Skopus: Mining top-k sequential patterns under leverage

Article *in* Data Mining and Knowledge Discovery · June 2016

DOI: 10.1007/s10618-016-0467-9

François Petitjean
Monash University (Australia)
63 PUBLICATIONS **782** CITATIONS
SEE PROFILE

Geoffrey I Webb
Monash University (Australia)
377 PUBLICATIONS **6,757** CITATIONS
SEE PROFILE

Data Mining and Knowledge Discovery manuscript No.
(will be inserted by the editor)

SkOPUS

Mining top-k sequential patterns under leverage

François Petitjean · Tao Li · Nikolaj Tatti · Geoffrey I. Webb

Received: 14 December 2015 / Accepted: 23 May 2016 / Published online: 14 June 2016

Abstract This paper presents a framework for exact discovery of the top-k sequential patterns under Leverage. It combines (1) a novel definition of the expected support for a sequential pattern — a concept on which most interestingness measures directly rely — with (2) SkOPUS: a new branch-and-bound algorithm for the exact discovery of top-k sequential patterns under a given measure of interest. Our interestingness measure employs the partition approach. A pattern is interesting to the extent that it is more frequent than can be explained by assuming independence between any of the pairs of patterns from which it can be composed. The larger the support compared to the expectation under independence, the more interesting is the pattern. We build on these two elements to exactly extract the k sequential patterns with highest leverage, consistent with our definition of expected support.

We conduct experiments on both synthetic data with known patterns and real-world datasets; both experiments confirm the consistency and relevance of our approach with regard to the state of the art.

This article was published in Data Mining and Knowledge Discovery and is accessible at http://dx.doi.org/10.1007/s10618-016-0467-9.

Keywords data mining · pattern mining · sequential data · exact discovery · interestingness measures

Francois Petitjean
Faculty of Information Technology, Monash University
E-mail: francois.petitjean@monash.edu

Tao Li
School of Electronic and Information Engineering, Nanjing University of Information Science and Technology, E-mail: litao@nuist.edu.cn

Nikolaj Tatti
Department of Information and Computer Science, Aalto University
E-mail: nikolaj.tatti@aalto.fi

Geoffrey I. Webb
Faculty of Information Technology, Monash University
E-mail: geoff.webb@monash.edu

1 Introduction

Extracting interesting patterns from data is a core data mining task. This paper introduces a method to efficiently and exactly identify the top-k patterns in a *sequential* database, using as the measure of interest *leverage* — the difference between a pattern's observed and expected frequency. To define the expected frequency, we use a maximum-likelihood estimate under an assumption of independence between any pair of patters from which it can be composed.

The notion of interestingness is at the core of this paper. In early work on pattern mining, patterns were considered *interesting* if they appeared *frequently* in data [5]. The underlying idea was that the fact that something happens often is useful information for the data practitioner. However, as a mature body of research has now shown [30,43,16,44,8,15,34,39,22,47,36], *frequency* is often a poor proxy for *interestingness*. One reason for this is that many patterns should be expected to be frequent in real data. For instance, in the traditional market basket use case, if 90% of people buy apples and 90% of people buy pears, then the pattern $\{apples, pears\}$ will be frequent even if the two events are completely independent. When handling large databases, this phenomenon creates a deluge of frequent but uninteresting patterns. This is especially problematic for real-world applications, because the most frequent patterns will often be well-known; it is less frequent interactions within the data that are most likely to provide novel insights.

When tackling sequential databases, this issue becomes even more critical; patterns such as "buying apples and subsequently buying pears" will be extremely common.

This is a significant problem in real-world data, even with relatively short sequences. As a simple demonstration of this, we report in Table 1 the five most frequent sequential patterns in the book "The Adventures of Tom Sawyer", with every sentence constituting a record in the database (the rest of the paper will make it clear how we have performed this extraction).

Table 1 The five most frequent sequential patterns in the *Adventures of Tom Sawyer.*

pattern	*support*
$\langle and, and\rangle$	13%
$\langle and, to\rangle$	9.8%
$\langle to, and\rangle$	9.1%
$\langle of, and\rangle$	8.6%
$\langle and, of\rangle$	8.0%

We can observe that most frequent sequential patterns are not interesting, and simply correspond to the permutations of the most frequent word 'and' and each of the two next most frequent words 'to' and 'of'. This has motivated several approaches to mining interesting sequential patterns, which we detail in Section 2. The above example illustrates two main points about the incompatibility of frequency as a proxy for interestingness in sequential pattern discovery:

1. The main reason for the high frequency of all these patterns, is the frequency of the individual words. In English, it has been shown that one word in four comes

from $\{the, be, to, of, and, a, in, that, have, I\}$ [1]. It is then not surprising that a sentence would contain several occurrences of those. If the sequences are long enough, any sequence of independent events (which we believe will not often be of interest) will become frequent. In fact, for any pattern with probability $p > 0$ of occurring at any time in the data, the probability of its repetition tends to 100% as the length of the sequences in the database increases (this is even true for singletons).[1] This clearly demonstrates that frequency is not a good proxy for interestingness for sequential databases.

