Acta Univ. Sapientiae, Informatica, **6,** 2 (2014) 287–310

DOI: 10.1515/ausi-2015-0008

Finding sequential patterns with TF-IDF metrics in health-care databases

Zsolt T. KARDKOVÁCS
U1 Research
2 INFOPARK Gábor Dénes
Budapest, Hungary
email: kardkovacs@u1research.org

Gábor KOVÁCS
U1 Research
2 INFOPARK Gábor Dénes
Budapest, Hungary
email: kovacsg@u1research.org

Abstract. Finding frequent sequential patterns has been defined as finding ordered list of items that occur more times in a database than a user defined threshold. For big and dense databases that contain really long sequences and large itemset such as medical case histories, algorithm proposed on this idea of counting the occurrences output enourmous number of highly redundant frequent sequences, and are therefore simply impractical. Therefore, there is a need for algorithm that perform frequent pattern search and prefiltering simultaneously. In this paper, we propose an algorithm that reinterprets the term support on text mining basis. Experiments show that our method not only eliminates redundancy among the output sequences, but it scales much better with huge input data sizes. We apply our algorithm for mining medical databases: what diagnoses are likely to lead to a certain future health condition.

1 Introduction

The main goal of any data mining process is to find novel, interesting patterns. In the last two decades, data gathering has been accelerated, which brought the attention of data scientists from pattern or association oriented

Computing Classification System 1998: H.2.8. H.3.3. I.1.2. I.5.3. J.3.
Mathematics Subject Classification 2010: 68P10, 68T30, 68W05
Key words and phrases: sequence mining, frequent sequential pattern, TF-IDF, health care database

knowledge discovery to transition analysis or sequential data mining, which is a generalization of the former task. Sequential data mining has become an important task in many real-life applications, e.g. real-time recommendation systems, health care service optimization, fraud detection.

In the known data mining publications, absolute and relative frequencies (called support) have been used as the sole criterion in selecting patterns, while significance or interestingness have been addressed by different solutions. Published solutions suggest to calculate frequent and interesting patterns independently first, and to combine solutions into a resulting set later. This approach implicitly states that frequency is more important than significance, or more accurately: significance is taken into account if and only if an interesting pattern is frequent enough.

Note that, the two properties strongly correlate:

- if minimum frequency (threshold for support) is set too high, then interesting patterns might be omitted
- if minimum frequency is set too low, then the most of the found patterns are trivial, particular cases of others, or they are uninteresting otherwise
- there is no known golden rule how to set frequencies properly.

So independent processing guarantees information loss. Moreover, consider the following situation: there are known transactions (baskets with purchased goods) in a supermarket. Well-known pattern mining algorithms reveals frequent patterns about beers, cheese, and dumpers, however, the real profit is made from high-end products like branded whiskeys or top smart phones. Since there are a relatively few number of most profitable customers who purchase expensive goods, data mining algorithms do not discover the most significant, profitable customer needs, because they focus on frequent, mass products only.

From another point of view, if we consider that the baskets are textual documents and items are words, then well-known pattern mining algorithms would identify the most frequent words in all documents, and later on they would deal with those too frequent in all documents. As it is expected it would find tons of uninteresting patterns extremely slowly, while significant ones are completely missed. Since there are very efficient document indexing methods like *TF-IDF* [12] to handle this problem properly, our approach aims at adapting fundamental ideas from information retrieval techniques to boost sequence mining algorithms' performance.

Significance, interestingness, or relevance are vague notions. The most common definition is as follows:

Definition 1 (Absolute significance) *A data mining pattern* P *is said to be absolutely significant if* P *is previously unknown, and there exists a null model* $\mathcal{Z}$ *for which* $\Pr(\mathsf{P})$ *is statistically relevant.*

In this paper, we argue on that significance cannot be separated from the item or itemset we are analyzing at the time, i.e. there is or at least there shall be a fix point, a point of view to capture this notion properly.

Definition 2 (Relative significance) *A data mining pattern* P *is said to be relatively significant regarding an item* I *if* $\mathrm{I} \in \mathsf{P}$, P *is previously unknown, and there exists a null model* $\mathcal{Z}$ *for which* $\Pr(\mathsf{P}|\mathrm{I})$ *is statistically relevant.*

While our approach is more restrictive, it is easy to prove that absolute significance can be addressed by relative ones. In this paper, we introduce a novel algorithm called REVIEW (REIevance from the items' point of VIEW), a point of view oriented sequence mining algorithm, which finds all frequent and significant patterns in polynomial time. In addition, we also prove that REVIEW finds the most likely anti-patterns in a hand, i.e. those items, which tend to mutually exclude each other in itemsets. This property is beyond the capabilities of state-of-the-art algorithms.

The paper is organized as follows. In Section 2, we review the most important sequence mining algorithms and show an example of their scalability issues. We also overview the alternative definitions of importance that exist in the literature. Section 3 gives the elementary definitions of sequence mining: items, itemsets, sequences, sequence databases, and gives illustrative examples for the definitions. The algorithm we propose for finding frequent and important patterns is defined in Section 4. Empirical comparison is given in Section 5, our algorithm algorithm is tested on a real-life health care database against PrefixSpan and SPADE, the two fastest algorithms in the literature. Finally in Section 6 a brief summary is given.

2 Related work

In this section, we give an overview of the most important frequent sequential pattern mining algorithms: GSP, PrefixSpan, SPADE and SPAM. In the literature several other algorithms exist as well, however, those can be considered as the variants, extensions of the ones presented here. We also discuss the performance problems that arises when the size of the input database grows. In the literature, there are a lot of alternative definitions for importance, we review these definitions.

2.1 GSP

The GSP algorithm proposed by Srikant and Agrawal in [13] is built on the pattern of the a priori algorithm. First, it scans the database and counts the support of each item, detects all single item frequent sequences. Then in each subsequent pass a candidate generation and candidate counting takes place. Candidate generation uses the frequent sequential patterns of the previous pass: if removing the first element of a frequent sequence and removing the last element of another frequent sequence are the same, then the two sequences are joined and a new sequence with one more item is created. The candidate counting scans for each new sequence in the database counting the occurrences, and the ones with support greater than the user defined minimum support are retained as frequent sequences of the pass. The candidate generation and candidate counting are repeated until no frequent sequences are found.

2.2 PrefixSpan

PrefixSpan proposed by Pei et al. [8] is also based on the frequent pattern growth principle like GSP, however, it does not perform the search on the entire database for each candidate sequence, but on smaller projected databases. The sequence database is partitioned based on the itemsets of each frequent sequence of previous passes such that all sequences that support the frequent sequence are within the partition and the sequences not supporting are not. If several frequent sequences share the same itemset, then those use the same database partition. The hypothesis is that the support of a frequent sequence that is one item longer can be calculated on that partition as outside of that partition it is not supported. New candidate sequences are generated only locally by combining sequences that use the same partition. This method is a significant speed improvement over GSP as database partitions are smaller and because of the shared itemsets candidate counting does not need to be performed for each frequent sequence, but only for shared prefixes.

