in decomp(c)$.*

Property 1 (whose proof can be found in [12] and adapted to the current framework) is essential by allowing the reformulation of the c-generality property w.r.t. minimal contexts only. The checking of c-generality requirements for a context thus becomes much simpler: a pattern p is c-general if and only if the set of minimal contexts where p is frequent includes the decomposition of c. Extending this property to context-maximal CFPs is straightforward. `CFPM`, as presented in Algorithm 2, exploits this property. It can be decomposed into the following consecutive steps:

Mining. *(lines 2-4)* Frequent patterns are extracted from each minimal context. As opposed to Algorithm 1, `CFPM` does not mine non-minimal contexts.

Algorithm 2 `CFPM`: Contextual Frequent Pattern Mining

Require: A contextual database D, a minimum frequency threshold σ
Ensure: Set of contextual frequent patterns in D

```
1: C ← ∅ // initialize the set of discovered CFPs
2: for c ∈ V_H^- do
3:    F_c ← frequent patterns in c
4: end for
5: for p ∈ ⋃_{c∈V_H^-} F_c do
6:    l_p ← {c ∈ V_H^- | p ∈ F_c}
7:    K[l_p] ← K[l_p] ∪ {p}
8: end for
9: for l a key in K do
10:   for c ∈ maxContexts(l, H) do
11:      for p ∈ K[l] do
12:         C ← C ∪ (p, c) // generate and store a CFP
13:      end for
14:   end for
15: end for
```

Reading. *(lines 5-8)* Output files from previous step are read and patterns p are indexed by the set of minimal contexts where they are frequent, i.e., l_p. Then, K is a hash table with keys being sets of minimal contexts and values being sets of patterns, such as $K[l]$ containing the patterns p such that $l_p = l$. The cost of this step mainly lies on intensive I/O processing.

Coverage Computation and Pattern Generation. *(lines 9-15)* During this step, each key of K is given to the *maxContexts* routine *(line 10)* which performs a bottom-up traversal of the vertices of $\mathcal{H}$ in order to return the set of maximal contexts among $\{c \in V_{\mathcal{H}} \mid decomp(c) \subseteq l\}$. This is the coverage computation step. Then, for each pattern p such that $l = l_p$*(line 11)* and each context returned by *maxContexts* (*line 10*), one context-maximal CFP is generated and stored (*line 12*). Two patterns p and p' frequent in the same minimal contexts (i.e., $l_p = l_{p'}$) are general in the same contexts. They will generate the same result via the *maxContexts* routine. By using the hash table K to store the patterns that are frequent in the same minimal contexts, the number of calls to *maxContexts* is greatly reduced to the number of keys in K rather than the number of distinct patterns discovered during the *mining* step.

Discussion. Mining minimal contexts only is an essential advantage over the baseline approach. `CFPM`'s post-processing oriented design also offers the possibility to use it with any transactional frequent pattern miner, whatever the structure of mined patterns (e.g., subgraphs, episodes, sequential patterns, itemsets, etc.). This genericity also is the main advantage over previous work [12, 13].

4 Experimental Results

The implementation of the algorithm is divided into two parts. First, a `Ruby` wrapper is in charge of running external pattern miners, reading their output, and eventually generating the CFPs. `Ruby`'s flexibility is particularly relevant for designing a generic approach, where final users should be able to add new components with very little effort to support new pattern languages. Second, a `C++` module is in charge of the `maxContexts` routine of Algorithm 2, as it offers better performances without any drawbacks regarding genericity or usability.

The `CFPM` approach has been extensively experimented in order to assess its main features. Therefore, we have performed experiments implying real datasets and three common pattern languages, namely *itemsets* [1], *sequential patterns* [2] and *subgraphs* [9]. Each of these pattern languages involves different theoretical frameworks and algorithms. Experiments have been conducted on an Intel i7-3520M 2.90GHz CPU, with 16 GB memory.

Contextual Frequent Itemsets. First, in order to study the behavior of `CFPM` when considering the frequent itemset mining problem [1], we have used the `APriori` algorithm as implemented in [5]. The dataset used for this experiment initially comes from [8]. It consists of $100,000$ product reviews published on the *amazon.com* website. Reviews have been lemmatized and grammatically filtered[3]. The remaining words compose the item alphabet and each review is represented as a transaction. Contextual information associated with the reviews is composed of the *type of product*, the *rating* given by the user and the *proportion of positive feedbacks* received. The resulting context hierarchy contains 210 contexts, whose 48 are minimal. The interested user may refer to [12] for details.