2. Let us consider the second and third most frequent sequential patterns $\langle and, to\rangle$ and $\langle to, and\rangle$. These two patterns have similar frequency. This directly questions the relevance of frequency as an interestingness measure. For sequential patterns the order of items should be key to determining whether the pattern is interesting. If all the orderings of the terms in a sequential pattern are equally frequent then an unordered pattern, such as an itemset, captures all of the information about the potentially interesting regularity in the data. We argue that the sequential pattern should be reserved for regularities that cannot be fully captured by unordered patterns. In this case, we can see that with such simple patterns of length 2, frequency ranks the two possible orderings of $\{and, to\}$ as close to equivalent. The question is then: "Should a pattern $\langle i_1, i_2\rangle$ be considered *sequentially* interesting if it appears as often as $\langle i_2, i_1\rangle$ in the database?" We argue below that this contradicts the natural definition of interestingness for sequential patterns.

In this paper, we make the two following contributions:

1. Scoring: Expected support is the core element of standard pattern mining measures of interest, such as *leverage* or *lift* [46], because most measures of interest involve a comparison between the observed support of a pattern in the data and its expected support [28]. We introduce a new definition for the expected support of a sequential pattern and present an algorithm for its computation. Following our motivation above, our approach tries to find a model that is local to the tested pattern by considering its re-orderings.
2. Search: We introduce SkOPUS: a sequential extension of the branch-and-bound OPUS algorithm [47]. SkOPUS can extract, exactly and efficiently, the k sequential patterns with highest leverage, *i.e.*, the k patterns with the highest difference between their observed and expected support. Note that we will show that our algorithm is not limited to our definition of the expected support, and can be directly used to extract the top-k patterns under any definition of the expected support.

Our paper is divided into five main sections. We present the related research in Section 2. In Section 3, we detail the proposed framework for the discovery of the top-k patterns with highest leverage. In Section 4, we present the results of experiments conducted on both synthetic data for which we control what patterns are actually present, and on real-world datasets. We conclude the paper and present some future work in Section 5.

[1] If $p(\text{"buying apples"}) > 0$ and $p(\text{"buying pears"}) > 0$ and these events are independent, then the probability of observing their sequence in data follows: $\lim_{l\to\infty} p(\langle apples, pears\rangle) = 1$

2 Related work

We structure the related work around the two main elements that this paper addresses: discovery of *sequential* and *interesting* patterns.

2.1 Mining interesting non-sequential patterns

There is a very mature body of research about non-sequential frequent pattern mining [16]. As raised in the introduction, it has now long been identified that the major issue is not whether we can derive the complete set of frequent patterns under certain constraints, but whether we can derive a compact but high-quality set of patterns that will be useful for most applications [16, 8, 34].

While the major focus has been on efficient discovery of frequent patterns, there is a growing body of research into identifying *interesting* patterns. Several methods aim at finding the set of patterns that will best describe the dataset, using a variety of methods for scoring the set such as entropy [22] or information theoretic frameworks [30, 39]. Other approaches rather try to define measures of interest for patterns [36, 18], and then perform the extraction of the most interesting patterns for different measures [7, 42, 11, 15, 47]. We refer the reader to [4], for a more complete survey on frequent pattern mining, and discovering interesting patterns.

2.2 Mining frequent sequential patterns

Sequential pattern mining extends frequent pattern mining to sequential databases [21, 26]. Real-world applications are numerous and include analysis of the purchase behaviour of customers over time, analysis of Web clickstreams, and study of biological sequences. Algorithms for mining frequent sequential patterns from sequential databases were first proposed in the 1990s [6, 23, 24]. After these seminal papers, researchers quickly moved to the development of algorithms for the extraction of *frequent* sequential patterns of higher complexity (sequences of itemsets) [25, 27, 17, 48, 29, 38, 10].

Note that different researchers have used different definitions of support and that these directly influence the patterns that they extract. Some methods, like ours, consider the support as being the number of sequences that have the sequential pattern as a subsequence, while others consider the number of times that the pattern occurs in all the sequences of the dataset, multiple counting a sequence that embeds the subsequence multiple times. Some application domains and problems will benefit from one approach while others will from the other approach. Our techniques extend directly to the second definition, should such an approach be desirable, the primary change simply being in the counting of the support of a pattern and its sub-patterns.