2.3 SPADE

SPADE (Sequential PAttern Discovery using Equivalence Classes) proposed by Zaki [15] aims to reduce the number of database scans and minimize computational costs. During the database scans frequent sequences of length one and two are searched for and their support is counted. The algorithm maintains an id-list for each item where each element of the id-list is a pointer to a sequence id and an itemset the item occurs in. Candidate sequences with one more item are generated with temporal joins or intersections on the id-lists of frequent sequences of maximum length, the support is calculated in the memory, and the new sequence is frequent if the cardinality of the resulting id-list is greater than the minimum support value. Frequent sequences are clustered into smaller sub-lattices based on common prefixes that enables independent processing.

2.4 SPAM

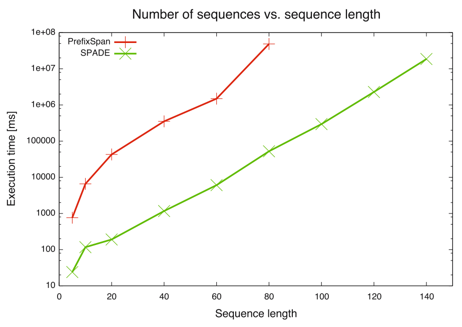
SPAM (Sequential PAttern Mining) proposed by Ayres et al. [3] assumes that the entire sequential database can completely fit in the memory and no sequences are longer than 64. The hypothesis is that frequent sequences can be found in the lexicographic tree with a simple depth-first search. Each sequence is represented with a vertical bitmap, if an item appears in a sequence then the corresponding element of the bitmap is set to one. Itemsets are generated with a bitwise and operation on the vectors of the items. Candidate sequences are generated with depth-first search from bitvectors of previous sequences and the vector of a next item in the lexicographic tree such that a bitwise and operation is performed on the two vectors, the candidate is frequent if it has more ones in its bitvector than the minimum support. The algorithm is fast, but very limited with regard to the input database.

2.5 Performance issues

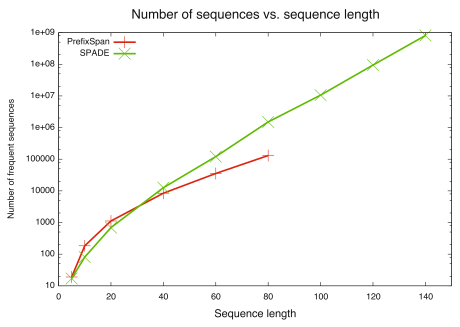
In [7], Gouda and Hassaan argue that typical sequential pattern mining algorithms tend to lose their efficiency when applied to a dense database. Their experiments confirm that the execution time increases exponentially as the number of frequent sequences increases even when the execution times in their experiments remain in the order of a few hundred seconds.

We conducted similar experiments on a subset of a medical database covering 23856 out of 455514 that is about 5% of the patient data. The average length of sequences related to a patient is 307 in that sample, the average sequence size is 10.88 itemsets. The relative or absolute minimum support thresholds were set so that the number of occurrences of a frequent sequence were at least 30.

Figure 1 shows how PrefixSpan and SPADE that are the sequential pattern mining algorithms considered to be the fastest in the literature scale as we consider longer sequences from our sample set. In the experiments, we used the reference implementations available in the SPMF library [4]. The horizontal



(a) Sequence length vs. execution time



(b) Sequence length vs. frequent sequences

Figure 1: The maximum length in a sequence database vs. the execution time and the number of frequent sequences for the two fastest sequential pattern mining algorithms, PrefixSpan and SPADE

axes show the maximum length of sequences used in the mining process. The sequence lengths considered in the experiment were 5, 10, 20, 40, 60 and 80. Note that these values are still far away from 300 that is the average length in our dataset. In Figure 1a, the vertical axis is the execution time of the algorithm in milliseconds in logarithmic scale. In Figure 1b the vertical axis is the number of frequent sequences detected in logarithmic scale by the algorithms that are proportional to their memory and disc space usages.

The results not surprisingly show that the algorithms fail to produce usable results. The algorithms do not scale with the average length. The time, memory and disk space requirements are exponentially proportional to the length of the input sequences.

2.6 Significance

A general idea to find *interesting* patterns is widely discussed in the literature. A very detailed description on different aspects of significance is found in [10, 6]. According to Geng et al. [6] definition interestingness can be broken down into the following categories:

- Conciseness. A pattern is concise if it contains relatively few attribute-value pairs, while a set of patterns is concise if it contains relatively few patterns.

- Generality/Coverage. A pattern is general if it covers a relatively large subset of a dataset.
- Reliability. A pattern is reliable if the relationship described by the pattern occurs in a high percentage of applicable cases.
- Peculiarity. A pattern is peculiar if it is far away from other discovered patterns according to some distance measure.
- Diversity. A pattern is diverse if its elements differ significantly from each other, while a set of pattern is diverse if the patterns in the set differ significantly from each other.
- Novelty. A pattern is novel to a person if he did not know it before and are not able to infer it from other known patterns.
- Surprisingness. A pattern is surprising (or unexpected) if it contradicts a person's existing knowledge or expectations.
- Utility. A pattern is of utility if its use by a person contributes to reaching a goal.
- Actionability or Applicability. A pattern is actionable (or applicable) in some domain if it enables decision making about the future actions in this domain.

Some of these notions correlate, some are subjective, some are objective, and some depends on the semantics, but they share a common feature: all of them use some kind of statistical relevance metrics to describe a particular meaning of interestingness. That is why we used the notion significance in Definition 1 as a union of these possible meanings where $\mathrm{Pr}(\mathsf{P})$ corresponds to the proper relevance metrics depending on the use case. Note that, $\mathrm{Pr}(\mathsf{P})$ is a statistical function but how to calculate is undetermined in general; it can be adapted to the problem specific needs. Later on this paper, we use the term significance measure for $\mathrm{Pr}(\mathsf{P})$. Table 1 summarizes the most common significance measure in sequential pattern mining [6].

3 Preliminaries

In this section, we give preliminary definitions necessary for the formalization of the problem statement: the definition of the frequent sequential pattern discovery and the definition of relevant pattern discovery.

