NegPSpan: efficient extraction of negative sequential patterns with embedding constraints

Thomas Guyet, René Quiniou

NegPSpan: efficient extraction of negative sequential patterns with embedding constraints

Thomas Guyet – Agrocampus-Ouest/IRISA UMR6074
René Quiniou, Univ Rennes, Inria, CNRS, IRISA

August 20, 2018

Abstract

Mining frequent sequential patterns consists in extracting recurrent behaviors, modeled as patterns, in a big sequence dataset. Such patterns inform about which events are frequently observed in sequences, *i.e.* what does really happen. Sometimes, knowing that some specific event does not happen is more informative than extracting a lot of observed events. Negative sequential patterns (NSP) capture recurrent behaviors by patterns containing both observed events and absent events. Few approaches have been proposed to mine such NSPs. In addition, the syntax and semantics of NSPs differ in the different methods which makes it difficult to compare them. This article provides a unified framework for the formulation of the syntax and the semantics of NSPs. Then, we introduce a new algorithm, NegPSpan, that extracts NSPs using a PrefixSpan depth-first scheme and enabling *maxgap* constraints that other approaches do not take into account. The formal framework allows for highlighting the differences between the proposed approach and the methods from the literature, especially with the state of the art approach eNSP. Intensive experiments on synthetic and real datasets show that NegPSpan can extract meaningful NSPs and that it can process bigger datasets than eNSP thanks to significantly lower memory requirements and better computation times.

1 Introduction

In many application domains such as diagnosis or marketing, decision makers show a strong interest for rules that associates specific events (a context) to undesirable events to which they are correlated or that are frequently triggered in such a context. Sequential pattern mining algorithms can extract such hidden rules from execution traces or transactions. in the classical setting, sequential patterns contain only positive events, *i.e.* really observed events. However, the absence of a specific action or event can often better explain the occurrence of an undesirable situation [2]. For example in diagnosis, if some maintenance operations have not been performed, *e.g.* damaged parts have not been replaced, then a fault will likely occur in a short delay while if these operations were performed in time the fault would not occur. In marketing, if some market-place customer has not received special offers or coupons for a long time then she/he has a high probability of churning while if she/he were provided such special offers she/he should remain loyal to her/his market-place. In these two cases, mining specific events, some present and some absent, to discover under which context some undesirable situation occurs or not may provide interesting so-called *actionable* information

for determining which action should be performed to avoid the undesirable situation, *i.e.* fault in diagnosis, churn in marketing.

We aim at discovering sequential patterns that take into account the absence of some events called *negative events* [2]. Moreover, we want to take into account some aspect of the temporal dimension as well, maximal pattern span or maximal gap between the occurrences of pattern events. For example, suppose that from a sequence dataset, we want to mine a sequential pattern $\boldsymbol{p} = \langle a\ b \rangle$ with the additional *negative* constraint telling that the event c should not appear between events a and b in $\boldsymbol{p}$. The corresponding negative pattern is represented as $\boldsymbol{p} = \langle a\ \neg c\ b \rangle$, where the logical sign $\neg$ denotes an absent event or set of events. Once the general idea of introducing negative statements in a pattern has been stated, the syntax and semantics of such negative patterns should be clearly formulated since they have a strong impact both on algorithms outcome and their computational efficiency. As we will see, the few algorithms from literature do not use the same syntactical constraints and rely on very different semantics principles (see Section 3). More precisely, the efficiency of eNSP [1], the state-of-the-art algorithm for NSP mining, comes from a negation semantics that enables efficient operations on the sets of supported sequences. The two computational limits of eNSP are memory requirements and the impossibility for eNSP to handle embedding constraints such as the classical *maxgap* and *maxspan* constraints. When mining relatively long sequences (above 20 itemsets), such constraints appear semantically sound to consider short pattern occurrences where events are not too distant. In addition, such constraints can efficiently prune the occurrence search space.

This article provides two main contributions:

- we clarify the syntactic definition of negative sequential patterns and we provide different negation semantics with their properties.
- we propose NegPSpan, an algorithm inspired by algorithm PrefixSpan to extract negative sequential patterns with *maxgap* and *maxspan* constraints.

Intensive experiments compare, on synthetic and real datasets, the performance of NegPSpan and eNSP as well as the pattern sets extracted by each of them. We show that algorithm NegPSpan is more time-efficient than eNSP for mining long sequences thanks to the maxgap constraint and that its memory requirement is several orders of magnitude lower, enabling to process much larger datasets. In addition, we highlight that eNSP misses interesting patterns on real datasets due to semantic restrictions.

2 Negative Sequential Patterns

This section introduces sequential patterns and negative sequential pattern mining. First we recall some basic definitions about sequences of itemsets, and classical sequential pattern mining then we introduce some definitions of negative sequential patterns.

In the sequel, $[n] = \{1, \ldots, n\}$ denotes the set of the first n strictly positive integers. Let $(\mathcal{I}, <)$ be the set of items (alphabet) associated with a total order (*e.g.* lexicographic order). An *itemset* $A = \{a_1\ a_2\ ...\ a_m\} \subset \mathcal{I}$ is a set of ordered items. A *sequence* $\boldsymbol{s}$ is a set of sequentially ordered itemsets $\boldsymbol{s} = \langle s_1\ s_2\ ...\ s_n \rangle$. This means that for all $i, j \in [n]$, $i < j$, s_i appends before s_j in sequence $\boldsymbol{s}$. This sequence starts by s_1 and finishes by s_n. Mining sequential patterns from a dataset of sequences, denoted $\mathcal{D}$, consists in extracting the frequent subsequences (patterns) included in database sequences having a support (*i.e.,* the number of sequences in which the pattern occurs)

greater than a given threshold σ. There is a huge literature about sequential pattern mining. We will not go into details and refer the reader to a survey of the literature, such as Mooney et *al.* [10].

Negative sequential patterns (NSP) extend classical sequential patterns by enabling the specification of absent itemsets. For example, $\boldsymbol{p} = \langle a\ b\ \neg c\ e\ f\rangle$ is a negative pattern. The symbol $\neg$ before c denotes that c is a negative itemset (here reduced to an item). Semantically, $\boldsymbol{p}$ specifies that items a and b happen in a row, then items e and f occur in a row, but item c does not occur between the occurrences of bs and e.

In the field of string matching, negation is classically defined for regular expression. In this case, a pattern is an expression that can hold any kind of negated *pattern*. The same principle gives the following most generic definition of negative sequential patterns: Let $\mathcal{N}$ be the set of negative patterns. A negative pattern $\boldsymbol{p} = \langle p_1\ \ldots\ p_n\rangle \in \mathcal{N}$ is a sequence where $\forall i,\ p_i$ is a positive itemset ($p_i \subset \mathcal{I}$) or a negated pattern ($p_i = \neg\{q_i\},\ q_i \in \mathcal{N}$).

Due to its infinite recursive definition, $\mathcal{N}$ appears to be too huge to be an interesting and tractable search space for pattern mining. For instance, with $\mathcal{I} = \{a, b, c\}$, it is possible to express simple patterns like $\langle a\ \neg b\ c\rangle$ but also complex patterns like $\langle a, \neg\langle b, c\rangle\rangle$. The combinatorics for such patterns is infinite.

We now provide our definition of negative sequential patterns (NSP) which introduces some syntactic restriction compare to the most generic case. These simple restrictions are broadly used in the literature [7] and enable us to propose efficient algorithms.

Definition 1 (Negative sequential patterns (NSP))**.** A negative pattern $\boldsymbol{p} = \langle p_1\ \ldots\ p_n\rangle$ is a sequence where $\forall i,\ p_i$ is a positive itemset ($p_i = \{p_i^j\}_{j\in[m]},\ p_i^j \in \mathcal{I}$) or a negated itemset ($p_i = \neg\{q_i^j\}_{j\in[m']},\ q_i^j \in \mathcal{I}$) under the two following constraints: consecutive negative itemsets and negative itemsets at the pattern boundaries are forbidden. The *positive part*[1] of pattern $\boldsymbol{p}$, denoted $\boldsymbol{p}^+$, is the subsequence of $\boldsymbol{p}$ restricted to its positive itemsets.

According to the constraint of non consecutive negative itemsets, a negative pattern $\boldsymbol{p}$ can be denoted by $\boldsymbol{p} = \langle p_1\ \neg q_1\ p_2\ \neg q_2\ \ldots p_{n-1}\ \neg q_{n-1}\ p_n\rangle$ where $\forall i,\ p_i \subseteq I \setminus \emptyset$ and $q_i \subseteq I$. With this notation, $\boldsymbol{p}^+ = \langle p_1\ p_2\ \ldots p_{n-1}\ p_n\rangle$.

Let us illustrate our syntactic restrictions by some **contra-examples of patterns** that our approach does not extract:

- first of all, a pattern is a sequence of positives and negative itemsets. It is not possible to have patterns such as $\langle a, \neg\langle b, c\rangle\rangle$
- then, successive negated itemsets are not allowed: $\langle a\ \neg b\ \neg cd\rangle$ is not possible.
- finally, a pattern finishing or starting by a negated itemsets is also not allowed $\langle \neg b\ d\rangle$.

2.1 Semantics of Negative Sequential Patterns

The semantics of negative sequential patterns relies on *negative containment*: a sequence s supports pattern p if s contains a sub-sequence s' such that every positive itemset of p is included in some itemset of s' in the same order and for any negative itemset $\neg i$ of p, i is *not included* in any itemset

[1] Called the *maximal positive subsequence* in PNSP and Neg-GSP or the *positive element id-set* in eNSP.

occurring in the sub-sequence of s' located between the occurrence of the positive itemset preceding $\neg i$ in p and the occurrence of the positive itemset following $\neg i$ in p.

So far in the literature, the absence or non-inclusion of itemsets (represented here as a negative itemset) has been specified by loose formulations. The authors of PNSP have proposed the set symbol $\not\subseteq$ to specify non-inclusion. This symbol is misleading since it does not correspond to the associated semantics given in PNSP: an itemset I is absent from an itemset I' if the entire set I is absent from I' (as opposed to at least some item from I is absent from I') which corresponds to $I \cap I' = \emptyset$ in standard set notation, and not $I \not\subseteq I'$. We will call PNSP interpretation *total non inclusion*. It should be distinguished from *partial non inclusion* which corresponds (correctly) to the set symbol $\not\subseteq$. The symbol $\not\subseteq$ was further used by the authors of Neg-GSP and eNSP. The semantics of non inclusion is not detailed in Neg-GSP and one cannot determine if it means total or partial non inclusion.[2] eNSP does not define explicitly the semantics of non inclusion but, from the procedure used to compute the support of patterns, one can deduce that it uses total non inclusion.