2.3 Mining interesting sequential patterns

It is only in the last decade that extracting interesting sequential patterns has emerged as an important research topic.

We distinguish two main families of methods:

1. methods using information theoretic approaches, such as the Minimum Description Length, to score a set of patterns that best explain the dataset.
2. methods defining an expected support for the patterns under a null model (or hypothesis), and using it to score and rank the patterns by comparison to the actual observed support of the pattern

Minimum Description Length methods scoring a set of patterns In [35,19], the authors propose a Minimum Description Length (MDL) approach for scoring a set of patterns relative to a dataset as well as a heuristic search method to construct the set. Here the idea is to score a pattern set instead of a just individual patterns; these patterns should explain the data well but at the same time be non-redundant. These goals are quantified, with scores derived from MDL principles.

Methods based on expected support Deriving an expected support under some null hypothesis is particularly complex for sequential patterns. Several approaches have made independence assumptions between the elements composing the sequential patterns. That is, they use a null hypothesis that the data are generated by a 0-order Markov model or stationary and independent stochastic process [13,31, 14,20].

Approaches to deriving expected support from Markov chains have been proposed in [12,9]; in particular, the statistical significance of the extracted patterns is studied in [9]. More complex Markov models have been studied in [2], with statistical significance studied in [3].

Tatti [32] takes a different approach to interestingness and posits that for some applications, interesting patterns will be the ones that occur in a short window. He then builds a model of the expected length of a sequence for a pattern to occur, and compares it to the actual one.

Finally, in very recent work Tatti [33] introduces an approach that is the most related to ours. This aproach takes inspiration from Webb's approach to finding interesting (non-sequential) itemsets [45], and builds the expected support by looking at different partitions of the sequential pattern. It then uses the derived score to rank the strict episodes. The key difference between our work in this paper and that in Tatti [33] is the information from which the expectation is derived: Tatti [33] uses item probabilities as well as the number of gaps in a subpattern to derive the expectation whereas, in this work, we compare the pattern against all alternative orderings of the same items. Among other technical differences, Tatti [33] defines its measure for strict episodes whereas we focus on sequential patterns, that is, serial episodes. Finally, the way we measure the difference between the observed support and expected support allows us to use a monotonic bound, and essentially mine top-k episodes without first generating a candidate set, whereas the previously discussed methods require that a candidate set be first generated, typically a set of frequent patterns mined with a low threshold.

3 Extracting the Top-K interesting sequential patterns

We introduce our method in this section. We first introduce some notation. We then detail our model for the expected frequency as well as how to derive our interestingness measure from it – leverage. Finally, we introduce our efficient search algorithm to extract the most interesting sequential patterns under these measures.

Note that we have made all code freely available to allow independent confirmation and extensions of our work at `https://github.com/fpetitjean/Skopus`.

3.1 Definitions

A *sequence* S over a finite set of items $\mathcal{I}$ is an ordered list $\langle s_1, \ldots, s_\ell \rangle$, with $s_i \in \mathcal{I}, 1 \leq i \leq \ell$, where ℓ is the length of sequence S, denoted by $|S| = \ell$. A *sequential database* $\mathcal{D} = \{S_1, S_2, \ldots, S_n\}$ of size n is a multiset of n sequences over $\mathcal{I}$, where each S_i is a sequence.

Definition 1 (Sub-sequence) A sequence $S' = \big\langle t_1, t_2, \ldots, t_\ell \big\rangle$ is a *sub-sequence* of the sequence $S = \big\langle s_1, s_2, \ldots, s_k \big\rangle$, if there exists an index sequence $1 \leq r_1 < r_2 < \cdots < r_\ell \leq k$, such that $t_j = s_{r_j}$ for all $1 \leq j \leq \ell$. In such a case, we write $s' \prec s$.

Definition 2 (Head and Tail) The *head* of a sequence is its first element and the *tail* is its remaining elements. Given an item a and a sequence T, we write $\langle a \mid T \rangle$ to mean a sequence starting with the head a followed by the tail T. For example, $\langle a \mid \langle b, c \rangle \rangle = \langle a, b, c \rangle$ and $\langle a \mid \langle\rangle \rangle = \langle a \rangle$.

Definition 3 (Cover) The *cover* of a sequence S in a sequential database $\mathcal{D}$ is the set of records of which S is a sub-sequence,

$$cover(S, \mathcal{D}) = \{S_i : S_i \in \mathcal{D}, S \prec S_i\}.$$

Such a sequence is often called *sequential pattern*.