Throughout this paper, we use the following conventions:

- sets and elements of sets are denoted by capital letters and lower case letters, respectively,

Measure	**Formula**
Support	$\Pr(AB)$
Lift/Interest	$\dfrac{\Pr(B\|A)}{\Pr(B)}$ or $\dfrac{\Pr(AB)}{\Pr(A)\Pr(B)}$
Interestingness Weighted Dependency	$\left(\left(\dfrac{\Pr(AB)}{\Pr(A)\Pr(B)}\right)^k - 1\right)\Pr(AB)^m$
Added value	$\Pr(B\|A) - \Pr(B)$
Relative risk	$\dfrac{\Pr(B\|A)}{\Pr(B\|\neg A)}$
Mutual information	$\sum_i \sum_j \Pr(A_iB_j)\log_2 \frac{\frac{\Pr(A_iB_j)}{\Pr(A_i)\Pr(B_j)}}{-\sum_i \Pr(A_i)\log_2\Pr(A_i)}$
Certainty factor	$\dfrac{\Pr(B\|A) - \Pr(B)}{1 - \Pr(B)}$
Conviction	$\dfrac{\Pr(A)\Pr(\neg B)}{\Pr(A\neg B)}$
Odds ratio	$\dfrac{\Pr(AB)\Pr(\neg A\neg B)}{\Pr(A\neg B)\Pr(\neg AB)}$
Yule's Q	$\dfrac{\Pr(AB)\Pr(\neg A\neg B) - \Pr(A\neg B)\Pr(\neg AB)}{\Pr(AB)\Pr(\neg A\neg B) + \Pr(A\neg B)\Pr(\neg AB)}$
Cosine	$\dfrac{\Pr(AB)}{\sqrt{\Pr(A)\Pr(B)}}$

Table 1: Some probability based objective measures for data mining[6]

- itemset and items are taken from the beginning of the latin alphabet,
- $|X|$ denotes the size of X where X is a set of attributes or itemsets,
- we use R, S symbols for database relations defined as subsets of a Cartesian products, and $r, s\ldots$ for tuples, records or elements of a relation. If $r \in R$ and $R \subseteq A \times B$ then $r(a, b)$ is short form to say $a \in A$, $b \in B$, and $r[A] = a$, $r[B] = b$ where $r[X]$ stands for the attribute values of r on attribute set X (projection in relational database theory),
- identifiers are denoted by letters near I,
- $\top$ and $\bot$ stand for logical values `true` and `false`, respectively,
- T, t are used for time related sets and variables, respectively,

patient 1			patient 2			patient 3			patient 4		
I	T	A	I	T	A	I	T	A	I	T	A
1	234	a	2	57	f	3	186	h	4	33	a
1	234	b	2	63	g	3	186	i	4	93	k
1	234	c	2	74	g	3	186	a			
1	234	d	2	78	e	3	186	j			
1	237	e	2	78	g	3	186	e			
						3	199	a			
						3	199	e			

Table 2: A small anonymized piece of the database

- we also introduce the symbol $\mathcal{D}_{\mathsf{R}}(\mathsf{X})$, which denotes the domain of an attribute set X in a relation R, i.e. $\mathcal{D}_{\mathsf{R}}(\mathsf{X}) = \{\mathsf{r}[\mathsf{X}]|\mathsf{r} \in \mathsf{R}\}$.

Let the input database be defined as follows, the definition is analogous to the one in [15].

Definition 3 (Sequence database) *Let* $\mathsf{A} = \{\mathsf{a}_1, \mathsf{a}_2, \ldots, \mathsf{a}_n\}$ *be a finite set of items, where* $n \geq 1$, T *is a non-empty set of timestamps, and* I *is a non-empty set of unique identifiers. Let a relation* R *be defined over* $\mathsf{I} \times \mathsf{T} \times \mathsf{A}$, *i.e.* $\mathsf{R} \subseteq \mathsf{I} \times \mathsf{T} \times \mathsf{A}$, *then* R *is a sequence database. For simplicity and better understanding, we use the notion* $\mathsf{R}(\mathsf{ITA})$ *to express* R *is determined by sets* I, T, *and* A, *i.e.* $\mathsf{R} \subseteq \mathsf{I} \times \mathsf{T} \times \mathsf{A}$ *in that order. If* $|\mathrm{I}| = 1$ *in a sequence database* $\mathsf{R}(\mathsf{ITA})$, *then* R *is called a sequence database.*

Example Table 2 shows a small set of records from the anonymized health care database we use in this paper. The columns of the table reflect relation R. The twelve columns of the table are organized into four groups of three columns. Each column group represents a patient. The first column in a group is a patient identifier, the real identifier is replaced with an integer number. The second column in a group is a timestamp, the real date is replaced with an integer number. The third column in a group contains the items, the real treatment codes are transformed to letters of the alphabet.

Definition 4 (Ordering of items) *Let a binary relationship* $\leq: \mathsf{R} \times \mathsf{R} \to \{\top, \bot\}$ *be defined on sequence databases such that the ordering of elements of* R *is determined by the natural ordering over* T. $\leq$ *is an ordering, i.e. it is transitive, antisymmetric, reflexive, and total. For simplicity, we also use* $\leq$

on relations such that if $S_1, S_2 \subseteq R$ *and* $S_1, S_2 \neq \emptyset$*, then* $S_1 \leq S_2$ *if and only if* $\forall s_1 \in S_1 \forall s_2 \in S_2 : s_1 \leq s_2$.

In other words, ordering of items in sequence databases are based on time related attributes. If time representation in sequence database does allow a clear distinction between when two events have happened, then we assume they are simultaneous events.

Definition 5 (Sequence) *Let* $R(ITA)$ *be a sequence database, and* $\mathcal{S} = < S_1, S_2, \ldots S_n >$ *be defined as an ordered set of relations where* $\forall i : 1 \leq i \leq n \implies S_i \subseteq R$ *such that*

$$S_i, S_j \in \mathcal{S} : 1 \leq i < j \leq n \implies S_i \leq S_j, \neg S_j \leq S_i.$$

We say $\mathcal{S}$ *is a sequence if and only if*

$$\begin{aligned} \forall r, s :\ & r \in S_i, s \in S_j \implies r[I] = s[I] \\ \forall r, s :\ & \Big(r \in S_i, s \in S_j \implies r[T] = s[T] \Big) \iff i = j \end{aligned}$$

for all $S_i, S_j \in \mathcal{S}$*. Since the identifiers are the same in the sequence, and there are itemsets that share the same timestamps, we use the representation* $\mathcal{S} =< A_{t_k}, \ldots, A_{t_l} >$ *for better readability where* $t_i \in T$*, where* $A_{t_i} \subseteq A$ *is a set of elements indexed by their shared timestamps. We also introduce the following notions:*

- $|\mathcal{S}| = n$ *denotes the length (number of relations) of the sequence,*
- $U(\mathcal{S})$ *stands for the shared identifier in* $\mathcal{S}$,
- *and* $\tau(S_i)$ *(or* $\tau(A_i)$*) for the shared timestamp in* S_i *where* $1 \leq i \leq n$.

This means that in sequence $\mathcal{S}$ all elements share the same identifier, and within an A_{t_i} itemset the elements are unordered as they are considered to have occurred simultaneously. If an itemset A_{t_i} precedes A_{t_j} in $\mathcal{S}$ ($A_i \leq A_j$), then all items in A_{t_i} precede any item in A_{t_j}. We also introduce operators on sequences to deal with more complex problems.