Definition 2 (non inclusion)**.** We introduce two operators relating two itemsets P and I:

- partial non inclusion: $P \not\preceq I \Leftrightarrow \exists e \in P,\, e \notin I$
- total non inclusion: $P \not\sqsubseteq I \Leftrightarrow \forall e \in P, e \notin I$

Choosing one non inclusion interpretation or the other has consequences on extracted patterns as well as on pattern search. Let's illustrate this on related pattern support in the sequence dataset

$$\mathcal{D} = \left\{ \begin{array}{l} \boldsymbol{s}_1 = \langle (bc)\ f\ a \rangle \\ \boldsymbol{s}_2 = \langle (bc)\ (cf)\ a \rangle \\ \boldsymbol{s}_3 = \langle (bc)\ (df)\ a \rangle \\ \boldsymbol{s}_4 = \langle (bc)\ (ef)\ a \rangle \\ \boldsymbol{s}_5 = \langle (bc)\ (cdef)\ a \rangle \end{array} \right\}.$$

Table 1 compares the support of progressively extended patterns under the two semantics to show whether anti-monotonicity is respected or not. Let's consider pattern $\boldsymbol{p}_2$ on sequence $\boldsymbol{s}_2$. Considering that the positive part of $\boldsymbol{p}_2$ is in $\boldsymbol{s}_2$, $\boldsymbol{p}_2$ occurs in the sequence iff $(cd) \not\subseteq (cf)$. In case of total non inclusion, it is false that $(cd) \not\sqsubseteq (cf)$ because of c that occurs in (cf), and thus $\boldsymbol{p}_2$ does not occur in $\boldsymbol{s}_2$. But in case of a partial non inclusion, it is true that $(cd) \not\preceq (cf)$, because of d that does not occurs in (cf), and thus $\boldsymbol{p}_2$ occurs in $\boldsymbol{s}_2$.

Obviously, partial non inclusion satisfies anti-monotonicity while total non inclusion does not. In the sequel we will denote the general form of itemset non inclusion by the symbol $\not\subseteq$, meaning either $\not\preceq$ or $\not\sqsubseteq$.

Now, we formulate the notions of sub-sequence, non inclusion and absence by means of the concept of embedding.

Definition 3 (positive pattern embedding)**.** Let $\boldsymbol{s} = \langle s_1 \ldots s_n \rangle$ be a sequence and $\boldsymbol{p} = \langle p_1 \ldots p_m \rangle$ be a (positive) sequential pattern. $\boldsymbol{e} = (e_i)_{i \in [m]} \in [n]^m$ is an *embedding* of pattern $\boldsymbol{p}$ in sequence $\boldsymbol{s}$ iff $\forall i \in [m],\ p_i \subseteq s_{e_i}$ and $\forall i \in [m-1],\ e_i < e_{i+1}$

Definition 4 (Strict and soft embeddings of negative patterns)**.** Let $\boldsymbol{s} = \langle s_1 \ldots s_n \rangle$ be a sequence and $\boldsymbol{p} = \langle p_1 \ldots p_m \rangle$ be a negative sequential pattern.

$\boldsymbol{e} = (e_i)_{i \in [m]} \in [n]^m$ is a **soft-embedding** of pattern $\boldsymbol{p}$ in sequence $\boldsymbol{s}$ iff $\forall i \in [m]$:

[2] Actually, though not clearly stated, it seems that the negative elements of Neg-GSP patterns consist of items rather than itemsets. In this case, total and partial inclusion are equivalent.

Table 1: Lists of supported sequences in $\mathcal{D}$ by negative patterns $\boldsymbol{p}_i$, $i = 1..4$ under the total and partial non inclusion semantics. Every pattern has the shape $\langle a\ \neg q_i\ b\rangle$ where q_i are itemsets such that $q_i \subset q_{i+1}$.

	partial non inclusion $\not\preceq$	total non inclusion $\not\sqsubseteq$
$\boldsymbol{p}_1 = \langle b\neg ca\rangle$	$\{\boldsymbol{s}_1, \boldsymbol{s}_3, \boldsymbol{s}_4\}$	$\{\boldsymbol{s}_1, \boldsymbol{s}_3, \boldsymbol{s}_4\}$
$\boldsymbol{p}_2 = \langle b\neg (cd)a\rangle$	$\{\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \boldsymbol{s}_4\}$	$\{\boldsymbol{s}_1, \boldsymbol{s}_4\}$
$\boldsymbol{p}_3 = \langle b\neg (cde)a\rangle$	$\{\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \boldsymbol{s}_4\}$	$\{\boldsymbol{s}_1\}$
$\boldsymbol{p}_4 = \langle b\neg (cdeg)a\rangle$	$\{\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \boldsymbol{s}_4, \boldsymbol{s}_5\}$	$\{\boldsymbol{s}_1\}$
	monotonic	anti monotonic

- $p_i \subseteq s_{e_i}$ if p_i is positive
- $p_i \not\subseteq s_j,\ \forall j \in [e_{i-1}+1, e_{i+1}-1]$ if p_i is negative

$\boldsymbol{e} = (e_i)_{i\in[m]} \in [n]^m$ is a **strict-embedding** of pattern $\boldsymbol{p}$ in sequence $\boldsymbol{s}$ iff for all $i \in [m]$:

- $p_i \subseteq s_{e_i}$ if p_i is positive
- $p_i \not\subseteq \bigcup_{j\in[e_{i-1}+1, e_{i+1}-1]} s_j$ if p_i is negative

Proposition 1. *soft-* and *strict*-embeddings are equivalent when $\not\subseteq \stackrel{\text{def}}{:=} \not\sqsubseteq$.

Proof. see Appendix A. □

Let $\boldsymbol{p}^+ = \langle p_{k_1} \dots p_{k_l}\rangle$ be the positive part of some pattern $\boldsymbol{p}$, where l denotes the number of positive itemsets in $\boldsymbol{p}$. If $\boldsymbol{e}$ is an embedding of pattern $\boldsymbol{p}$ in some sequence $\boldsymbol{s}$, then $\boldsymbol{e}^+ = \langle e_{k_1} \dots e_{k_l}\rangle$ is an embedding of the positive sequential pattern $\boldsymbol{p}^+$ in $\boldsymbol{s}$.

The following examples illustrate the impact of itemset non-inclusion operator and of embedding type.

Example 1 (Itemset absence semantics)**.** Let $\boldsymbol{p} = \langle a\ \neg(bc)\ d\rangle$ be a pattern and four sequences:

Sequence	$\not\sqsubseteq$	$\not\preceq$ / strict-embedding	$\not\preceq$ / soft-embedding
$\boldsymbol{s_1} = \langle a\ c\ b\ e\ d\rangle$	✓		
$\boldsymbol{s_2} = \langle a\ (bc)\ e\ d\rangle$			
$\boldsymbol{s_3} = \langle a\ b\ e\ d\rangle$	✓	✓	
$\boldsymbol{s_4} = \langle a\ e\ d\rangle$	✓	✓	✓

One can notice that each sequence contains a unique occurrence of $\langle a\ d\rangle$, the positive part of pattern $\boldsymbol{p}$. Using soft-embeddings and total non-inclusion ($\not\subseteq \stackrel{\text{def}}{:=} \not\sqsubseteq$), $\boldsymbol{p}$ occurs in $\boldsymbol{s_1}$, $\boldsymbol{s_3}$ and $\boldsymbol{s_4}$ but not in $\boldsymbol{s_2}$. Using the strict-embedding semantics and partial non-inclusion, $\boldsymbol{p}$ occurs in sequence $\boldsymbol{s_3}$ and $\boldsymbol{s_4}$ considering that items b and c occur between occurrences of a and d in sequences 1 and 2. With partial non inclusion ($\not\subseteq \stackrel{\text{def}}{:=} \not\preceq$) and either type of embeddings, the absence of an itemset is satisfied if any of its item is absent. As a consequence, $\boldsymbol{p}$ occurs only in sequence $\boldsymbol{s_4}$.

Another point that determines the semantics of negative containment concerns the multiple occurrences of some pattern in a sequence: should every or only one occurrence of the pattern positive part in the sequence satisfy the non inclusion constraints? This point is not discussed in previous propositions for negative sequential pattern mining. Actually, PNSP and Neg-GSP require a weak absence (at least one occurrence should satisfy the non inclusion constraints) while eNSP requires a strong absence (every occurrence should satisfy non inclusion constraints).

Definition 5 (Negative pattern occurrence)**.** Let $\boldsymbol{s}$ be a sequence, $\boldsymbol{p}$ be a negative sequential pattern, and $\boldsymbol{p}^+$ the positive part of $\boldsymbol{p}$.

- Pattern $\boldsymbol{p}$ *softly-occurs* in sequence $\boldsymbol{s}$, denoted $\boldsymbol{p} \preceq \boldsymbol{s}$, iff there exists at least one (strict/soft)-embedding of $\boldsymbol{p}$ in $\boldsymbol{s}$.
- Pattern $\boldsymbol{p}$ *strictly-occurs* in sequence $\boldsymbol{s}$, denoted $\boldsymbol{p} \sqsubseteq \boldsymbol{s}$, iff for any embedding $\boldsymbol{e'}$ of $\boldsymbol{p}^+$ in $\boldsymbol{s}$ there exists an embedding $\boldsymbol{e}$ of $\boldsymbol{p}$ in $\boldsymbol{s}$ such that $\boldsymbol{e'} = \boldsymbol{e}^+$.

Definition 5 allows for formulating two notions of absence semantics for negative sequential patterns depending on the occurrences of the positive part:

- *strict occurrence*: a negative pattern $\boldsymbol{p}$ occurs in a sequence $\boldsymbol{s}$ iff there exists at least one occurrence of the positive part of pattern $\boldsymbol{p}$ in sequence $\boldsymbol{s}$ and **every** such occurrence satisfies the negative constraints,
- *soft occurrence*: a negative pattern $\boldsymbol{p}$ occurs in a sequence $\boldsymbol{s}$ iff there exists at least one occurrence of the positive part of pattern $\boldsymbol{p}$ in sequence $\boldsymbol{s}$ and **one** of these occurrences satisfies the negative constraints.

Example 2 (Strict vs soft occurrence semantics)**.** Let $\boldsymbol{p} = \langle a\ b\ \neg c\ d\ \rangle$ be a pattern and $\boldsymbol{s_1} = \langle a\ b\ e\ d \rangle$ and $\boldsymbol{s_2} = \langle a\ b\ c\ a\ d\ e\ b\ d \rangle$ be two sequences. The positive part of $\boldsymbol{p}$ is $\langle a\ b\ d \rangle$. It occurs once in $\boldsymbol{s_1}$ so there is no difference for occurrences under the two semantics. But, it occurs thrice in $\boldsymbol{s_2}$ with embeddings $(1, 2, 5)$, $(1, 2, 8)$ and $(4, 7, 8)$. The two first occurrences do not satisfy the negative constraint $(\neg c)$ while the second occurrence does. Under the soft occurrence semantics, pattern p occurs in sequence $\boldsymbol{s_2}$ whereas under the strict occurrence semantics it does not.