Contextual Frequent Sequential Patterns. The second pattern mining problem addressed with `CFPM` considers *frequent sequential patterns* as defined in [2]. To this end, `CFPM` uses the `PrefixSpan` [11] algorithm as implemented in the SPMF project [5]. The used dataset is the same as the one described above for frequent itemset mining, except that reviews have been converted to sequences of itemsets[4]. It is also the same as the one previously used in [12].

Contextual Frequent Subgraphs. In order to address the subgraph pattern language [9], we use a dataset that has been constructed to study the mutagenicity property of some molecules [7]. It contains $6,512$ molecular graphs, such that each one is associated with a label that indicates whether it is mutagene or not. Contextual information is composed of the mutagenicity label, the molecular weight, and the source dataset from which the molecule has been extracted. The merging of these pieces of information produces a context hierarchy containing 39 contexts whose 10 are minimal.

[3] Remaining terms are non-modal verbs, nouns, adjectives and adverbs.

[4] The conversion follows the principle that each sentence of the review is an itemset, and the order of itemsets in a sequence results from the order of sentences.

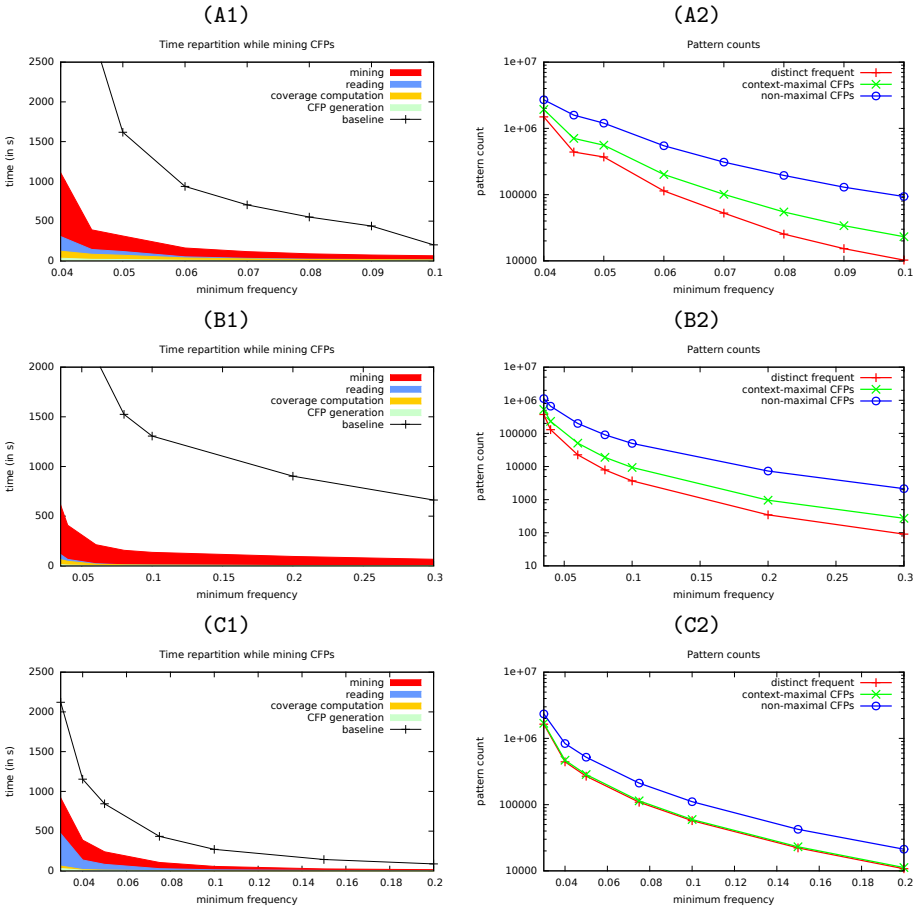


Fig. 2: Time consumption for mining CFPs for itemsets (A1), sequential patterns (B1) and subgraphs (C1), and number of distinct patterns and contextual patterns for itemsets (A2), sequential patterns (B2) and subgraphs (C2).