Definition 6 (Operators on sequences) *Let* $\mathcal{S}_1 =< A_{t_k}, \ldots, A_{t_l} >$ *and* $\mathcal{S}_2 =< A_{t_m}, \ldots, A_{t_n} >$ *be two sequences defined on* $R(ITA)$*. We say*

- $\mathcal{S}_1$ *is a proper subsequence of* $\mathcal{S}_2$ *denoted by* $\mathcal{S}_1 \sqsubseteq \mathcal{S}_2$ *if and only if* $U(\mathcal{S}_1) = U(\mathcal{S}_2)$ *and* $\forall A_{t1} \exists A_{t2} : A_{t1} \in \mathcal{S}_1, A_{t2} \in \mathcal{S}_2 \implies A_{t1} \subseteq A_{t2}, \tau(A_{t1}) = \tau(A_{t2})$,

- $\mathcal{S}_1$ *is a subsequence of* $\mathcal{S}_2$ *denoted by* $\mathcal{S}_1 \preceq \mathcal{S}_2$ *if and only if for all* a_1, a_2, *and* $t1, t2 \in T$ *there exist* $t3, t4 \in T$ *such that*

$$a_1 \in A_{t1}, a_2 \in A_{t2}, t1 \leq t2 \implies a_1 \in A_{t3}, a_2 \in A_{t4}, t3 \leq t4$$

  where $A_{t1}, A_{t2} \in \mathcal{S}_1$, *and* $A_{t3}, A_{t4} \in \mathcal{S}_2$,

- *the union of sequences for which* $U(\mathcal{S}_1) = U(\mathcal{S}_2)$ *denoted by* $\mathcal{S}_1 \cup \mathcal{S}_2$ *is defined as an* $\leq$*-ordering preserving merge of these sets such that if* $A_{t1} \in \mathcal{S}_1$, $A_{t2} \in \mathcal{S}_2$ *and* $\tau(A_{t1}) = \tau(A_{t2})$ *then the resulting* $A_t = A_{t1} \cup A_{t2}$,

- *the intersection of sequences for which* $U(\mathcal{S}_1) = U(\mathcal{S}_2)$ *denoted by* $\mathcal{S}_1 \cap \mathcal{S}_2$ *is defined as the largest possible subsequence* $\mathcal{S}$ *in number of items for which* $\mathcal{S} \sqsubseteq \mathcal{S}_1$ *and* $\mathcal{S} \sqsubseteq \mathcal{S}_2$,

- *the difference of the sequences for which* $U(\mathcal{S}_1) = U(\mathcal{S}_2)$ *denoted by* $\mathcal{S}_1 \backslash \mathcal{S}_2$ *is defined as the largest possible subsequence* $\mathcal{S}$ *in number of items for which* $\mathcal{S} \sqsubseteq \mathcal{S}_1$ *and there is no* $\mathcal{S}'$ *such that* $\mathcal{S}' \sqsubseteq \mathcal{S}$ *and* $\mathcal{S}' \sqsubseteq \mathcal{S}_2$ *if and only if* $\mathcal{S}_1$ *is not a subsequence of* $\mathcal{S}_2$. *The difference does not exist otherwise.*

- $\mathcal{S}_2$ *is the prefix cut of* $\mathcal{S}_1$ *by an item* $a \in A$ *denoted by* $\varphi(\mathcal{S}_1, a)$ *if and only if* $\mathcal{S}_2 \sqsubseteq \mathcal{S}_1$ *and if there exists* $A_t \in \mathcal{S}_1$ *such that* A_t *is a set for which* $a \in A_t$ *then* $\max_{A_i \in \mathcal{S}_2}(\tau(A_i)) \leq \tau(A_t)$. *In this paper, we call maximum cut of* $\mathcal{S}_1$ *by* $a \in A$ $(\Phi(\mathcal{S}_1, a))$ *the union of all possible prefix cuts of* $\mathcal{S}_1$ *by* $a \in A$.

Notice that, there can be many different sequences with the same identifier according to Definition 5, and there is no sequence with the length of 0. It is easy to prove that the maximal number of closed sequences in a sequence database $R(ITA)$ equals to $\mathcal{D}_R(I)$, hence every closed sequence has a natural identifier: the elements of set I in our database.

Definition 7 (Closed sequences) *Let a sequence* $\mathcal{S} =< S_1, S_2, \ldots, S_n >$ *be defined on a sequence database* $R(ITA)$. *We say* $\mathcal{S}$ *is a closed sequence and it is denoted by* $\overline{\mathcal{S}}$ *if and only if*

$$\forall r \exists S_i \; r \in R, S_i \in \mathcal{S}, r[I] = U(\mathcal{S}) \implies r \in S_i$$

for some $1 \leq i \leq n$. *The largest possible set of closed sequences in* $R(ITA)$ *is called macseq and it is denoted with* Σ.

Example 8 *Table 3 shows the records of Table 2 transformed into the form used for representing sequence databases in the literature. The first column is the sequence identifier, which comes from the patient identifying* I *attribute of Table 2. The second column contains the sequences, where each sequence is a comma separated list of itemsets shown in braces. The ordering of the itemsets is determined by attribute* T*. If the* T *value is identical for two* A *items, then those appear in the same itemset.*

I	Σ
1	$<(a,b,c,d),(e)>$
2	$<(f),(g),(g),(e,g)>$
3	$<(h,i,a),(j,e)>$
4	$<(a),(k)>$

Table 3: Relation R transformed to a sequence database

Sequence $<(a),(e)>$ *is a subsequence of both the sequence identified by* $\mathrm{I}=1$ *and* $\mathrm{I}=3$*. In both sequences itemset* (a) *is a subset of the first itemset, and* (e) *is a subset of the second itemset.*

The following terms are the foundations for capturing the proper concept of frequent sequences:

Definition 9 (Support of sequences) *We introduce the following support metrics for a sequence* $\mathcal{S} =< S_1, S_2, \ldots, S_n >$ *defined on* $\mathsf{R}(\mathsf{ITA})$*:*

- *the support of* $\mathcal{S}$ *denoted by* $\mathrm{supp} : \mathcal{S} \to [0,1]$*:*

$$\mathrm{supp}(\mathcal{S}) = \frac{||\{\mathcal{S}_i \mid \mathcal{S}_i \in \Sigma, \mathcal{S} \preceq \mathcal{S}_i\}||}{||\Sigma||},$$

where $||S||$ *stands for the number of elements in set* S*,*

- *the conditional support of* $\mathcal{S}$ *assuming there is a* $\mathcal{S}_i \in \Sigma$*, which has an element containing* $a \in A$ *is*

$$\mathrm{supp}(\mathcal{S}|a) = \frac{||\{\mathcal{S}_i \mid \mathcal{S}_i \in \Sigma, \mathcal{S} \preceq \Phi(\mathcal{S}_i, a)\}||}{||\{\mathcal{S}_i \mid \mathcal{S}_i \in \Sigma, \exists A_t \in \mathcal{S}_i, a \in A_t\}||}.$$

If there is no sequence in Σ *that contains* a*, then let* $\mathrm{supp}(\mathcal{S}|a) = 0$*,*