We also introduce **constrained negative sequential patterns**. We consider the two most common anti-monotonic constraints on sequential patterns: *maxgap* ($\theta \in \mathbb{N}$) and *maxspan* ($\tau \in \mathbb{N}$) constraints. These constraints impact NSP embeddings. An embedding $\boldsymbol{e}$ of a pattern $\boldsymbol{p}$ in some sequence $\boldsymbol{s}$ satisfies the *maxgap* (resp. *maxspan*) constraint iff $\boldsymbol{e}^+ = \{e_i, \ldots, e_n\}$, the embedding of the positive part of $\boldsymbol{p}$ satisfies the constraint, *i.e.* $\forall i \in [n-1],\ e_{i+1} - e_i \leq \theta$ (resp. $e_n - e_1 \leq \tau$).

The definitions of pattern support, frequent pattern and pattern mining task derives naturally from the notion of occurrence of a negative sequential pattern, no matter the choices for embedding (soft or strict), non inclusion (partial or total) and occurrence (soft or strict). However, these choices concerning the semantics of NSPs impact directly the number of frequent patterns (under the same minimal threshold) and further the computation time. The stronger the negative constraints, the lesser the number of sequences that hold some pattern, and the lesser the number of frequent patterns.

Finally, we introduce a partial order on NSPs that is the foundation of our efficient NSP mining algorithm.

Definition 6 (NSP partial order)**.** Let $\boldsymbol{p} = \langle p_1 \neg q_1\, p_2 \neg q_2 \ldots p_{k-1} \neg q_{k-1}\, p_k \rangle$ and $\boldsymbol{p'} = \langle p'_1 \neg q'_1\, p'_2 \neg q'_2 \ldots p'_{k'-1} \neg q'_{k'-1}\, p'_{k'} \rangle$ be two NSPs s.t. $\forall i \in [k],\ p_i \neq \emptyset$ and $\forall i \in [k'],\ p'_i \neq \emptyset$. By definition, $\boldsymbol{p} \lhd \boldsymbol{p'}$ iff $k \leq k'$ and:

1. $\forall i \in [k-1],\ p_i \subseteq p'_i$ and $q_i \subseteq q'_i$
2. $p_k \subseteq p'_k$
3. $k' \neq k \implies p_k \neq p'_k$ (non-reflexive)

Intuitively, $\boldsymbol{p} \lhd \boldsymbol{p'}$ if $\boldsymbol{p}$ is shorter than $\boldsymbol{p'}$ and the positive and negative itemsets of $\boldsymbol{p}$ are pairwise included into the itemsets of $\boldsymbol{p'}$, but, in case of extension by additional itemsets. The classical pattern inclusion fails to be anti-monotonic [19], since the change of scope of negative itemsets. We illustrate what's happening on two examples. Let first consider the case of an ending negated itemset illustrated by Zheng et *al.* . with patterns $\boldsymbol{p'} = \langle b \neg c\ a \rangle$ and $\boldsymbol{p} = \langle b \neg c \rangle$: removing the a make the positive pattern less constraint (more frequent), but is extend the scope of the negative constraint. Negation are more constraint and the anti-monotonicity is lost. This specific case does not impact our framework as our definition of NSP (see Definition 1) does not allow ending negated itemsets. But let us now consider the patterns $\boldsymbol{p'} = \langle b \neg c\ da \rangle$ and $\boldsymbol{p} = \langle b \neg ca \rangle$, and the sequences $\boldsymbol{s} = \langle b\ e\ d\ c\ a \rangle$. $\boldsymbol{p'}$ occurs in $\boldsymbol{s}$ but not $\boldsymbol{p}$ has the scope of the negated itemset $\neg c$ changed it was restricted to the interval between b and d occurrence for $\boldsymbol{p'}$, but between b and a for $\boldsymbol{p}$.

What is important in our partial order $\lhd$, is that the embedding of the positive pattern yields an embedding for $\boldsymbol{p}$ that imposes the negative constraints on the exact same scopes than negative constraints of $\boldsymbol{p'}$. Thanks to the anti-monotonicity of $\not\sqsubseteq$, additional itemsets in negative patterns leads to over constraints the sequence. These remarks give some intuition behind the following anti-monotonicity property (Proposition 2). The formal proof of the proposition can be found in Appendix A.

Proposition 2 (Anti-monotonicity of NSP support)**.** The support of NSP is anti-monotonic with respect to $\lhd$ when $\not\preceq \stackrel{\text{def}}{=} \not\sqsubseteq$ and soft-occurrences ($\preceq$) are considered.

We can notice that while the strict occurrence semantic ($\sqsubseteq$) is used, $\lhd$ lost the anti-monotonicity. Considering $\boldsymbol{p'} = \langle a\ (bc)\ \neg c\ d \rangle$, $\boldsymbol{p} = \langle a\ b\ \neg c\ d \rangle$ and $\boldsymbol{s} = \langle a\ \mathbf{b}\ (bc)\ e\ d \rangle$, then it is true that $\boldsymbol{p'} \sqsubseteq \boldsymbol{s}$, but not that $\boldsymbol{p} \sqsubseteq \boldsymbol{s}$. In the second case, there are two possible embeddings and the second one (which does not derived from the embedding of $\boldsymbol{p'}$) does not satisfy the negative constraint.

A second example illustrates another case that is encountered when a postfixed sequence restricts the set of valid embeddings: $\boldsymbol{p'} = \langle a \neg b\ d\ \mathbf{c} \rangle$, $\boldsymbol{p} = \langle a \neg b\ d \rangle$ and $\boldsymbol{s} = \langle a\ e\ d\ c\ b\ d \rangle$. Again, $\boldsymbol{p'}$ occurs only once while $\boldsymbol{p}$ occur twice and one of its embeddings does not satisfy the negated itemset. This example shows that a simple postfix extension of NSP leads to loose the monotonicity property while the strict occurrence semantic is considered.

3 Related Work

Kamepalli et *al.* provide a survey of the approaches proposed for mining negative patterns [7]. The three most significant algorithms appear to be PNSP, Neg-GSP and eNSP. We briefly review each of them in the following paragraphs.

PNSP (Positive and Negative Sequential Patterns mining) [6] is the first algorithm proposed for mining full negative sequential patterns where negative itemsets are not only located at the

end of the pattern. PNSP extends algorithm GSP [15] to cope with mining negative sequential patterns. PNSP consists of three steps: i) mine frequent positive sequential patterns, by using algorithm GSP, ii) preselect negative sequential itemsets — for PNSP, negative itemsets must not be too infrequent (should have a support less than a threshold *miss_freq*) — iii) generate candidate negative sequences levelwise and scan the sequence dataset again to compute the support of these candidates and prune the search when the candidate is infrequent. This algorithm is incomplete: the second parameter reduces the set of potential negative itemsets. Moreover, the pruning strategy of PNSP is not correct [19] and PNSP misses potentially frequent negative patterns.

Zheng et *al.* [19] also proposed a negative version of algorithm GSP, called Neg-GSP, to extract negative sequential patterns. They showed that traditional Apriori-based negative pattern mining algorithms relying on support anti-monotonicity have two main problems. The first one is that the Apriori principle does not apply to negative sequential patterns. They gave an example of sequence that is frequent even if one of its sub-sequence is not frequent. The second problem has to do with the efficiency and the effectiveness of finding frequent patterns due to a vast candidate space. Their solution was to prune the search space using the support anti-monotonicity over positive parts. This pruning strategy is correct but incomplete and it is not really efficient considering the huge number of remaining candidates whose support has to be evaluated. To improve the efficiency of their approach, the authors proposed an incomplete heuristic search based on Genetic Algorithm to find negative sequential patterns [20]. We will see in Section 2.1 that anti-monotonicity can be defined considering a partial order relation based on common prefixes that a enable to design a complete, correct and efficient algorithm.

eNSP (efficient NSP) has been recently proposed by Cao et *al.* [1]. It identifies NSPs by computing only frequent positive sequential patterns and deducing negative sequential patterns from positive patterns. Precisely, Cao et *al.* showed that the support of some negative pattern can be computed by arithmetic operations on the support of its positive sub-patterns, thus avoiding additional sequence database scans to compute the support of negative patterns. However, this necessitates to store all the (positive) sequential patterns with their set of covered sequences (tid-lists) which may be impossible in case of big dense datasets and low minimal support thresholds. This approach makes the algorithm more efficient but it hides some restrictive constraints on the extracted patterns. First, a frequent negative pattern whose so-called positive partner (the pattern where all negative events have been switched to positive) is not frequent will not be extracted. Second, every occurrence of a negative pattern in a sequence should satisfy absence constraints. We call this *strong absence semantics* (see Section 2.1). These features lead eNSP to extract less patterns than previous approaches. In some practical applications, eNSP may miss potentially interesting negative patterns from the dataset.

The first constraint has been partly tackled by Dong et *al.* with algorithm eNSPFI, an extension of eNSP which mines NSPs from frequent and some infrequent positive sequential patterns from the negative border [5]. E-msNSP [17] is another extension of eNSP which uses multiple minimum supports: an NSP is frequent if its support is greater than a local minimal support threshold computed from the content of the pattern and not a global threshold as in classical approaches. A threshold is associated with each item, and the minimal support of a pattern is defined from the most constrained item it contains. Such kind of adaptive support prevents from extracting some useless patterns still keeping the pattern support anti-monotonic. The same authors also proposed high utility negative sequential patterns based on the same principles [16] and applied on smart city data [18]. An alternative approach has been proposed by Lin consisting in mining high-utility itemsets with negative unit profits [8] but is not applied on sequential patterns. It is worth noting that

Table 2: Comparison of negative pattern mining proposals. Optional constraints are specified in Italic.

	PNSP [6]	**NegGSP** [19]	**eNSP** [1]	NegPSpan
negative elements	itemsets	items*?*	itemsets	itemsets
itemsets	$\not\subset$?	$\not\subset$	$\not\sqsubseteq$	$\not\sqsubseteq$
embeddings	strict	strict?	strict	strict/*soft*
occurrences	soft	soft	strict	soft
constraints on negative itemsets	not too infrequent ($supp \leqslant less\_freq$)	frequent items	positive partner is frequent	frequent items, *bounded size*
global constraints on patterns			positive part is frequent (second greater threshold)	*maxspan*, *maxgap*

this algorithm relies basically on the same principle as eNSP and so, present the same drawbacks, heavy memory requirements, strong absence semantics for negation. F-NSP+ [4] extends the eNSP algorithm to use bitmap representations of itemsets. Using bitmap representations enable to speed up the eNSP algorithm, thanks to very efficient set operation on bitmaps. The F-NSP algorithm has a poor memory usage, while F-NSP+, which adapts the bitmap size to the dataset, requires slightly less memory.

SAPNSP [9] tackles the problem of large amount of patterns by selecting frequent negative and positive patterns that are actionable. Patterns are actionable while they conform to *special rules*.