Definition 4 (Count) The *count* of a sequence S in a sequential dataset $\mathcal{D}$ is number of records of which S is a subsequence,

$$\#(S, \mathcal{D}) = |cover(S, \mathcal{D})|.$$

Definition 5 (Support) The *support* of a pattern S is the proportion of the n records in database $\mathcal{D}$ of which S is a subsequence,

$$sup(S, \mathcal{D}) = |cover(S, \mathcal{D})|/n.$$

For notational convenience, and when it is clear in the context, we will omit $\mathcal{D}$.

Note that this paper focuses on sequential patterns and as such does not aim at extracting patterns of the type "A, then either B or C, and then D". As mentioned in related work, such patterns refer to the more general class of episodes (see for example [33]). Nor does it seek patterns of the form "sequences that contain A are more likely to contain B, irrespective of their order." Such patterns are already addressed by the substantial literature on itemset discovery.

3.2 When is a sequential pattern interesting?

We now introduce our model for the expected frequency of a given sequential pattern, as well as the interestingness measures that we derive from it. We start by providing an intuition for our framework and then introduce its formal definition.

3.2.1 From interestingness to expected support

"When is a sequential pattern interesting?" is the main question that we tackle in this paper. We have motivated in the introduction that, for example, it is unlikely that a pattern $\langle and, of \rangle$ would be interesting if the pattern $\langle of, and \rangle$ is also. It is unclear when both these patterns could be interesting except in a context where the non-sequential pattern $\{and, of\}$ holds, when the presence of *and* increases the frequency of of and vice versa, irrespective of order. Our philosophy is that such patterns are more succinctly captured as non-sequential patterns.

The starting point of our approach to assessing interestingness is that a pattern should be interesting if is not possible to explain its frequency by the frequency of its sub-patterns [40]. $\langle a, b \rangle$ should only be interesting if its frequency is greater than can be explained just by the frequency of $\langle a \rangle$ and $\langle b \rangle$. $\langle a, b, c, d \rangle$ should not be interesting if its frequency can be explained by the frequency of any of its constituent sequential sub-patterns such as $\langle a, c \rangle$, $\langle b, d \rangle$, and $\langle a, c, d \rangle$. For example, if buying shoes is often followed by buying socks, and buying jeans is often followed by buying a belt, then the pattern $\langle \text{shoes}, \text{jeans}, \text{socks}, \text{belt} \rangle$ should only be interesting if it is more frequent than should be expected given the frequency of $\langle \text{shoes}, \text{socks} \rangle$ and $\langle \text{jeans}, \text{belt} \rangle$ (as well as any other sub-sequences of this 4-element sequence).

Following the standard approach for defining interestingness measures, we assess interestingness in terms of deviation between observed support and expected support.

An intuitive first approach to deriving the expected support might be to simply multiply the supports of the two subsequences. If $\langle a \rangle$ and $\langle b \rangle$ both occur in 50% of sequences then perhaps $\langle a, b \rangle$ should be expected in 25%. However, this ignores that they may also occur in the order $\langle b, a \rangle$, suggesting that there is a need to adjust for the number of permutations that could occur. But there is a further complication — $\langle a \rangle$ and $\langle b \rangle$ can both occur in sequences of length 1 and $\langle a, b \rangle$ cannot. The length of the sequences in which the pattern appears must be taken into account, but exactly how this will affect the expectation is greatly dependent upon the type of distribution from which the data are drawn. It is far from clear what simple models might take account of all these factors without making very strong assumptions about the form of the distribution. We seek to develop a simple null hypothesis which can be tested without making any such assumptions.

3.2.2 Our proposed definition of expected support for sequential patterns

Before developing the general formulas for patterns of any length, it is instructive to consider patterns of limited length to gain intuition about our general definitions.

Pattern of length 2 We seek sequential patterns, that is, patterns in which the order of the items is significant. That is, we want a pattern $\langle a, b \rangle$ to indicate that the presence of an a in a sequence increases the probability of a b appearing subsequently. We want to distinguish such a pattern from a co-occurrence pattern, $\{a, b\}$ which indicates that a sequence that contains an 'a' is more likely to contain a 'b', irrespective of the order of their appearance. One simply hypothesis to test, which makes no assumptions about the form of the distribution from which the data are drawn, is that of all sequences in which a occurs, there are more in which b occurs after a than there are sequences in which b occurs before a. One way of stating this hypothesis is that $sup\langle a,b\rangle > sup\langle b,a\rangle$. If a statistical hypothesis test were developed, the null hypothesis would be that $sup\langle a,b\rangle \leq sup\langle b,a\rangle$.