- *the conditional unsupport of* $\mathcal{S}$ *assuming there is a* $\mathcal{S}_i \in \Sigma$ *that contains an item* $a \in A$ *is*

$$\mathrm{supp}(\mathcal{S}|\neg a) = \frac{\|\{\mathcal{S}_i \mid \mathcal{S}_i \in \Sigma, \mathcal{S} \preceq \mathcal{S}_i, \forall A_t \in \mathcal{S}_i : \ a \notin A_t\}\|}{\|\{\mathcal{S}_i \mid \mathcal{S}_i \in \Sigma, \forall A_t \in \mathcal{S}_i, a \notin A_t\}\|}.$$

If each sequence in Σ *contains* a*, then let* $\mathrm{supp}(\mathcal{S}|\neg a) = 0$*.*

Example 10 *Table 4 show the support of sequences with length of one based on Table 3. Items* a *and* e *occur in three different sequences. Though* g *has three occurrences as well, those are limited to a single sequence.*

Σ	$\mathrm{supp}(\mathcal{S})$
a	3
b	1
c	1
d	1
e	3
f	1
g	1
h	1
i	1
j	1
k	1

Table 4: Support of items

Theorem 11 *Let* $\mathcal{S}_1 =< A_{t_1}, \ldots, A_{t_n} >$*, and* $\mathcal{S}_2 =< A_{t_1}, \ldots, A_{t_{n-1}} >$ *be two sequences defined on* $\mathsf{R}(\mathsf{ITA})$*, then*

$$\forall a \in A_{t_n} : \ \mathrm{supp}(\mathcal{S}_1) \leq \mathrm{supp}(\mathcal{S}_2|a).$$

Proof. According to Definition 9, $\mathrm{supp}(\mathcal{S})$ equals to number closed sequences that contain $\mathcal{S}$ as a pattern divided by the number of all closed sets. Firstly, let assume that A_{t_n} consists of a single item a. In that case, the numerator of $\mathrm{supp}(\mathcal{S}_1)$ and $\mathrm{supp}(\mathcal{S}_2|a)$ is the same since $\mathcal{S}_1$ equals to $\mathcal{S}_2$ followed by an a. The denominator is different: for $\mathrm{supp}(\mathcal{S}_2|a)$ it is the number of closed sequences containing a in one of its elements, which must be less or equal than the number of all closed sequences. As a consequence, $\mathrm{supp}(\mathcal{S}_1) \leq \mathrm{supp}(\mathcal{S}_2|a)$.

If A_{t_n} contains more than one element, then $\mathrm{supp}(\mathcal{S}_1)$ is determined by the least frequent item in A_{t_n}. That is, the numerator of $\mathrm{supp}(\mathcal{S}_2|a)$ must be greater or equal to one of $\mathrm{supp}(\mathcal{S}_1)$ also leads to $\mathrm{supp}(\mathcal{S}_1) \leq \mathrm{supp}(\mathcal{S}_2|a)$. $\square$

4 Relevance base candidate selection

The problem of frequent sequential pattern discovery has been defined in [1]: *Given a set of sequences, where each sequence consists of a list of elements and each element consists of a set of items, and given a user-specified min_support threshold, sequential pattern mining is to find all frequent subsequences, i.e., the subsequences whose occurrence frequency in the set of sequences is no less than min_support.* This definition originally targeted the mining of database transactions, however this very same definition can be applied to a much wider range of problems. In our case, the set of sequences are identified by attribute I in the elements of relation R. The elements of sequences are ordered by attribute T, and hence form a list. As it is possible for elements of R to have the same $\mathsf{t} \in \mathsf{T}$ value, the elements of that list are not individual items, but a set of items.

Frequent sequence construction is based on the hypothesis that all subsequences of a frequent sequence are frequent sequences themselves, formally if $\mu \leq \text{supp}(\mathcal{S})$, then $\forall \mathcal{S}' \preceq \mathcal{S} \implies \mu \leq \text{supp}(\mathcal{S}')$ where $\mu \in [0, 1]$ stands for min_support. This assumption makes it possible to build a lattice of subsequences. The lattice can be constructed bottom-up with increasing length on the pattern of the a priori algorithm or by combining frequent subsequences already detected based on their prefixes.

4.1 Problem statement

Average Apriori like algorithms are quasi linear whenever the size of frequent itemsets are small; polynomial in the sum of the size of the input (transactions) plus output (frequent patterns) [11]. That is, if every shopper buys every item, the algorithm must output each subset of A items. The basic characteristics do not change for sequence mining using distributed Apriori-like algorithms [2], however, the size of the input is multiplied by the length of sequences. In other words, sequence mining algorithms are exponential for long sequences or large inputs.

As Figure 1 indicates for the case of sequential databases that contain very long sequences, an improvement is necessary in the candidate generation for sequential pattern mining algorithms. Counting the number of occurrences may be simply infeasible for mining some real life datasets, like in health care database.

Moreover, interesting patterns are not necessarily frequent ones. If some items are frequent enough among sequences by themselves independently from others, then frequent pattern mining algorithms always include those in almost all elements of the output, which increases the size of the output, and decreases significantly the interestingness of such patterns. As a consequence, the problem is how to improve the computational performance with or by increasing the interestingness of patterns, or to be more precise: how to improve the overall performance by pruning non-relevant or independent consumptions.

Consider a database of medical diagnoses; anamneses can easily be constructed by building sequences joining diagnoses/treatment by patient identifiers. Almost all frequent sequences contain diagnoses frequent in the population such as flu or hypertension that are not necessarily relevant in general for the course of the main case. Such items should not be considered when building frequent sequences and provide a basis for prefiltering. We call the prefiltered set of frequent sequences frequent-and-relevant sequences.

In this section, we propose a new method for calculating importance metrics, which combine relevance and support of a sequence when generating candidate sequences. The basic idea is that candidate generation and prefiltering of the sequences should take place at the same time to reduce the search space and hence the computational complexity.

The idea for prefiltering in our support calculation method comes from text mining where the TF-IDF [12] metric has been successfully used to connect different documents based on their contents. The sequence metric is similar to the SIF-IDF metric defined in [9] for protecting sensitive data in databases.

Importance metric of a pattern is derived from two values associated with two sequences generated by the pattern. For a pattern we maintain two sets of key-value pairs: one in which support values are calculated on closed sequences of identifiers that appear in the pattern, and another one set of those that do not. The former one is called *frequent set*, the latter is called *inverse set*. The normalized rate of relative occurrences of an item in the frequent and inverse sets is a suitable parameter for that item for filtering when generating a new candidate sequence.

In the next section, we give the definitions necessary for formalizing this idea.