NegI-NSP [14] proposes additional syntactic constraints on negative itemsets and uses the same strategy as e-NSP.

To conclude this section on formal aspects of negative pattern mining, we provide in Table 2 a comparison of several negative sequential pattern mining approaches wrt several features investigated in this section. It is also important to precise that not any semantics is "more correct" than another one. Its relevancy depends on the information the data scientists want to capture in its datasets, and the nature of the data at hand. In this work, one of our objective is to provide a sound and insightful framework about negative patterns to enable users to choose the tool to use and to make this choice according to the semantic of the negation they want to use. Execution time is obviously an important choice criteria but it must overtake by semantic choice to first provide interesting, intuitive and sound results.

4 Algorithm NegPSpan

In this section, we introduce algorithm NegPSpan for mining NSPs from a sequence database under *maxgap* and *maxspan* constraints and under a weak absence semantics with $\not\subseteq \stackrel{\text{def}}{:=} \not\sqsubseteq$ for itemset inclusion. As stated in proposition 1, no matter the embedding strategy, they are equivalent under strict itemset inclusion. Considering occurrences, NegPSpan uses the soft-occurrence semantic: at least one occurrence of the negative pattern is sufficient to consider that it is supported by the

sequence.

For computational reasons, we make an additional assumption on the admissible itemsets as negative itemsets. The negative itemsets are restricted to one element of some language $\mathcal{L}^-$ in order to cut the combinatorics of negative itemsets. In the algorithm NegPSpan presented below, $\mathcal{L}^- = \{I = \{i_1, \ldots, i_n\} | \forall k,\ supp(i_k) \geq \sigma\}$ denotes the set of itemsets that can be built from frequent items. But this set could also be user defined when the user is interested in some specific sets of non-occurring events. For instance, $\mathcal{L}^-$ could be the set of frequent itemsets, which would be more restrictive than the set of itemsets made of frequent itemsets.

4.1 Main Algorithm

NegPSpan is based on algorithm PrefixSpan [12] which implements a depth first search and uses the principle of database projection to reduce the number of sequence scans. NegPSpan adapts the pseudo-projection principle of PrefixSpan which uses a projection pointer to avoid copying the data. For NegPSpan, a projection pointer of some pattern $\boldsymbol{p}$ is a triple $\langle sid, ppred, pos \rangle$ where *sid* is a sequence identifier in the database, *pos* is the position in sequence *sid* that matches the last itemset of the pattern (necessarily positive) and *ppred* is the position of the previous positive pattern.

Algorithm 1 details the main recursive function of NegPSpan for extending a current pattern $\boldsymbol{p}$. The principle of this function is similar to PrefixSpan. Every pattern $\boldsymbol{p}$ is associated with a pseudo-projected database represented by both the original set of sequences $\mathcal{S}$ and a set of projection pointers *occs*. First, the function evaluates the size of *occs* to determine whether pattern p is frequent or not. If so, it is outputted, otherwise, the recursion is stopped because no larger patterns are possible (anti-monotonicity property).

Then, the function tries three types of pattern extensions of pattern $\boldsymbol{p}$ into a pattern $\boldsymbol{p'}$:

- the positive sequence composition ($\rightsquigarrow_c$) consists in adding one item to the last itemset of $\boldsymbol{p}$ (following the notations of Definition 6, the extension corresponds to the case of $\boldsymbol{p'}$ is the extension of $\boldsymbol{p}$ where $k' = k$, $\forall i \in [k-1]$, $q_i = q'_i$ and $|p'_k| = |p_k| + 1$),
- the positive sequence extension ($\rightsquigarrow_s$) consists in adding a new positive singleton itemset at the end of $\boldsymbol{p}$ ($k' = k+1$, $\forall i \in [k-1]$, $q_i = q'_i$ and $|p'_{k'}| = 1$),
- the negative sequence extension ($\rightsquigarrow_n$) consists in inserting a negative itemset between the positive penultimate itemset of $\boldsymbol{p}$ and the last positive itemset of $\boldsymbol{p}$ ($k' = k$, $\forall i \in [k-2]$, $q_i = q'_i$, $|q'_{k-1}| = |q_{k-1}| + 1$ and $p'_k = p_k$). In addition, NSP are negatively extended iff $|p_k| = 1$ to prevent from redundant pattern generation (see section 4.3).

The negative pattern extension is specific to our algorithm and is detailed in the next section. The first two extensions are identical to PrefixSpan pattern extensions, including their gap constraints management, *i.e. maxgap* and *maxspan* constraints between positive patterns.

Proposition 3. The proposed algorithm is correct and complete.

Intuitively, the algorithm is complete considering that the three extensions enables to generate any NSP. For instance, pattern $\langle a\ \neg e\ b\ (ce)\ \neg(bd)\ a \rangle$ would be evaluated after evaluating the following patterns: $\langle a \rangle \rightsquigarrow_s \langle a\ b \rangle \rightsquigarrow_n \langle a\ \neg e\ b \rangle \rightsquigarrow_s \langle a\ \neg e\ b\ c \rangle \rightsquigarrow_c \langle a\ \neg e\ b\ (ce) \rangle \rightsquigarrow_s \langle a\ \neg e\ b\ (ce)\ a \rangle \rightsquigarrow_n \langle a\ \neg e\ b\ (ce)\ \neg b\ a \rangle \rightsquigarrow_n \langle a\ \neg e\ b\ (ce)\ \neg(bd)\ a \rangle$. Secondly, according to proposition 2, the pruning strategy is correct.

4.2 Extension of Patterns with Negated Itemsets

Algorithm 2 extends the current pattern $\boldsymbol{p}$ with negative items. It generates new candidates by inserting an item $it \in \mathcal{I}^f$, the set of frequent items. Let $\boldsymbol{p}[-2]$ and $\boldsymbol{p}[-1]$ denote respectively the penultimate itemset and the last itemset of $\boldsymbol{p}$. If $\boldsymbol{p}[-2]$ is positive, then a new negated itemset is inserted between $\boldsymbol{p}[-2]$ and $\boldsymbol{p}[-1]$. Otherwise, if $\boldsymbol{p}[-2]$ is negative, item it is added to $\boldsymbol{p}[-2]$. To prevent redondant enumeration of negative itemsets, only items it (lexicographically) greater than the last item of $\boldsymbol{p}[-2]$ can be added.

Then, lines 10 to 20, evaluate the candidate by computing the pseudo-projection of the current database. According to the selected semantics associated with $\not\sqsubseteq$, *i.e.* total non inclusion (see Definition 4), it is sufficient to check the absence of it in the subsequence included between the occurrences of positive itemsets surounding it. To achieve this, the algorithm checks the sequence positions in the interval $[occ.ppred+1, occ.pos-1]$. If it does not occur in itemsets from this interval, then the extended pattern occurs in the sequence $occ.sid$. Otherwise, to ensure the completeness of the algorithm, another occurrence of the pattern has to be searched in the sequence (*cf.* `Match` function that takes into account gap constraints).

For example, the first occurrence of pattern $\boldsymbol{p} = \langle abc \rangle$ in sequence $\langle abecabc \rangle$ is $occ_p = \langle sid, 2, 4 \rangle$. Let's now consider $\boldsymbol{p'} = \langle ab\neg ec \rangle$, a negative extension of p. The extension of the projection-pointer occ_p does not satisfy the absence of e. So a new occurrence of p has to be searched for. $\langle sid, 6, 7 \rangle$, the next occurrence of $\boldsymbol{p}$, satisfies the negative constraint. Then, NegPSpan is called recursively for extending the new current pattern $\langle ab\neg ec \rangle$.

We can note that the gap constraints τ and θ does not explicitly appear in this algorithm (except while a complete matching is required), but it impact indirectly the algorithm by narrowing the possible interval of line 13.

4.2.1 Extracting NSP without surrounding negations

An option restricts negated item to be not surrounded by itemsets containing this item. This alternative is motivated by the objective to simplify pattern understanding. A pattern $\langle a\neg bbc \rangle$ may be interpreted as "there is exactly one occurrence of b between a and c". But, this may also lead to redundant patterns: $\langle ab\neg bc \rangle$ matches exactly the same sequences than $\langle a\neg bbc \rangle$ (see section 4.3). This second restriction can be disabled in our algorithm implementation. If so and for sake of simplicity, we preferred to yield the pattern $\langle ab\neg bc \rangle$.

The set of such NSP can be extracted using the same algorithm, simply changing the candidate generation in Algorithm 2, line 2 by $it \in \mathcal{I}^f \setminus (\boldsymbol{p}[-1] \cup \boldsymbol{p}[-2])$. Items to add to a negative itemset are among frequent items except surrounding items.

4.2.2 Extracting NSP with partial non inclusion ($\not\subseteq \stackrel{\text{def}}{=} \not\preceq$)

Algorithm 3 present the variant of the negative extension algorithm while the partial non-inclusion is used ($\not\subseteq \stackrel{\text{def}}{=} \not\preceq$). The backbone of the algorithm is similar: a candidate pattern with a negated itemset at the penultimate position is generated and it assesses whether this candidate is frequent or not. It is done by checking the absence of the itemset is in the itemsets of the sequence at positions defined by the occurrence. The test of line 7 assesses that it is false that $is \not\preceq \boldsymbol{s}_{occ.sid}[sp]$: is is not partial non-included in one the itemset of the sequence iff is is a subset of it.