Patterns of length 3 or more The considerations become more complicated when we consider patterns of length three. One possibility would be that we want $sup\langle a,b,c\rangle$ to be greater than the support of every other permutation of a, b and c. However, that would disallow the possibility of both $\langle a,b,c\rangle$ and $\langle a,c,b\rangle$ being interesting patterns. While we want to disallow all permutations of a set of items being interesting sequential patterns, there is no reason to mandate that only one permutation should be interesting.

It is well understood in the context of non-sequential patterns, that unless specific steps are taken to prevent patterns from being accepted that are the composition of multiple interesting sub-patterns, or the inclusion of frequent singletons into an interesting pattern, the results of a pattern discovery system will often be swamped by myriads of spurious byproducts of the core patterns [40]. One effective approach to prevent this is to test a pattern X against every partition Y, Z such that $Y \subsetneq X$ and $Z = X \backslash Y$. The counterpart of such a test for sequential patterns would test against every pair of subsequences from which it can be composed. Thus, we want to test $\langle a,b,c\rangle$ against $\langle a,b\rangle,\langle c\rangle$; $\langle a,c\rangle,\langle b\rangle$; and $\langle b,c\rangle,\langle a\rangle$. For the 'partition' $\langle a,b\rangle,\langle c\rangle$ we can compose three sequential patterns by interweaving the two patterns, $\langle c,a,b\rangle$; $\langle a,c,b\rangle$; and $\langle a,b,c\rangle$. There are two obvious criteria that might be imposed to determine whether $\mathrm{sup}\langle a,b,c\rangle$ can be explained by $\mathrm{sup}\langle a,b\rangle$ and $\mathrm{sup}\langle c\rangle$. One is that $\mathrm{sup}\langle a,b,c\rangle$ must be greater than at least one of the supports of the other compositions of the generator. The other is that $\mathrm{sup}\langle a,b,c\rangle$ must be greater than the mean of $\mathrm{sup}\langle c,a,b\rangle$; $\mathrm{sup}\langle a,c,b\rangle$; and $\mathrm{sup}\langle a,b,c\rangle$. Both approaches are credible. The latter is a stronger constraint than the former, but still allows two of the three patterns that can be composed from a two-element and one-element sequential pattern to be found to be 'interesting'. In the current work we choose this stronger approach, but our software supports both. We thus assess $\langle a,b,c\rangle$ to be 'interesting' if its support is greater than mean of the supports of the recompositions of every one of its sequential partitions.

We now turn to the general formula for the expected support of a sequential pattern, for which we first introduce the notions of *binary sequential partition* and of *composition of two sequences.*

Definition 6 (Sequential compositions) The *sequential compositions* of sequences S and T, denoted $comp(S,T)$, is the set of all sequences that contain all and only the elements of S and T, respecting their order. More formally, we define the composition recursively

$$\begin{aligned} comp(S,\langle\rangle) &= \{S\} \\ comp(\langle\rangle,T) &= \{T\} \\ comp(\langle s_1 \mid S^*\rangle,\langle t_1 \mid T^*\rangle) &= \{\langle s_1 \mid U\rangle \mid U \in comp(S^*,\langle t_1 \mid T^*\rangle)\} \\ &\quad \cup \{\langle t_1 \mid V\rangle \mid V \in comp(\langle s_1 \mid S^*\rangle,T^*)\}. \end{aligned}$$

Definition 7 (Binary sequential partition) Let S, S_1 and S_2 be three sequences. We call $\{S_1,S_2\}$ a binary sequential partition of S, if $|S_1| > 0$, $|S_2| > 0$

and $S \in comp(S_1, S_2)$. We denote as $BSP(S)$ the set of all binary sequential partitions of S,

$$BSP(S) = \{\{S_1, S_2\} \mid S \in comp(S_1, S_2)\} .$$

For example, the binary sequential partitions of

$$BSP(\langle a, b, c, d\rangle) = \{\{\langle a\rangle, \langle b, c, d\rangle\},\ \{\langle a, b\rangle, \langle c, d\rangle\},\ \{\langle a, b, c\rangle, \langle d\rangle\},\\ \{\langle a, b, d\rangle, \langle c\rangle\}, \{\langle a, c\rangle, \langle b, d\rangle\},\ \{\langle a, c, d\rangle, \langle b\rangle\},\ \{\langle a, d\rangle, \langle b, c\rangle\}\} .$$

We can now introduce our general definition for the expected support of a sequential pattern.