4.2 Definitions

Definition 12 (Frequent set of a pattern) *Let $\mathcal{S}$ be a pattern over a relation* $\mathsf{R}(\mathsf{ITA})$*, and* F *be a function which maps sequences to a set of a set of item-number pairs such that*

$$\mathsf{F}(\mathcal{S}) = \{(\mathfrak{a}, \operatorname{supp}(\mathcal{S}|\mathfrak{a}))|\ \mathfrak{a} \in \mathsf{A}\}.$$

Definition 13 (Inverse set of a pattern) *Let $\mathcal{S}$ be a pattern over a relation* $\mathsf{R}(\mathsf{ITA})$, *and* $\overline{\mathsf{F}}$ *be a function which maps sequences to a set of a set of item-number pairs such that*

$$\overline{\mathsf{F}}(\mathcal{S}) = \{(\mathfrak{a}, \mathrm{supp}(\mathcal{S}|\neg\mathfrak{a})) \mid \mathfrak{a} \in \mathsf{A}\}.$$

$\mathsf{F}(\mathcal{S})$ and $\overline{\mathsf{F}}(\mathcal{S})$ contain information on each item, e.g. $(\mathfrak{a}, \mathfrak{n}) \in \mathsf{F}(\mathcal{S})$, and $(\mathfrak{a}, \mathfrak{m}) \in \overline{\mathsf{F}}(\mathcal{S})$. Notice that, a correlation between values $\mathfrak{n}$, $\mathfrak{m}$ might indicate relevance. If $\mathfrak{m} \simeq 0$ and $\mathfrak{n} \geq \mu$ then $\mathcal{S} \to \mathfrak{a}$ (i.e. $\mathcal{S}$ is followed by $\mathfrak{a}$) show high correlation, which means that the presence of $\mathcal{S}$ as a series of events highly suggests $\mathfrak{a}$ to be happening. If $\mathfrak{m} \geq \mu$ and $\mathfrak{n} \simeq 0$, then $\mathcal{S}$ and $\mathfrak{a}$ show high inverse correlation, i.e. the pattern of $\mathcal{S}$ almost always inhibits the event $\mathfrak{a}$ to happen.

Let $\mathsf{F}(\mathcal{S})[\mathfrak{a}] = \mathfrak{n}$ be a shorthand for the fact that $(\mathfrak{a}, \mathfrak{n}) \in \mathsf{F}(\mathcal{S})$. Let $\mathsf{E}(\mathsf{F}(\mathcal{S}))$, $\mathrm{Var}(\mathsf{F}(\mathcal{S}))$, and $\mathrm{Sum}(\mathsf{F}(\mathcal{S}))$ be the mean, deviation, and the sum of $\mathsf{F}(\mathcal{S})[\mathfrak{a}]$ values, respectively, for all $\mathfrak{a} \in \mathsf{A}$.

Definition 14 (Relevance measure) *Let* Imp *be defined as an importance measure on a sequence $\mathcal{S}$, and* $\mathfrak{a} \in \mathsf{A}$ *item of a relation* $\mathsf{R}(\mathsf{ITA})$ *such that*

$$\mathrm{Imp}(\mathcal{S}, \mathfrak{a}) = \begin{cases} 0 & \text{if } \mathrm{Sum}(\mathsf{F}(\mathcal{S}))\mathrm{Sum}(\overline{\mathsf{F}}(\mathcal{S})) = 0 \\ \dfrac{\left| \dfrac{\mathsf{F}(\mathcal{S})[\mathfrak{a}]}{\mathrm{Sum}(\mathsf{F}(\mathcal{S}))} - \dfrac{\overline{\mathsf{F}}(\mathcal{S})[\mathfrak{a}]}{\mathrm{Sum}(\overline{\mathsf{F}}(\mathcal{S}))} \right|}{\max\left(\dfrac{\mathsf{F}(\mathcal{S})[\mathfrak{a}]}{\mathrm{Sum}(\mathsf{F}(\mathcal{S}))}; \dfrac{\overline{\mathsf{F}}(\mathcal{S})[\mathfrak{a}]}{\mathrm{Sum}(\overline{\mathsf{F}}(\mathcal{S})]} \right)} & \text{otherwise} \end{cases},$$

where $|\mathfrak{n}|$ *stands for the absolute value of a number* $\mathfrak{n}$. *We say $\mathcal{S}$ is a relevant antecedent of* $\mathfrak{a}$ *if* $\mathrm{Imp}(\mathcal{S}, \mathfrak{a})$ *is greater or equal to a certain threshold.*

Relevance measure indicates that there is a connection between the frequency and rareness of an item, that is, if an item occurs in every sequence or that item occurs in no sequences, then relevance is equally 0 according to Definition 14. Nevertheless, if there is an item $\mathfrak{a}$ which always appears before or together with an item $\mathfrak{b}$ then relevance is 1 because $\mathsf{F}(\mathcal{S}, \mathfrak{b}) = 1$ and $\overline{\mathsf{F}}(\mathcal{S}, \mathfrak{b}) = 0$, where $\mathcal{S}$ consists of a single itemset that has a single value $\mathfrak{a}$ ($\mathcal{S} =< \{\mathfrak{a}\} >$ for short). By symmetry, if $\mathfrak{a}$ never occurs together or before $\mathfrak{b}$ in any sequences then relevance is still 1 indicating some kind of rejection or inhibition. For example, those who buy lactose free products will not buy milk or cheese. Also notice that, relevance both measures frequencies and infrequences, which leads to a re-formulation how important an item $\mathfrak{a}$ regarding a preliminary series of events $\mathcal{S}$. That is, it is a potential relative significance measure (see Definition 2).

Definition 15 (Importance measure) *Let* Ind *be defined as a measure on* $\mathcal{S}$ *sequences, and* $a \in A$ *items of a relation* $R(ITA)$ *such that*

$$\mathrm{Ind}(\mathcal{S}, a) = \frac{\mathrm{Imp}(\mathcal{S}, a) - E_{a \in A}(\mathrm{Imp}(\mathcal{S}, a))}{\mathrm{Var}_{a \in A}(\mathrm{Imp}(\mathcal{S}, a))}$$

We say $\mathcal{S}$ *is an import antecedent of* a *if* $|\mathrm{Ind}(\mathcal{S}, a)|$ *is greater or equal to a certain threshold.*

Importance is a normalized value of relevance to measure how much $\mathcal{S} \to a$ is unusual. In most of the cases, mean value of relevance shall be about 0, i.e. occurrences of items are independent in general. Statistically, if absolute value of the $\mathrm{Ind}(\mathcal{S}, a) \geq 3$ (the triple of the variance), then it is an outlier value that is usually a strong indicator for a deep connection between variables.

We propose Algorithm 1 for identifying important sequences in the sequence database R. The inputs of the algorithm are the database R itself, a μ minimum support threshold, and a ν importance threshold. The output is the set of frequent-and-relevant (important) sequences Σ_f. In the body of the algorithm, a loop variable k, the frequent set $F(\mathcal{S})$, the inverse set $\overline{F}(\mathcal{S})$, and the set of new sequences Σ_c are used locally.

The algorithm works as follows. In the initialization phase (line 1), we add items as sequences of length 1 to the Σ_f set, if their support is over the minimum threshold μ. The main loop iterates over the sequences of maximum length. First, it removes all elements from the Σ_f new important sequence set (line 8). It computes the frequent $F(\mathcal{S})$ and the inverse $\overline{F}(\mathcal{S})$ sets for the current $\mathcal{S}$ sequence (line 10). If there are candidate postfix items, then we iterate over it, and filter the sequences with the formula of Definition 15. As threshold, we utilize ν an importance threshold input parameter (line 11). If the importance is over that threshold, then that item c is appended to the end of $\mathcal{S}$ (line 12), and the new sequence is added to the important set of sequences (line 13). The main loop is repeated until the Σ_c set generated is not an empty set. If no further candidate sequences can be generated, the algorithm returns the Σ_f set (line 17), otherwise the elements if Σ_c are added to Σ_f (line 18). If all sequences are processed, the k maximum length loop variable is increased (line 20), and the main loop is restarted.