On the contrary to the previous approach, candidate patterns are generated based on $\mathcal{L}^-$ the list of itemsets. It is not possible to build itemsets from the list of items because, using this non-

Algorithm 1: NegPSpan: recursive function for negative sequential pattern extraction

input: $\mathcal{S}$: set of sequences, $\boldsymbol{p}$: current pattern, σ: minimum support threshold, *occs*: list of occurrences, $\mathcal{I}^f$: set of frequent items, θ: maxgap, τ: maxspan

```
Function NegPSpan(S, σ, p, occs, I^f, θ, τ):
    //Support evaluation of pattern p
    if |occs| ≥ σ then
        OutputPattern(p, occs);
    else
        return;
    //Positive itemset composition
    PositiveComposition(S, σ, p, occs, I^f, θ, τ);
    //Positive sequential extension
    PositiveSequence(S, σ, p, occs, I^f, θ, τ);
    if |p| ≥ 2 and |p_{|p|}| = 1 then
        //Negative sequential extension
        NegativeExtension(S, σ, p, occs, I^f, θ, τ);
```

Algorithm 2: NegPSpan: negative extensions

input: $\mathcal{S}$: set of sequences, $\boldsymbol{p}$: current pattern, σ: minimum support threshold, *occs*: list of occurrences, $\mathcal{I}^f$: set of frequent items, θ: maxgap, τ: maxspan

```
Function NegativeExtension(S, σ, p, occs, I^f, θ, τ):
    for it ∈ I^f do
        if p[−2] is pos then
            //Insert the negative item at the penultimate position
            p.insert(¬it);
        else
            if it > p[−2].back() then
                //Insert an item to the penultimate (negative) itemset
                p[−2].insert(¬it);
            else
                continue;
        newoccs ← ∅;
        for occ ∈ occs do
            found ← false;
            for sp = [occ.pred + 1, occ.pos − 1] do
                if it ∈ s_{occ.sid}[sp] then
                    found ← true;
                    break;
            if !found then
                newoccs ← newoccs ∪ {occ};
            else
                //Look for an alternative occurrence
                newoccs ← newoccs∪ Match(s_sid, p, θ, τ);
        NegPSpan(D, σ, p, newoccs, I^f);
        p[−2].pop();
```

Algorithm 3: NegPSpan: negative extensions with partial non-inclusion (alternative to Algorithm 2)

input: $\mathcal{S}$: set of sequences, $\boldsymbol{p}$: current pattern, σ: minimum support threshold, *occs*: list of occurrences, $\mathcal{I}^f$: set of frequent items, θ: maxgap, τ: maxspan

```
1  Function NegativeExtension(S, σ, p, occs, I^f, θ, τ):
2      for is ∈ L^− do
           //Insert the negative itemset at the penultimate position
3          p.insert(¬is);
4          newoccs ← ∅;
5          for occ ∈ occs do
6              found ← false;
7              for sp = [occ.pred + 1, occ.pos − 1] do
8                  if is ⊆ s_{occ.sid}[sp] then
9                      found ← true;
10                     break;
11             if !found then
12                 newoccs ← newoccs ∪ {occ};
13             else
                   //Look for an alternative occurrence
14                 newoccs ← newoccs∪ Match(s_sid, p, θ, τ);
15         NegPSpan(D, σ, p, newoccs, I^f);
16         p[−2] = ∅;
```

inclusion semantic, the support is monotonic (and not anti-monotonic). The combinatorics of this variant is thus significantly higher in practice because all element of $\mathcal{L}^-$ would be evaluated.

4.3 Redundancy avoidance

The NegPSpanalgorithm is syntactically non-redundant but can in practice generates patterns that are semantically redundant.

The semantic redundancy appears for pairs of patterns like $\langle a\ \neg b\ b\ c\rangle$ and $\langle a\ b\ \neg b\ c\rangle$: there are syntactically different but match the exact same set of sequences. Semantically, such pattern could be interpreted as "*there is not much than one occurrence of b between a and c*". For such patterns, it is possible to avoid generating both efficiently. Our solution is to not generate candidate patterns with negative items that are in the last itemset. Thus, only $\langle a\ b\ \neg b\ c\rangle$ would be generated. In Algorithm 2 line 2, the list of frequent items $\mathcal{I}^f$ is then replaced by $\mathcal{I}^f \setminus \boldsymbol{p}[1]$. But, this modification makes loose the completeness of the algorithm. In fact, the pattern $\langle a\ \neg b\ b\rangle$ is not generated neither its semantically equivalent pattern $\langle a\ b\ \neg b\rangle$ because of the syntactic constraint on NSP that can not end with a negative itemset. In practice, we do not manage this kind of redundancy or prefer the sound and correct option of not surrounding negative itemsets (see section 4.2.1).

A syntactic redundancy is introduced by adding the extension by negative items. For instance, the pattern $\langle a\ \neg b\ (cd)\rangle$ may be reached by two distinct paths $p_1 : \langle a\ c\rangle \rightsquigarrow_c \langle a\ (cd)\rangle \rightsquigarrow_n \langle a\ \neg b\ (cd)\rangle$ or $p_2 : \langle a\ c\rangle \rightsquigarrow_n \langle a\ \neg b\ c\rangle \rightsquigarrow_c \langle a\ \neg b\ (cd)\rangle$. To solve this problem, the algorithm first specifies the negative itemsets as a composition of negative items and then to compose the last itemset with new items. This discard the path p_1. In Algorithm 1, line 8 enables negative extension only if the last (positive) itemset is of size 1.

4.4 Execution Example

This section illustrates the execution of the algorithm on a small example. Let us consider the dataset of sequences of Table 3 and the minimal support threshold $\sigma = 2$. In this example, we consider the following negative patterns semantic: total non-inclusion and strong absence. No gap constraints are considered ($\theta = \infty$ and $\tau = \infty$). Then, we have $\mathcal{L} = \{a, b, c, d, e\}$. The f event occurs only once and are thus not frequent according to σ value.

Table 3: Dataset of sequences used in the execution example.

SID	Sequence
$\boldsymbol{s}_1$	$\langle a\ c\ b\ e\ d\rangle$
$\boldsymbol{s}_2$	$\langle a\ (bc)\ e\rangle$
$\boldsymbol{s}_3$	$\langle a\ b\ e\ d\rangle$
$\boldsymbol{s}_4$	$\langle a\ e\ d\ f\rangle$

Figure 1 illustrates the execution of NegPSpan algorithm on the dataset of Table 3 starting from pattern $\langle a\rangle$. The tree illustrates successive patterns explored by the depth-first search strategy. Each node detailed both the pattern and the corresponding projected database. For sake of space, the tree is simplified and some nodes are missing.

For patterns larger than two, projected sequences have two colors corresponding, in green, of the part of the sequence that can be used to make positive extensions and, in red, of the part of the sequence that is used to assess absence of items for negative extension. Two markers locate positions of the projection pointer. The second pointer is the same as the one computed by PrefixSpan.

Let us consider projected sequences of pattern $\langle ae\rangle$. In the first sequence, d is green as the part of the sequence ending the sequence after position of e. cb are in read because this events are inbetween occurrence of a and e. This pattern can be extended in two ways:

- with negative items among $\mathcal{L} \setminus \{a, e\}$ (a and e are removed if the restriction on second restriction is activated),
- with positive items among items that are frequent in the green parts.

Considering extension of pattern $\langle a\ e\rangle$ with a negative item, *e.g.* $\neg c$, each sequence whose red part contains the item is discarded, the others remains identical. Extension by $\neg c$ leads to pattern $\langle a\neg c\ e\rangle$ only for sequences s_3 and s_4.

The extension of pattern $\langle a\ e\rangle$ by a positive item follows the same strategy as PrefixSpan. In this case, the algorithm only explore extension by d item and projected pointers are updated to reduce further scanning.

Adding a new negative item while the penultimate item is negative, append it in the negative itemset. In case of pattern $\langle a\neg c\ e\rangle$, d is the only candidate because e is one of the surrounding events and b is above c is the lexicograpic order. With the total non-inclusion, we again simply have to discard sequences that contain the item d within their red part.
We can see in the case of the $\langle a\neg d\ e\ d\rangle$ extension that all combinatorics of itemsets may quickly satisfy all negation constraints. This suggests first to carefully select the appropriate $\mathcal{L}$ and second to use a maximum size for negative itemsets to avoid pattern explosion.

Finally, we also notice that extensions with negative items are not terminal recusive steps. Once negative items have been inserted, new positive items can be append to the pattern. We encounter this case with pattern $\langle a \neg d\ e \rangle$ which is extended by pattern d.

5 Experiments

This section presents experiments on synthetic and real data. Experiments on synthetic data aims at exploring and comparing NegPSpan and eNSP for negative sequential pattern mining. The other experiments were conducted on medical care pathways and illustrates results for negative patterns. NegPSpan and eNSP have been implemented in C++. We pay attention on the most significant results. More detailed results can be found in a compagnon website.[3]

5.1 Benchmark

This section presents experiments on synthetically generated data. The principle of our sequence generator is the following: generate random negative patterns and hide or not some of their occurrences inside randomly generated sequences. The main parameters are the total number of sequences (n, default value is $n = 500$), the mean length of sequences ($l = 20$), the number of different items ($d = 20$), the total number of patterns to hide (3), their mean length (4) and the minimum occurrence frequency of patterns in the dataset (10%).

Generated sequences are sequences of items (not itemsets). For such kind of sequences, patterns extracted by eNSP hold only items because positive partners have to be frequent. For a fair evaluation and preventing NegPSpan from generating more patterns, we restricted $\mathcal{L}^-$ to the set of frequent items. For both approaches, we limit the pattern length to 5 items.

Figure 2 illustrates the computation time and number of patterns extracted by eNSP and NegPSpan on sequences of length 20 and 30, under three minimal thresholds ($\sigma = 10\%$, 15% and 20%) and with different values for the maxgap constraint ($\tau = 4$, 7, 10 and ∞). For eNSP, the minimal support of positive partners, denoted ς, is set to 70% of the minimal threshold σ. Each boxplot has been obtained with a 20 different sequence datasets. Each run has a timeout of 5 minutes.

The main conclusion from Figure 2 is that NegPSpan is more efficient than eNSP when maxgap constraints are used. As expected, eNSP is more efficient than NegPSpan without any maxgap constraint. This is mainly due to the number of extracted patterns. NegPSpan extracts significantly more patterns than eNSP because of different choices for the semantics of NSPs. First, eNSP uses a stronger negation semantics. Without maxgap constraints, the set of patterns extracted by NegPSpan is a superset of those extracted by eNSP (see proof in Appendix B).

An interesting result is that, for reasonably long sequences (20 or 30), even a weak maxgap constraint ($\tau = 10$) significantly reduces the number of patterns and makes NegPSpan more efficient. $\tau = 10$ is said to be a *weak* constraint because it does not cut early the search of a next occurring item compare to the length of the sequence (20 or 30). This is of particular interest because the maxgap is a quite natural constraint when mining long sequences. It prevents from taking into account long distance correlations that are more likely irrelevant. Another interesting question raised by this results is the real meaning of extracted patterns by eNSP. In fact, under low frequency thresholds, it extracts numerous patterns that are not frequent when weak maxgap

[3]Code, data generator and synthetic benchmark datasets can be downloaded here: `http://people.irisa.fr/Thomas.Guyet/negativepatterns/`.