Definition 8 (Expected support for a sequential pattern) Let S be a sequential pattern, we define the expected support of S as

$$ExpSupport(S) = \max_{(S_1,S_2)\in BSP(S)} \{ \operatorname*{mean}_{S' \in comp(S_1,S_2)} \{sup(S')\}\}.$$

Let us give an intuition about this definition. We consider all the possible pairs of subsequences that respect the order in the target sequence (all the binary sequential partitions). Each of these partitions corresponds to a potential generator of the targeted sequence. For example, to determine the interestingness of the sequence $S = \langle a, b, c, d\rangle$, we look at how much more frequent it is than the maximum likelihood expectation if the data was generated from, say, $U = \langle a, d\rangle$ and $V = \langle b, c\rangle$ assuming that all possible ways to interweave U and V are equiprobable. We then take the maximum expected support over all potential binary sequential partitions (i.e. over all possible generators). This is because finding one pair (U, V) that 'explains' the support S should be enough to establish that pattern S is not interesting. It is important to note that we do not need to consider the expected support of compositions of more than two subsequences, as the relevant pairwise partitions will subsume compositions of more than two subsequences. The aim of this paper is to extract the top-k sequential patterns under *leverage*, i.e. that will have maximum difference between their observed and expected support:

$$leverage = sup(S) - ExpSupport(S).$$

A pattern will be considered interesting if its support cannot be explained by any of its sub-patterns.

It is finally important to note that if a pattern is considered interesting under our framework, then some of its sub-patterns might also be considered interesting. This will for example be the case with singular patterns that have low expected support. An example is the pattern $\langle if, you, can\rangle$ present in the poem "If—" by R. Kipling: it is present in more than 1/3 of the verses while none of its reorderings are, and also none of the re-orderings of its sub-patterns. For example the pattern $\langle can, you\rangle$, which is a reordering of $\langle you, can\rangle \prec \langle if, you, can\rangle$, never occurs either, thus making the pattern $\langle you, can\rangle$ interesting as well. It is however here interesting to note that our approach will rank the full pattern $\langle if, you, can\rangle$ higher than $\langle you, can\rangle$, because there are more compositions for patterns of length 3 than there are for patterns of length 2, and thus the mean over all those compositions is lower.

3.2.3 Our framework's null hypothesis

Having completely defined our proposed definition of expected support, it is important to take a step back and look at what it is achieving. **Its null hypothesis is that the frequency of a pattern can be explained by the frequency of its sub-patterns (or generators).** We have imbricated two elements:

1. We define the expected frequency for each pair of sub-patterns (what we call a *binary sequential partition*). Our null hypothesis there (and null model from which we get the expectancy) is that all possible ways to "interweave" the patterns – such that their order is respected – are equally likely. The expected frequency of a pair of generators is then the average frequency observed over all possible ways to interweave these patterns.
2. Our null hypothesis being that the frequency of a pattern can be explained by a pair of generators, the expected frequency for our framework simply considers the partition that produces the greatest expectation.

Note on the difficulty of assessing our framework's statistical significance We will demonstrate in Section 4 that our framework is very effective at extracting interesting (with all its subjectivity) patterns. It is important here to note that extracting a measure of the statistical significance of patterns under our model is a difficult task, because it requires to be able to assess the probability of a pattern given our null model. Although we can easily obtain its expected frequency, computing its probability requires more assumptions about the form of the distribution from which the data are drawn. Each sequence indeed only has a certain length, and it is unclear what model would best match our framework. We however feel that this is a consequence of the strength of our framework; we do not assume a simplistic model of the underlying distribution for all patterns. In some sense, our partitioning approach allows us to find and fit a dedicated model independently for each sequence which makes the extraction powerful, but at the cost of failing to support a simple test for assessing statistical significance.

3.2.4 Algorithmic notes

We detail here the algorithmic considerations regarding the generation of Binary Sequential Partitions (BSPs) and of Compositions.

Templates. It is first interesting to note that in either case, the results only depend on the length of the considered pattern and are independent of the actual letters themselves. All partitions and compositions are thus generated as templates depending as a function of the length.

Indexing templates. Our first optimization directly follows from this observation; every time we create a template for a BSP of a particular length, we index it with its length so that it is only computed once for each length. We employ a similar indexing for Compositions, with the difference that they have two associated lengths; we thus index them using a matrix. Note that the template for a Composition of length l_1 and l_2 is identical, we only use the upper triangle in the matrix.