Lemma 16 *Algorithm 1 identifies all frequent patterns, which have a support greater or equal to a min_support according to Definition 9, but the important ones are returned.*

```
input  : $\mathsf{R(ITA)}$ database, $\mu$ minimum support threshold, $\nu$
         importance threshold
output: $\Sigma_f$ set of important sequences
data   : $k$ cycle variable, $\Sigma_c$ set of new sequences, $\mathsf{F}(\mathcal{S})$ frequent next
         item set, $\overline{\mathsf{F}}(\mathcal{S})$

/* Initialization                                              */
$k := 1$;
$\Sigma_f = \{\mathcal{S} | a \in A, \mathcal{S} = <\{a\}_{-\infty}>, \mathrm{supp}(\mathcal{S}) \geq \mu\}$;
/* Main loop                                                   */
while true   :
do
    foreach $\mathcal{S} \in \Sigma_f$ where $\mathrm{len}(\mathcal{S}) = k$ do
        $\Sigma_c := \emptyset$;
        foreach $a \in A$ do
            Compute the sets $\mathsf{F}(\mathcal{S}, a)$ and $\overline{\mathsf{F}}(\mathcal{S}, a)$;
            if $|\mathrm{Ind}(R, \mathcal{S}, c)| \geq \nu$ then
                $S' := \mathrm{concat}(\mathcal{S}, c)$;
                $\Sigma_c := \Sigma_c \cup \{S'\}$;
            end
        end
        if $\Sigma_c = \emptyset$ then return $\Sigma_f$;
        ;
        $\Sigma_f := \Sigma_f \cup \Sigma_c$;
    end
    $k := k + 1$;
end
```

Algorithm 1: Importance based frequent sequential pattern generation

According to Theorem 11, conditional support $\mathrm{supp}(\mathcal{S}|a)$ is greater or equal to the $\mathrm{supp}(\mathcal{S} \to a)$. It means, that by generating $\mathsf{F}(\mathcal{S})$ Algorithm 1 finds all candidates for which support is greater or equal to a certain threshold. However, if either a or $\mathcal{S}$ is independent, or too frequent in general, it entails $\overline{\mathsf{F}}(\mathcal{S}, a) \simeq \mathsf{F}(\mathcal{S}, a)$ and as such is omitted from the output. As a consequence, Algorithm 1 is a one-step method to find frequent *and* relevant patterns.

Lemma 17 (Infrequent important candidate) *Algorithm 1 can identify important sequences with regard to the ν importance threshold that are not frequent regarding a μ threshold.*

Definition 15 is independent from the minimum support threshold μ, so it is possible to construct an example, where the statement of Lemma 17 holds. If $\Sigma = \{<\{a\},\{b\},\{c\}>, <\{c\},\{d\}>\}$, and $\mu = 60\%$, then subsequence $<\{a\},\{b\}>$ can not be frequent as it occurs only in the first closed sequence. However, it is important because $|\mathrm{Ind}(<\{a\}, b>)| \geq \nu$ for an appropriate ν because b is always preceded by a.

Algorithm 1 builds important sequences on the pattern of GSP. The number of database scans is two times the number of important sequences identified: the computation of sets $F(\mathcal{S})$ and $\overline{F}(\mathcal{S})$ requires a scan each. With regard to candidate generation and data structure efficiency, there is a lot of room for improvements.

Lemma 18 (Candidate generation) *All subsequences of important sequences generated by Algorithm 1 are important sequences.*

Lemma 18 gives a property similar to that exists in case of sequential pattern generation algorithms, and this way patterns can not only be grown, but joined as well.

5 Empirical analysis

5.1 Application to medical data

In this section, we present the experiments we conducted on real-life clinical data. The clinical database was anonymized [5] before use.

We defined one sequence for each unique patient identifier, i.e. the I set comprises patient identifiers. Treatments and diagnoses have unique medical codes that define the A itemset. Treatment and diagnosis timestamps are aggregated on daily level, that is, two treatments that happened on the same day are considered to be simultaneous and have the same $t \in T$ element associated with them.

The properties of the data set are shown in Table 5. The total number of patient records is about 67 million, which is the size of the relation in the context of this paper. The patient cases, which is equal to the number of natural identifiers, is in the order of 10^5. The average number of examinations of a patient is in the order of 10^2, this value is the average number of items in a sequence. The average number of days when examinations are performed or diagnoses are given on a patient is around 6, this value is the average number of itemsets in the maximum sequences. The number of treatments, i.e. the number of items is around 10^4.

Number of records in the database	66870306
Number of natural identifiers	455514
Average length of maximum sequences	146.8
Average size of maximum sequences	6.18
Number of items	9291

Table 5: Properties of the data set

5.2 Experimental results

The experiments we conducted on an Oracle Sun Server X3-2 with 256GB RAM and 32 cores of 4 Intel Xeon E5-2660 CPUs. The mining processes were allowed to use up to 48GB of RAM and 200GB of disk space.

As the experiments shown in Section 2.5 use the API of [4], where the algorithms are implemented in Java, we used the Java implementation of our method. We experimented with the implementation of PrefixSpan provided by [14], however that run out of the 200GB disk space limit before finishing. The minimum support threshold was set to the absolute value of 10 occurrences in all cases. In REVIEW, we used a 3 as the importance threshold to provide output sets of similar size as the other two algorithms for short sequences. User time usage and memory usage were both measured with the UNIX command `time`.

Since the preliminary experiments with PrefixSpan and SPADE have shown that these algorithms are not able to process this amount of data within reasonable time, once again we have used a random sample and limited the maximum length of sequences to 5, 10, 20, 40, 60, 80, 100, 120 and 140. The highest value used is still below the average length of sequences in the whole database. Table 6 shows the properties of these samples.

Figure 2 compares REVIEW with PrefixSpan and SPADE over the same dataset with the same minimum support threshold settings. The figures show how the execution time requirements and the disk space usage scale as the maximum length of input sequences grows. The number shown is the average of three runs. The algorithms are deterministic, so the number of frequent sequences does not vary. We represent the output sequences in the same form

Max. length	Sequences	Length	Itemsets
5	2150	5048	3441
10	4082	18064	28255
20	6416	50781	17379
40	8979	124289	36448
60	10499	197817	53894
80	11571	271818	71685
100	12263	333065	86099
120	12789	390307	100058
140	13186	441514	112012

Table 6: Properties of the sample data sets

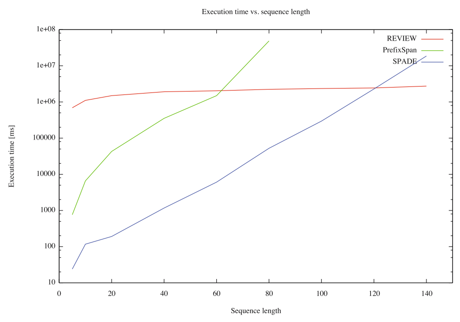
on the disk, so we consider that the number of output sequences is proportional to the disk usage.