Results. First of all, Figures 2(A1,B1,C1) show a large difference of runtimes when comparing the baseline approach with CFPM, for every used dataset and pattern language. This gap of runtime is mainly due to the fact that the baseline approach first requires to mine frequent patterns from each context, while CFPM only mines minimal contexts, that are less numerous and smaller. Then, Figures 2(A1,B1,C1) provide a view on how the runtime of CFPM is decomposed according to the algorithm steps. The largest fraction of time corresponds to the mining step, i.e., running pattern miners for each minimal context. This fraction systematically increases while the minimum frequency threshold decreases. Of

course, this fraction of runtime also depends on the underlying implementations and algorithms. For instance, the time required to mine sequential patterns is relatively much larger (*cf.* Figure 2(B1)).

Figures 2(A2,B2,C2) show the total amount of patterns regarding their type. First, let consider the number of distinct patterns (i.e., the distinct frequent patterns discovered during the mining step) compared with the number of context-maximal CFPs. As expected, the latter is always greater than or equal to the number of distinct patterns since every distinct pattern generates at least one CFP. On the other hand, the total number of CFPs (i.e., not necessarily context-maximal ones) is much higher, therefore demonstrating the interest of proposing a condensed representation of contextual frequent patterns.

5 Conclusion and Prospects

In this paper, we adapt the transactional setting for mining frequent patterns by considering the contextual information associated with transactions. We therefore define a theoretical framework for CFP mining and propose a relevant algorithm for mining such patterns in a totally generic way regarding the pattern language. By generalizing the typical transactional setting and by post-processing the output of existing frequent pattern miners, the proposed approach provides the benefit of being able to be used in conjunction with any such frequent pattern miner developed during the last decades. Such an approach provides opportunities to be exploited in numerous application domains where data are often accompanied with contextual information, e.g., the mining of mobile data where spatial and temporal information may be used as contextual data or user profiling on the Web, where user activities may be mined under the scope of contextual information about the user such as location, age, etc.

The contextual pattern mining approach offers numerous prospects, such as for instance adapting it to the case of relying on discriminative patterns rather than frequent patterns in the context hierarchy, or considering the case where contextual information is imprecise (i.e., some transactions are mapped to a non-minimal context).

References

1. Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.
2. Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *International Conference on Data Engineering*, pages 3–14. IEEE, 1995.
3. Björn Bringmann and Siegfried Nijssen. What is frequent in a single graph? In *Advances in Knowledge Discovery and Data Mining*, pages 858–863. Springer, 2008.
4. Björn Bringmann, Siegfried Nijssen, and Albrecht Zimmermann. Pattern-based classification: A unifying perspective. *LeGo Workshop*, page 36, 2009.
5. P. Fournier-Viger, A. Gomariz, A. Soltani, H. Lam, and T. Gueniche. Spmf: Open-source data mining platform. http://www.philippe-fournier-viger.com/spmf, 2014.

6. Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86, 2007.
7. Katja Hansen, Sebastian Mika, Timon Schroeter, Andreas Sutter, Antonius ter Laak, Thomas Steger-Hartmann, Nikolaus Heinrich, and Klaus-Robert Müller. Benchmark data set for in silico prediction of ames mutagenicity. *Journal of Chemical Information and Modeling*, 2009.
8. Nitin Jindal and Bing Liu. Opinion spam and analysis. In *International Conference on Web Search and Data Mining*, pages 219–230. ACM, 2008.
9. Michihiro Kuramochi and George Karypis. Frequent subgraph discovery. In *International Conference on Data Mining*, pages 313–320. IEEE, 2001.
10. Heikki Mannila and Hannu Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data mining and knowledge discovery*, 1(3):241–258, 1997.
11. Jian Pei, Helen Pinto, Qiming Chen, Jiawei Han, Behzad Mortazavi-Asl, Umeshwar Dayal, and Mei-Chun Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 0215–0215. IEEE Computer Society, 2001.
12. Julien Rabatel, Sandra Bringay, and Pascal Poncelet. Contextual sequential pattern mining. In *International Conference on Data Mining Workshops*, pages 981–988. IEEE, 2010.
13. Julien Rabatel, Sandra Bringay, and Pascal Poncelet. Mining sequential patterns: a context-aware approach. In *Advances in Knowledge Discovery and Management*, pages 23–41. Springer, 2013.