Generating BSPs. Generating all Binary Sequential Partitions for a template of length l is an enumeration exercise. The first element to note is that knowing the pattern and its left partition set (S_1 in Definition 7) completely determines its right partition set (S_2 in Definition 7), and conversely. This means that, to construct the template for all BSPs of a particular length, we only need to find what is the left partition set, *i.e.*, the sequence of positions of elements in the original pattern. Furthermore, for a pattern of length l, we only need to consider S_1 of lengths l_1 from 1 to $\lfloor \frac{l}{2} \rfloor$, because of symmetry.

To generate all possible templates for S_1, we enumerate all the l_1-combinations of the set of l positions in S; there are $\binom{l}{l_1}$ such combinations. To this end, we use the standard enumeration algorithm for combinations as described in [37].

Generating Combinations. Generating combinations is simpler than BSPs. The same symmetry observations hold here, *i.e.*, the template for all combinations of lengths (l_1, l_2) is the same as the one for all combinations of lengths (l_2, l_1). We then generate all the possible l_1-combinations of the set of $l_1 + l2$ positions; here again we use the standard algorithm described in [37].

3.3 Exploring the space of all sequential patterns

Our algorithm to explore the space of sequential patterns and exactly extract the top-k sequential patterns with highest leverage[2] is based on the OPUS Miner algorithm [41, 47], first introduced for mining non-sequential databases.

3.3.1 Sequential top-k OPUS Miner (SkOPUS)

OPUS Miner finds the top-k itemsets under given interestingness measures from a non-sequential database, with regard to a given measure of interest. OPUS Miner is a depth-first branch-and-bound algorithm, that exploits the monotonicity of itemsets ($sup(I \cup i) \leqslant sup(I)$). It performs an exhaustive depth-first search of the space of sequential patterns except in so far as it can prune parts of the search space that cannot contain itemsets in the top-k with respect to the measure of interest. For example, if the minimum (*i.e.* worse) value in the top-k is α, and we know that any specialization of an itemset I will have an score that is lower than α, then the sub-tree for which the root is I does not need to be explored.

OPUS Miner's depth-first search has the advantage that the number of open nodes at any one time is minimized. This allows extensive data to be maintained for every open node and ensures that only one such record will be stored at any given time of each level of depth that is explored. It also promotes locality of memory access, as most new nodes that are opened are minor variants of the most recently explored node.

Algorithms 1 and 2 describe SkOPUS.The main elements that differ from the original OPUS Miner algorithm are as follows:

[2] In the remainder of this paper, we will use the term 'leverage' as a proxy for 'leverage under our definition of expected support'.

1. When a sequential pattern is specialized, the specializing item is always placed at the end of the sequential pattern (as a suffix – appending item i to sequence S is noted $S.i$ hereafter). This ensures that sub-spaces that are pruned will not be reconsidered later in the search.
2. When retrieving an item from the queue of available items to specialize a pattern with, the item is left in the queue (as opposed to OPUS Miner where the item is removed from the queue). This is necessary to handle repetitions of items in the patterns such as $\langle a, a, b \rangle$. Thus, we use the variant of OPUS [41] that allows multiple applications of a single operator (the addition of a given item).
3. Upper bounds for any sequential extension of a pattern S have to be adapted for our measure of interest (leverage) and our definition of expected support; we detail this element in Section 3.3.2.

Algorithm 1: SkOPUS Miner

```
Input: Sequential database $\mathcal{D}$, a measure of interest $M$, integer $k$
Output: The top-$k$ sequential patterns with regard to $M$
$q \leftarrow$ a queue of all items $i \in \mathcal{D}$ in descending order on $sup(\langle i \rangle)$;
$topK \leftarrow$ an empty queue to contain the top-$k$ sequential patterns;
return ExpandSequence($\langle\rangle$, $q$, $topK$, $M$);
```

Algorithm 2: ExpandSequence

```
Input: A sequence $S$, a queue $q$, the current $topK$ and a measure of interest $M$
Output: The top-$k$ sequential patterns with regard to $M$ in the search space explored
        to date
initialize $q'$ to be an empty queue of items;
for all $i \in q$ do
    $T \leftarrow \langle S \mid i \rangle$;
    $score \leftarrow M(T)$;
    if $score > topK.min$ then
        $topK.add(T, score)$;
    end
    if $Upper\_bound(M, T) > topK.min$ then
        $q' \leftarrow append(q', i)$;
    end
end
for all $i \in q'$ in descending order w.r.t. $M$ do
    $topK \leftarrow$ExpandSequence($T$, $q'$, $topK$, $M$);
end
return $topK$;
```

Algorithm 1 simply initializes the necessary data structures to then call Algorithm 2 that systematically explores the entire space of sequential patterns other than those branches of the search space of which it can be determined no top-k can be contained therein. The correctness of SkOPUS follows from the correctness of OPUS. As SkOPUS investigates each pattern it maintains a list *topK* of the top-k patterns found so far. Thus, on termination, as the entire search space has been explored other than those branches that cannot contain patterns in the top-k, $topK$ contains the top-k patterns.