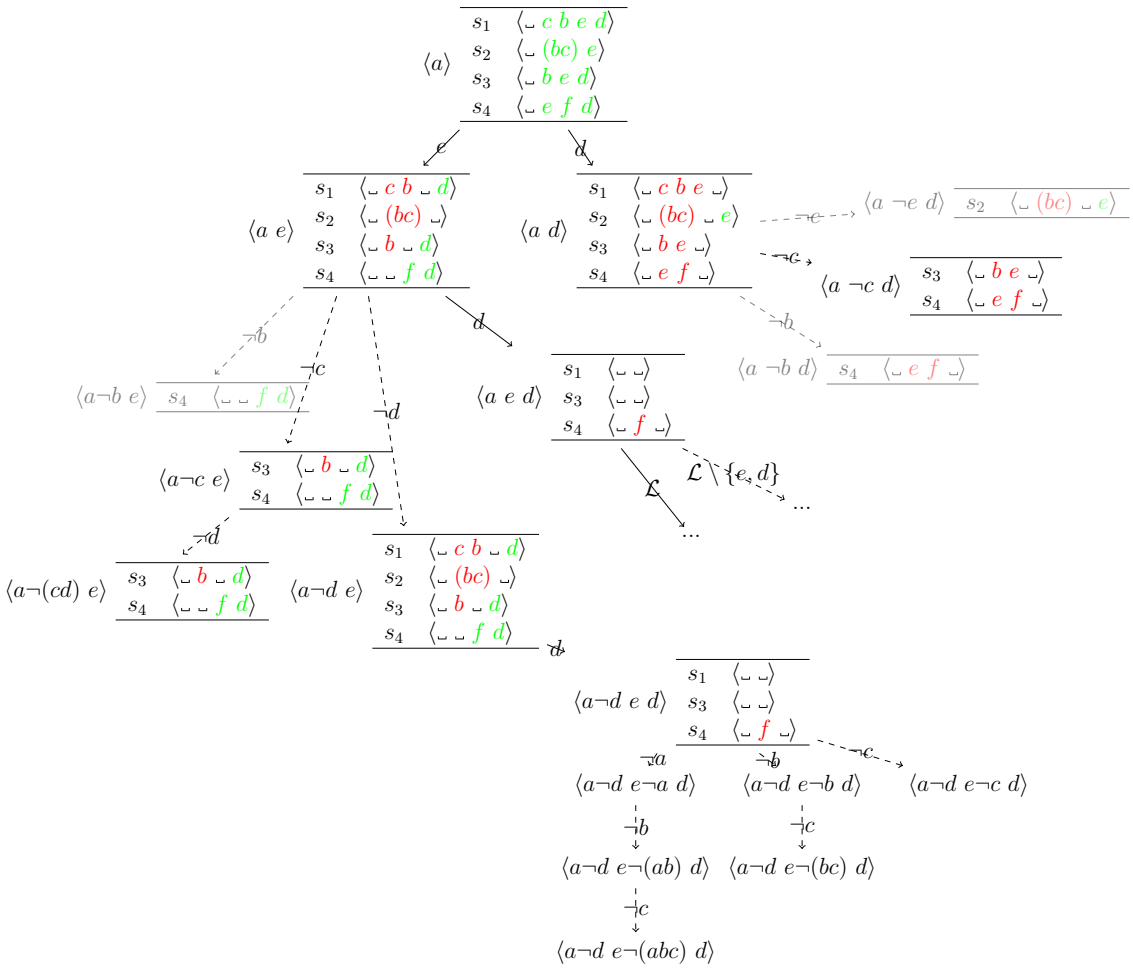

Figure 1: Example of the tree search of the NegPSpan algorithm of dataset of Table 3. Each node tree represents the pattern on the left and the projected database on the right. Itemsets are in red when the are used to assess future negations while green itemsets are used for sequential ($\rightsquigarrow_s$). Dashed arrows represent negative extensions ($\rightsquigarrow_n$) while plain arrows are sequential ($\rightsquigarrow_s$) or compositional extensions ($\rightsquigarrow_c$). Arrow label holds the item that is used in the extension.

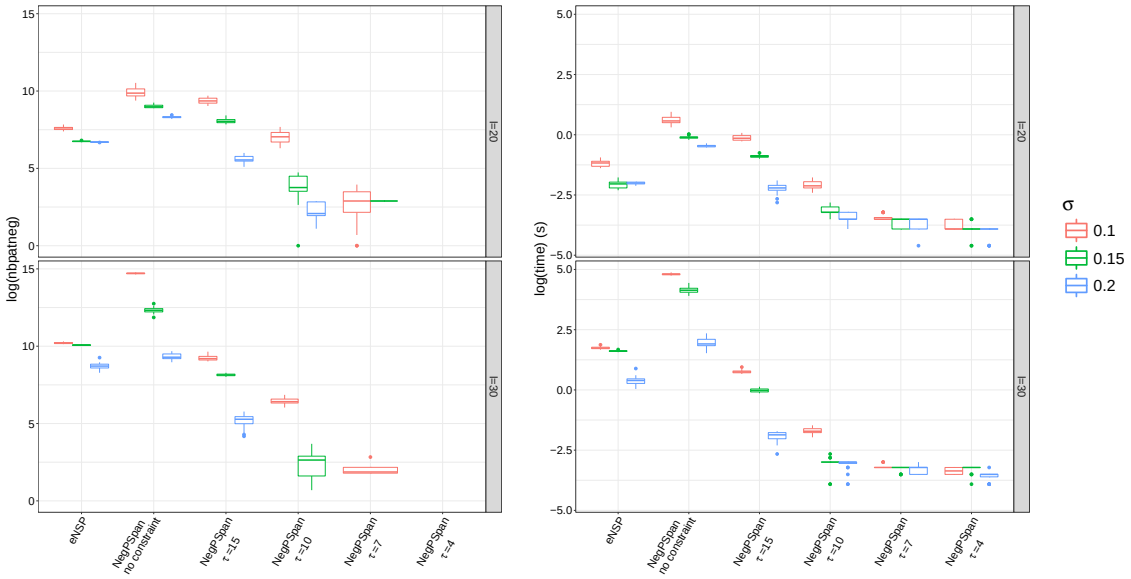

Figure 2: Comparison of number of patterns (left) and computing time (right) between eNSP and NegPSpan, with different values for maxgap (τ). Top (resp. bottom) figures correspond to database with mean sequence length equal to 20 (resp. 30). Boxplot colors correspond to different values of σ (10%, 15% and 20%).

constraints are considered. As a consequence, the significance of most of the patterns extracted by eNSP seems poor while processing "long" sequences datasets.

Figure 2 also illustrates classical results encountered with sequential pattern mining algorithms. We can note that, for both algorithms, the number of patterns and runtime increase exponentially as the minimum support decreases. Also, the number of patterns and the runtime increase notably with sequence length.

Figure 3 illustrates computation time and memory consumption with respect to minimum threshold for different settings: eNSP is ran with different values for ς, the minimal frequency of the positive partner of negative patterns (100%, 80% and 20% of the minimal frequency threshold) and NegPSpan is ran with a maxgap of 10 or without. Computation times show similar results as in previous experiments: NegPSpan becomes as efficient as eNSP with a (weak) maxgap constraint. We can also notice that the minimal frequency of the positive partners does not impact eNSP computing times neither memory requirements.

The main result illustrated by this Figure is that NegPSpan consumes significantly less memory than eNSP. This comes from the depth-first search strategy which prevents from memorizing many patterns. On the opposite, eNSP requires to keep in memory all frequent positive patterns and their occurrence list. The lower the threshold is, the more memory is required. This strategy appears to

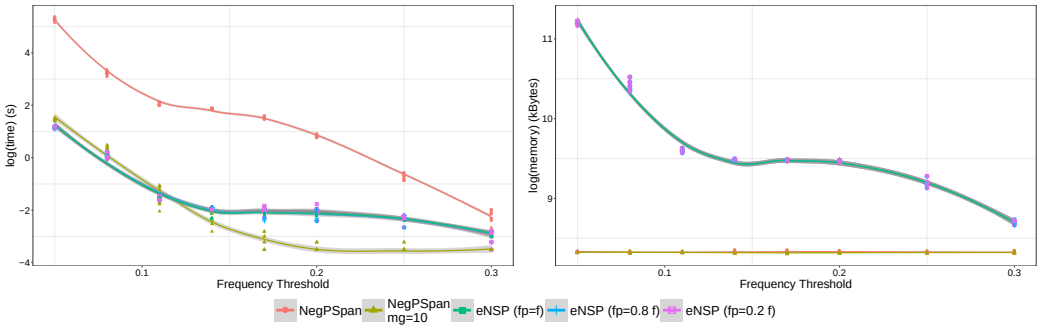

Figure 3: Comparison of computing time (left) and memory consumption (right) between eNSP and NegPSpan wrt minimal support.

be practically intractable for large/dense databases.

5.2 Experiments on Real Datasets

This section presents experiments on the real datasets from the SPMF repository.[4] These datasets consist of click-streams or texts represented as sequences of items. Datasets features and results are reported in Table 4. For every dataset, we have computed the negative sequential patterns with a maximum length of $l = 5$ items and a minimal frequency threshold set to $\sigma = 5\%$. NegPSpan is set with a maxgap $\tau = 10$ and eNSP is set up with $\varsigma = .7\sigma$. For each dataset, we provide the computation time, the memory consumption and the numbers of positive and negative extracted patterns. Note that the numbers of positive patterns for eNSP are given for ς threshold, *i.e.* the support threshold for positive partners used to generate negative patterns.

For the *sign* dataset, the execution has been stopped after 10 mn to avoid running out of memory. The number of positive patterns extracted by eNSP considering the σ threshold is not equal to NegPSpan simply because of the maxgap constraint.

The results presented in Table 4 confirm the results from experiments on synthetic datasets. First, it highlights that NegPSpan requires significant less memory for mining every dataset. Second, NegPSpan outperforms eNSP for datasets having a long mean sequence length (*Sign*, *Leviathan*, and *MSNBC*). In case of the *Bible* dataset, the number of extracted patterns by eNSP is very low compared to NegPSpan due to the constraint on minimal frequency of positive partners.

5.3 Case Study: Care Pathway Analysis

This section presents the use of NSPs for analyzing epileptic patient care pathways. Recent studies suggest that medication changes may be associated with epileptic seizures for patients with long term treatment with anti-epileptic (AE) medication [13]. NSP mining algorithms are used to extract patterns of drugs deliveries that may inform about the suppression of a drug from a patient treatment. In [3], we studied discriminant temporal patterns but it does not explicitly extract the information about medication absence as a possible explanation of epiletic seizures.

[4] `http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php`

Table 4: Results on real datasets with setting $\sigma = 5\%$, $l = 5$, $\tau = 10$, $\varsigma = .7\sigma$. Bold faces highlight lowest computation times or memory consumptions.

	Dataset			NegPSpan				eNSP			
	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{I}\rvert$	length	time (s)	mem (kb)	#pos	#neg	time (s)	mem (kb)	#pos	#neg
Sign	730	267	51.99	**15.51**	**6,220**	348	1,357,278	349.84 (!)	13,901,600	1,190,642	1,257,177
Leviathan	5,834	9,025	33.81	**6.07**	**19,932**	110	39797	28.43	428,916	7,691	17,220
Bible	36,369	13,905	21.64	38.82	**68,944**	102	43,701	**27.38**	552,288	1,364	2,621
BMS1	59,601	497	2.51	**0.16**	**22,676**	5	0	0.18	34,272	8	7
BMS2	77,512	3,340	4.62	0.37	**39,704**	1	0	**0.35**	53,608	3	2
kosarak25k	25,000	14804	8.04	0.92	**24,424**	23	409	**0.53**	43,124	50	51
MSNBC	31,790	17	13.33	**40.97**	**41,560**	613	56,418	41.44	808,744	2,441	5,439

Our dataset was obtained from the french insurance database [11] called SNIIRAM. 8,379 epileptic patients were identified by their hospitalization identified by their hospitalization related to an epileptic event. For each patient, the sequence of drugs deliveries within the 90 days before the epileptic event was obtained from the SNIIRAM. For each drug delivery, the event id is a tuple $\langle m, grp, g\rangle$ where m is the ATC code of the active molecule, $g \in \{0, 1\}$ is the brand-name (0) vs generic (1) status of the drug and grp is the speciality group. The speciality group identifies the drug presentation (international non-proprietary name, strength per unit, number of units per pack and dosage form). The dataset contains 251,872 events over 7,180 different drugs. The mean length of a sequence is 7.89±8.44 itemsets. Length variance is high due to the heterogenous nature of care pathways. Some of them represent complex therapies involving the consumption of many different drugs while others are simple case consisting of few deliveries of anti-epileptic drugs.