REVIEW has been found to scale better as the length and number of input sequences grow than Preview and PrefixSpan. The chart on the left shows that though REVIEW has a high initial time requirement, however it does have a much lower gradient on the log scale than the other two. Around the sequence length of 60 REVIEW becomes quicker than PrefixSpan, and around the sequence length of 120 it surpasses SPADE in speed. Though REVIEW works over the same search space as shown in Theorem 11, it is more effective in filtering frequent sequence candidates than the other algorithms, and yet it keeps the relevant information.

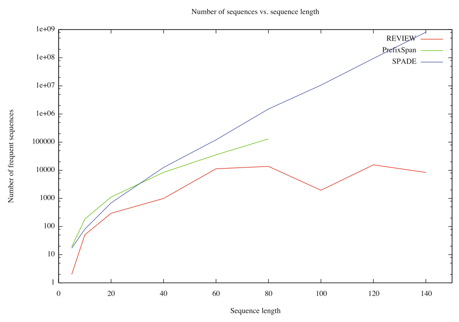
There is no correlation between the memory consumption and the efficiency of the algorithms in case of REVIEW and SPADE. The PrefixSpan implementation used up all of the available memory, while the other two remained well below the limit. In the latter cases, memory consumption seems to depend rather on the Java virtual machine, than the complexity of the algorithm.

6 Conclusions

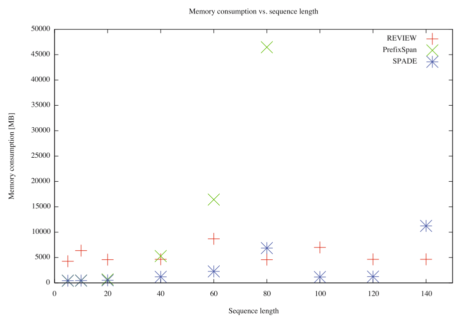
Many studies have elaborated sequential pattern mining methods to improve the overall performance because of the time complexity issues. However, a problem arises when the length of the frequent sequences increases or the number of apriori frequent items is high enough. The previously developed sequential pattern mining algorithms address the performance issue only regardless of whether a pattern with two or more items correlate somehow or



(a) Sequence length vs. execution time



(b) Sequence length vs. frequent sequences



(c) Sequence length vs. memory consumption

Figure 2: Performance of REVIEW against PrefixSpan and SPADE. The maximum length in a sequence database vs. the execution time, the number of frequent sequences and memory consumption

they co-occurrence is frequent because their apriori frequencies are independently high among customers. On the other hand, sequential pattern mining algorithms often ignore niche segments' patterns due to their relative infrequencies.

In this paper, we proposed REVIEW, a new approach how to deal with frequent closed sequences. It iteratively calculates the conditional frequencies of patterns and their possible follow-ups for those closed sequences in which a follow-up appears, and those in which it does not. If measures show statistically significant differences then pattern is extended by the follow-up item, and it is found to be important, and a new cycle with the extended sequence begins. The algorithm stops when there can be made no extensions.

We proved that this method finds all relevant and frequent sequential patterns in linear time regarding the number of closed sequences. We demonstrated by experiments that our method significantly improves performance of those known from literature on a health care database where both independent, apriori frequent items, and long sequences are both present at the same time. Moreover, REVIEW also pointed out that not so frequent diagnoses show strong correlation with others which would be missed by other methods.

Acknowledgement

Publishing of this work and the research project were funded by the European Union and co-financed by the European Social Fund under the name of ,,MEDICSPHERE – Complex, multipurpose, ICT technology driven medical, economic, and educational use of clinical data" and grant number TÁMOP-4.2.2.A-11/1/KONV-2012-0009.

References

[1] R. Agrawal, R. Srikant, Mining sequential patterns, *Proc. Eleventh International Conference on Data Engineering*, Taipei, Taiwan, 1995, pp. 3–14. ⇒300

[2] L. M. Aouad, Nhien-An Le-Khac, T. M. Kechadi, Performance study of distributed apriori-like frequent itemsets mining, *Knowledge and Information Systems*, **23**, 1 (2009) 55–72. ⇒300

[3] J. Ayres, J. Gehrke, T. Yiu, J. Flannick, Sequential pattern mining using bitmaps, *Proc. Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Canada, July 2002, pp. 429–435. ⇒291

[4] P. Fournier-Viger, SPMF – an open-source data mining library, 2014. ⇒ 291, 306

[5] T. Z. Gál, G. Kovács, Z. T. Kardkovács, Survey on privacy preserving data mining techniques in health care databases, *Acta Univ. Sapientiae, Informatica*, **6,** 1 (2014) 33–55. ⇒305

[6] L. Geng, H. J. Hamilton, Interestingness measures for data mining: A survey, *ACM Computing Surveys (CSUR)*, **38,** 3 (2006) ⇒292, 293, 294

[7] K. Gouda, M. Hassaan, Mining sequential patterns in dense databases, *International Journal of Database Management Systems (IJDMS)*, **3,** 1 (2011) 179–194. ⇒291

[8] J. Han, J. Pei, Y. Yin, Mining frequent patterns without candidate generation, *Proc. International Conference Management of Data (ACM-SIGMOD '00)*, Dallas, USA, May 2000, pp. 1–12. ⇒290

[9] T. P. Hong, C. W. Lin, K. T. Yang, S. L. Wang, A heuristic data-sanitization approach based on TF-IDF, *Proc. 24th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems*, *Lecture Notes in Artificial Intelligence* **6703** (2011) 156–164. ⇒301

[10] K. McGarry, A survey of interestingness measures for knowledge discovery, *The Knowledge Engineering Review*, **20,** 1 (2005) 39–61. ⇒292

[11] P. W. Purdom, D. Van Gucht , D. P. Groth, Average-case performance of the apriori algorithm, *SIAM Journal on Computing*, **33,** 5 (2004) 1223–1260. ⇒300

[12] G. Salton, E. A. Fox, H. Wu, Extended boolean information retrieval, *Communications of ACM*, **26,** 12 (1983) 1022–1036. ⇒288, 301

[13] R. Srikant, R. Agrawal, Mining sequential patterns: Generalizations and performance improvements, *Proc. 5th International Conference on Extending Database Technology: Advances in Database Technology (EDBT '96)*, *Lecture Notes in Security and Cryptology* **1057**, (1996) 3–17. ⇒290

[14] Y. Tabei, An imprementation of PrefixSpan (prefix-projected sequential pattern mining), 2008. ⇒306

[15] M. J. Zaki, SPADE: An efficient algorithm for mining frequent sequences, *Machine Learning*, **42,** 1–2 (2001) 31–60. ⇒290, 295

Received: September 11, 2014 • Revised: November 10, 2014