Let first compare results obtained by eNSP and NegPSpan to illustrate the differences in the patterns sets extracted by each algorithm. To this end, we set up the algorithms with $\sigma = 14.3\%$ $(1,200$ sequences), a maximum pattern length of $l = 3$, $\tau = 3$ for NegPSpan and $\varsigma = .1 \times \sigma$ the minimal support for positive partners for eNSP. eNSP extracts 1,120 patterns and NegPSpan only 10 patterns (including positive and negative patterns). Due to a very low ς threshold, many positive patterns are extracted by eNSP leading to generate a lot of singleton negative patterns (*i.e.* a pattern that hold a single negated item).

Table 5: Patterns involving *valproic acid* switches with their supports computed by eNSP and NegPSpan.

pattern	support eNSP	support NegPSpan
$\boldsymbol{p}_1 = \langle 383\ \neg(86, 383)\ 383\rangle$	1,579	
$\boldsymbol{p}_2 = \langle 383\ \neg 86\ 383\rangle$	1,251	1,243
$\boldsymbol{p}_3 = \langle 383\ \neg 112\ 383\rangle$	1,610	
$\boldsymbol{p}_4 = \langle 383\ \neg 114\ 383\rangle$	1,543	1,232
$\boldsymbol{p}_5 = \langle 383\ \neg 115\ 383\rangle$	1,568	1,236
$\boldsymbol{p}_6 = \langle 383\ \neg 151\ 383\rangle$	1,611	
$\boldsymbol{p}_7 = \langle 383\ \neg 158\ 383\rangle$	1,605	
$\boldsymbol{p}_8 = \langle 383\ \neg 7\ 383\rangle$		1,243

Precisely, we pay attention to the specific specialty of *valproic acid* which exists in generic form (event 383) or brand-named form (event 114) by selecting patterns that start and finish with event

383. The complete list of these patterns is given in Table 5. Other events correspond to other anti-epileptic drugs (7: *levetiracetam*, 158: *phenobarbital*) or psycholeptic drugs (112: *zolpidem*, 115: *clobazam*, 151: *zopiclone*) except 86 which is *paracetamol*.

First, it is interesting to note that with this setting, the two algorithms share only 3 patterns $\boldsymbol{p}_2$, $\boldsymbol{p}_4$ and $\boldsymbol{p}_5$, which have lower support with NegPSpan because of the maxgap constraint. This constraint also explains that pattern $\boldsymbol{p}_3$ and $\boldsymbol{p}_6$ are not extracted by NegPSpan. These patterns illustrate that in some cases, the patterns extracted by eNSP may not be really interesting because they involve distant events in the sequence. Pattern $\boldsymbol{p}_1$ is not extracted by NegPSpan due to the strict-embedding pattern semantics. With eNSP semantics, $\boldsymbol{p}_1$ means that there is no delivery of *paracetamol* and *valproic acid* at the same time. With NegPSpan semantics, $\boldsymbol{p}_1$ means that there is no delivery of *paracetamol* neither *valproic acid* between two deliveries of *valproic acid*. The latter is stronger and the pattern support is lower. On the opposite, NegPSpan can extract patterns that are missed by eNSP. For instance, pattern $\boldsymbol{p}_8$ is not extracted by eNSP because its positive partner, $\langle 383, 7, 383\rangle$, is not frequent. In this case, it leads eNSP to miss a potentially interesting pattern involving two anti-epileptic drugs.

Now, we look at patterns involving a switch from generic form to brand-named form of *valproic acid* with the following settings $\sigma = 1.2\%$, $l = 3$ and $\tau = 5$. Mining only positive patterns extracts the frequent patterns $\langle 114, 383, 114\rangle$ and $\langle 114, 114\rangle$. It is impossible to conclude about the possible impact of a switch from 114 to 383 as a possible event triggering an epileptic crisis. From negative patterns extracted by NegPSpan, we can observe that the absence of switch $\langle 114\, \neg 383\, 114\rangle$ is also frequent in this dataset. Contrary to eNSP semantics which does bring a new information (that can be deduced from frequent patterns), this pattern concerns embeddings corresponding to real interesting cases thanks to gap constraints.

6 Conclusion and Perspectives

In this article, we investigated negative sequential pattern mining (NSP). It highlights that state of the art algorithms do not extract the same patterns, not only depending on their syntax and algorithms specificities, but also depending on the semantical choices. In this article, we have proposed definitions that clarify the negation semantics encountered in the literature. We have showed that NSP support depends on the semantics of itemset non-inclusion, two possible alternatives for considering negation of itemsets and two manners for considering multiple embeddings in a sequence. This let us point out the limits of the state of the art algorithm eNSP that imposes a minimum support for positive partner and that is not able to deal with embedding constraints, and more especially maxgap constraints.

We have proposed NegPSpan a new algorithm for mining negative sequential patterns that overcomes these limitations. Our experiments show that NegPSpan is more efficient than eNSP on datasets with medium long sequences (more than 20 itemsets) even when weak maxgap constraints are applied and that it prevents from missing possibly interesting patterns.

In addition, NegPSpan is based on theoretical foundations that enable to extend it to the extraction of closed or maximal patterns to reduce the number of extracted patterns even more.

Acknowledgments

The authors would like to thank REPERES Team from Rennes University Hospital for spending time to discuss our case study results.

References

[1] Longbing Cao, Xiangjun Dong, and Zhigang Zheng. e-NSP: Efficient negative sequential pattern mining. *Artificial Intelligence*, 235:156–182, 2016.

[2] Longbing Cao, Philip S. Yu, and Vipin Kumar. Nonoccurring behavior analytics: A new area. *Intelligent Systems*, 30(6):4–11, 2015.

[3] Yann Dauxais, Thomas Guyet, David Gross-Amblard, and André Happe. Discriminant chronicles mining - application to care pathways analytics. In *Proceedings of 16th Conference on Artificial Intelligence in Medicine*, volume 10259 of *Lecture Notes in Computer Science*, pages 234–244. Springer, 2017.

[4] Xiangjun Dong, Yongshun Gong, and Longbing Cao. F-NSP+: A fast negative sequential patterns mining method with self-adaptive data storage. *Pattern Recognition*, 2018.

[5] Yongshun Gong, Tiantian Xu, Xiangjun Dong, and Guohua Lv. e-NSPFI: Efficient mining negative sequential pattern from both frequent and infrequent positive sequential patterns. *International Journal of Pattern Recognition and Artificial Intelligence*, 31(02):1750002, 2017.

[6] Sue-Chen Hsueh, Ming-Yen Lin, and Chien-Liang Chen. Mining negative sequential patterns for e-commerce recommendations. In *Proceedings of Asia-Pacific Services Computing Conference*, pages 1213–1218. IEEE, 2008.

[7] Sujatha Kamepalli, Raja Sekhara, and Rao Kurra. Frequent Negative Sequential Patterns – a Survey. *International Journal of Computer Engineering and Technology*, 5, 3:115–121, 2014.

[8] Jerry Chun-Wei Lin, Philippe Fournier-Viger, and Wensheng Gan. FHN: An efficient algorithm for mining high-utility itemsets with negative unit profits. *Knowledge-Based Systems*, 111:283–298, 2016.

[9] Chuanlu Liu, Xiangjun Dong, Caoyuan Li, and Yan Li. Sapnsp: Select actionable positive and negative sequential patterns based on a contribution metric. In *12th International Conference on Fuzzy Systems and Knowledge Discovery*, pages 811–815. IEEE, 2015.

[10] Carl H. Mooney and John F. Roddick. Sequential pattern mining – approaches and algorithms. *ACM Computing Survey*, 45(2):1–39, 2013.

[11] G. Moulis, M. Lapeyre-Mestre, A. Palmaro, G. Pugnet, J.-L. Montastruc, and L. Sailler. French health insurance databases: What interest for medical research? *La Revue de Médecine Interne*, 36:411–417, 2015.

[12] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. *IEEE Transactions on knowledge and data engineering*, 16(11):1424–1440, 2004.

[13] Elisabeth Polard, Emmanuel Nowak, André Happe, Arnaud Biraben, and Emmanuel Oger. Brand name to generic substitution of antiepileptic drugs does not lead to seizure-related hospitalization: a population-based case-crossover study. *Pharmacoepidemiology and drug safety*, 24:1161–1169, 2015.

[14] Ping Qiu, Long Zhao, and Xiangjun Dong. NegI-NSP: Negative sequential pattern mining based on loose constraints. In *43rd Annual Conference of the IECON*, pages 3419–3425. IEEE, 2017.

[15] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *International Conference on Extending Database Technology*, pages 1–17. Springer, 1996.

[16] Tiantian Xu, Xiangjun Dong, Jianliang Xu, and Xue Dong. Mining high utility sequential patterns with negative item values. *International Journal of Pattern Recognition and Artificial Intelligence*, 31(10):1750035, 2017.

[17] Tiantian Xu, Xiangjun Dong, Jianliang Xu, and Yongshun Gong. E-msNSP: Efficient negative sequential patterns mining based on multiple minimum supports. *International Journal of Pattern Recognition and Artificial Intelligence*, 31(02):1750003, 2017.

[18] Tiantian Xu, Tongxuan Li, and Xiangjun Dong. Efficient high utility negative sequential patterns mining in smart campus. *IEEE Access*, 6:23839–23847, 2018.

[19] Zhigang Zheng, Yanchang Zhao, Ziye Zuo, and Longbing Cao. Negative-GSP: An efficient method for mining negative sequential patterns. In *Proceedings of the Australasian Data Mining Conference*, pages 63–67, 2009.

[20] Zhigang Zheng, Yanchang Zhao, Ziye Zuo, and Longbing Cao. An efficient GA-based algorithm for mining negative sequential patterns. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 262–273. Springer, 2010.

A Proofs

Proof of Proposition 1. Let $\boldsymbol{s} = \langle s_1, \ldots, s_n \rangle$ be a sequence and $\boldsymbol{p} = \langle p_1, \ldots, p_m \rangle$ be a negative sequential pattern. Let $\boldsymbol{e} = (e_i)_{i \in [m]} \in [n]^m$ be a soft-embedding of pattern $\boldsymbol{p}$ in sequence $\boldsymbol{s}$. Then, the definition matches the one for strict-embedding if p_i is positive. If p_i is negative then $\forall j \in [e_{i-1}+1, e_{i+1}-1],\ p_i \not\sqsubseteq s_j$, *i.e.* $\forall j \in [e_{i-1}+1, e_{i+1}-1],\ \forall \alpha \in p_i,\ \alpha \notin s_j$ and then $\forall \alpha \in p_i,\ \forall j \in [e_{i-1}+1, e_{i+1}-1],\ \alpha \notin s_j$. It thus implies that $\forall \alpha \in p_i,\ \alpha \notin \bigcup_{j \in [e_{i-1}+1, e_{i+1}-1]} s_j$, *i.e.* by definition, $p_i \not\sqsubseteq \bigcup_{j \in [e_{i-1}+1, e_{i+1}-1]} s_j$.

The exact same reasoning is done in reverse way to prove the equivalence. □

Proof of Proposition 2 (Anti-monotonicity of NSP). Let $\boldsymbol{p} = \langle p_1\ \neg q_1\ p_2\ \neg q_2\ \ldots p_{k-1}\ \neg q_{k-1}\ p_k \rangle$ and $\boldsymbol{p'} = \langle p'_1\ \neg q'_1\ p'_2\ \neg q'_2\ \ldots p'_{k'-1}\ \neg q'_{k'-1}\ p'_{k'} \rangle$ be two NSP s.t. $\boldsymbol{p} \lhd \boldsymbol{p'}$. And let $\boldsymbol{s} = \langle s_1, \ldots, s_n \rangle$ be a sequence s.t. $\boldsymbol{p'} \preceq \boldsymbol{s}$, *i.e.* it exists an embedding $(e_i)_{i \in [k']}$:

- $\forall i,\ e_{i+1} > e_i$ (embedding), $e_{i+1} - e_i \leq \theta$ (*maxgap*) and $e_{k'} - e_1 \leq \tau$ (*maxspan*),
- $\forall i,\ p'_i \subseteq s_{e_i}$,
- $\forall j \in [e_i + 1, e_{i+1} - 1],\ q'_i \not\sqsubseteq s_{e_j}$

To prove that $\boldsymbol{p} \preceq \boldsymbol{s}$, we prove that $(e_i)_{i \in [k]}$ is an embedding of $\boldsymbol{p}$ in $\boldsymbol{s}$.
Let us first consider that $k = k'$, then by definitions of $\lhd$ and the embedding,

(i) $\forall i \in [k],\ p_i \subseteq p'_i \subseteq s_{e_i}$,

(ii) $\forall i \in [k-1],\ \forall j \in [e_i + 1, e_{e+1} - 1],\ q'_j \not\sqsubseteq s_{e_i}$, and thus $q_j \not\sqsubseteq s_{e_i}$ (because of anti-monotonicity of $\not\sqsubseteq$ and $q_i \subseteq q'i$)

In addition, *maxgap* and *maxspan* constraints are satisfied by the embedding, *i.e.*

(iv) $\forall i \in [k],\ e_{i+1} - e_i \leq \theta$

(v) $e_k - e_1 = e_{k'} - e_1 \leq \tau$

This means that $(e_i)_{i \in [k]}$ is an embedding of $\boldsymbol{p}$ in $\boldsymbol{s}$.

Let us now consider that $k' > k$, (i), (ii) and (iii) still holds, and we have in addition that $e_k < e_{k'}$ (embedding property), then $e_k - e_i < \theta$. This means that $(e_i)_{i \in [k]}$ is an embedding of $\boldsymbol{p}$ in $\boldsymbol{s}$. □

Proof of proposition 3 (Complete and correct algorithm). The correction of the algorithm is given by lines 2-3 of Algorithm 1. A pattern is outputted only if it is frequent (line 2).

We now prove the completeness of the algorithm. First of all, we have to prove that any pattern can be reached using a path of elementary transformations ($\rightsquigarrow \in \{\rightsquigarrow_n, \rightsquigarrow_s, \rightsquigarrow_c\}$). Let $\boldsymbol{p'} = \langle p'_1 \ldots p'_m \rangle$ be a pattern with a total amount of n items, $n > 0$, then it is possible to define $\boldsymbol{p}$ such that $\boldsymbol{p} \rightsquigarrow \boldsymbol{p'}$ where $\rightsquigarrow \in \{\rightsquigarrow_n, \rightsquigarrow_s, \rightsquigarrow_c\}$, and $\boldsymbol{p}$ will have exactly $n-1$ items:

- if the last itemset of $\boldsymbol{p'}$ is such that $|p'_m| > 1$ we define $\boldsymbol{p} = \langle p'_1 \ldots p'_{m-1}\ p_m \rangle$ as the pattern with the same prefix as $\boldsymbol{p'}$ and an additional itemset, p_m such that $|p_m| = |p'_m| - 1$ and $p_m \subset p'_m$: then $\boldsymbol{p} \rightsquigarrow_c \boldsymbol{p'}$

- if the last itemset of $\boldsymbol{p}'$ is such that $|p'_m| == 1$ and p'_{m-1} is positive then we define $\boldsymbol{p} = \langle p'_1 \dots p'_{m-2}\ p'm \rangle$: then $\boldsymbol{p} \rightsquigarrow_s \boldsymbol{p}'$
- if the last itemset of $\boldsymbol{p}'$ is such that $|p'_m| == 1$ and p'_{m-1} is negative (non-empty) then we define $\boldsymbol{p} = \langle p'_1 \dots p_{m-1}\ p'm \rangle$ where p_{m-1} is such that $|p_{m-1}| = |p'_{m-1}| - 1$ and $p_{m-1} \subset p'_{m-1}$: then $\boldsymbol{p} \rightsquigarrow_n \boldsymbol{p}'$

Applying recursively this rules we have that for any pattern $\boldsymbol{p}$ there is a path from the empty sequence to it: $\emptyset \rightsquigarrow^* \boldsymbol{p}$. We can also notice that there is only one possibility between the three extensions, meaning that these path is unique. This prove that our algorithm is not redundant.

Second, the pruning strategy is correct such that any frequent pattern will be missed. It is given by the anti-monotonicity property.

Let $\boldsymbol{p}$ and $\boldsymbol{p}'$ be two patterns such that $\boldsymbol{p} \rightsquigarrow \boldsymbol{p}'$ where $\rightsquigarrow \in \{\rightsquigarrow_n, \rightsquigarrow_s, \rightsquigarrow_c\}$, then is is quite obvious that $\boldsymbol{p} \lhd \boldsymbol{p}'$. Let's now consider that $\boldsymbol{p} \rightsquigarrow^* \boldsymbol{p}'$ from $\boldsymbol{p}$ to $\boldsymbol{p}'$ then, by transitivity of $\lhd$, we also have that $\boldsymbol{p} \lhd \boldsymbol{p}'$. And then by anti-monotonicity of the support, we have that $supp(\boldsymbol{p}) \geq supp(\boldsymbol{p}')$.

Let us now proceed by contradiction and consider that $\boldsymbol{p}'$ is a pattern such that $supp(\boldsymbol{p}') \geq \sigma$ but that the algorithm didn't find out. This means that for all paths[5] $\emptyset \rightsquigarrow^* \boldsymbol{p}'$ there exist $\boldsymbol{p}$ such that $\emptyset \rightsquigarrow^* \boldsymbol{p} \rightsquigarrow^* \boldsymbol{p}'$ with $supp(\boldsymbol{p}) < \sigma$. $\boldsymbol{p}$ the pattern that has been used to prune the search exploration of this path to $\boldsymbol{p}'$. This is not possible considering that $\boldsymbol{p} \rightsquigarrow^* \boldsymbol{p}'$ and thus that $supp(\boldsymbol{p}) \geq supp(\boldsymbol{p}') \geq \sigma$. □

B NegPSpan extracts a superset of eNSP

Proposition 4. Soft-embedding $\implies$ strict-embedding for patterns consisting of items.

Proof. Let $\boldsymbol{s} = \langle s_1, \dots, s_n \rangle$ be a sequence and $\boldsymbol{p} = \langle p_1, \dots, p_m \rangle$ be a NSP s.t. each $\forall i,\ |p_i| = 1$ and $\boldsymbol{p}$ occurs in $\boldsymbol{s}$ according to the soft-embedding semantic.

There exists $\epsilon = (e_i)_{i \in [m]} \in [n]^m$ s.t. for all $i \in [n]$, p_i is positive implies $p_i \in s_{e_i}$ and p_i is negative implies that for all $j \in [e_{i-1} + 1, e_{e+1} - 1]$, $p_i \notin s_j$ (items only) then $p_i \notin \bigcup_{j \in [e_{i-1}+1, e_{e+1}-1]} s_j$ *i.e.* $p_i \not\subseteq \bigcup_{j \in [e_{i-1}+1, e_{e+1}-1]} s_j$ (no matter $\not\preceq$ or $\not\sqsubset$). As a consequence ϵ is a strict-embedding of p. □

Proposition 5. Let $\mathcal{D}$ be a dataset of sequences of items and $\boldsymbol{p} = \langle p_1, \dots, p_m \rangle$ be a sequential pattern extracted by eNSP, then without embedding constraints $\boldsymbol{p}$ is extracted by NegPSpan with the same minimum support.

Proof. If $\boldsymbol{p}$ is extracted by eNSP, it implies that its positive partner is frequent in the dataset $\mathcal{D}$. As a consequence, each p_i, $i \in [m]$ is a singleton itemset.

According to the search space of NegPSpan defined by $\lhd$ if $\boldsymbol{p}$ is frequent then it will be reached by the depth-first search. Then it is sufficient to prove that for any sequence $\boldsymbol{s} = \langle s_1, \dots, s_n \rangle \in \mathcal{D}$ such that $\boldsymbol{p}$ occurs in $\boldsymbol{s}$ according to eNSP semantic (strict-embedding, strong absence), then $\boldsymbol{p}$ also occurs in $\boldsymbol{s}$ according to the NegPSpan semantics (soft-embedding, weak absence). With that and considering the same minimum support threshold, $\boldsymbol{p}$ is frequent according to NegPSpan. Proposition 4 gives this result. □

[5] Note that we proved that this path is actually unique.

Then we conclude that NegPSpan extracts more patterns than eNSP on sequences of items. In fact, NegPSpan can extract patterns with negative itemsets larger than 2.

eNSP extract patterns that are not extracted by NegPSpan on sequences of itemsets. Practically, NegPSpan uses a size limit for negative itemsets $\nu \geq 1$. eNSP extracts patterns whose positive partners are frequent. The positive partner, extracted by PrefixSpan may hold itemsets larger than ν, and if the pattern with negated itemset is also frequent, then this pattern will be extract by eNSP, but not by NegPSpan.