3.3.2 Upper bounding

We now detail how to upper bound the score of any extension of a sequential pattern S with regard to leverage (*i.e.*, how to compute $Upper\_bound(M, S\star)$ in Algorithm 2). As presented earlier, we measure the interestingness of a pattern with a *leverage*,

$$leverage = sup(S) - ExpSupport(S).$$

In order to prune the search space we need an upper-bound on *leverage* for any extension pattern, say T, of a pattern S. Since $ExpSupport(S) \geq 0$ and $sup(T) \leq sup(S)$, we immediately obtain $leverage(T) \leq sup(S)$, allowing us to use $sup(S)$ as an upper bound. Note that, possibly surprisingly, the lower bound on $ExpSupport(S)$ is tight. Consider the example of a database where all sequences start with the same pattern P of length ℓ, and then do not use any of the elements composing the pattern in the rest of the sequence. We will then have all sequential compositions of all possible binary sequential partitions with 0 frequency, except for the one with the actual pattern. The highest frequency will then be for the one with the lowest number of compositions, which is ℓ.

$$ExpSupport(P) \geqslant \frac{sup(P)}{\ell}$$

Given that we want to bound all possible extensions of P, ℓ is unbounded, which makes $ExpSupport(P\star) > 0$.

In the definitions above, we exploit the fact that support and expected support are positive. It is interesting to note that, so long as the definition of expected support does not allow negative values, these elements will remain valid. It follows that our SkOPUS algorithm can be directly used with any sensible definition of expected support.

3.3.3 Bootstrap — How to quickly get a good top-k?

In practice, the time requirements of SkOPUS depend on the efficiency of the pruning mechanisms, and can vary greatly from dataset to dataset[3]. SkOPUS traverses the search space maintaining the set of top-k sequential patterns discovered in the search space explored so far. The search space can only be pruned of branches that cannot contain a pattern with higher leverage than the k^{th} best leverage found so far. Thus, efficient pruning relies critically on our ability to quickly fill $topK$ with high-scoring patterns so that as much of the search space can be pruned as possible.

We use two main strategies. First, we consider the addition (as suffix) of items with highest support first; this is apparent in the order in which we go through the different queues in Algorithms 1 and 2.

Algorithm 2 performs a depth-first search. The primary reason for doing so is that it limits the number of nodes in the search space that must be simultaneously

[3] Note that if, for instance, most of the sequential patterns in a dataset are non-interesting, then the algorithm will have to explore a large part of the space.

open. To efficiently compute the cover of a new node it is important to store the cover of the parent and only update it with respect to the item appended to the sequence. Minimizing the number of open nodes minimizes the number of covers that must be stored. However, we store the cover of every item, and so this issue does not affect two-item sequences. Our second strategy uses a hybrid search. First a breadth-first search is performed over all two-item sequences. Then the regular depth-first search is employed for sequences of length three and greater. We present the two-item breadth-first bootstrap in Algorithm 3. The algorithm remains correct as it still systematically searches all of the search space that might contain top-k patterns, all that changes is the order in which they are explored.

Algorithm 3: SkOPUS Bootstrapping

```
Input: Sequential database $\mathcal{D}$ and a measure of interest $M$
Output: A prefilled $topK$
Let $topK$ be an empty ordered queue
foreach $i_1, i_2 \in \mathcal{I}$ do
    $S \leftarrow \langle i_1, i_2 \rangle$
    $Es \leftarrow ExpSupport(S)$
    $score \leftarrow M(S, Es)$
    if $score > topK.min$ then
        $topK \leftarrow topK.add(S, score)$
    end
end
return $topK$
```

4 Experiments

In this section, we present the results of SkOPUS for the exact extraction of the k sequential patterns with highest leverage. In the synthetic experiments we compare the performance of SkOPUS using leverage as the measure of interest against SkOPUS using support as the measure of interest (included to provide a contrast against frequent pattern discovery), state-of-the-art sequential pattern discovery algorithm r_{prt} [33] and baseline r_{ind} [13]. As the synthetic experiments clearly show the baseline is less effective than either SkOPUS using leverage or r_{prt}, we do not include it in the real-world experiments.

4.1 Datasets

We shall start this section by making a general observation about the availability of interpretable sequential databases. Applications of sequential pattern mining methods are numerous and include:

1. sequences of the different webpages browsed by users on a website, where each visit represent a new transaction in the database;
2. locations of series of events, such as the neighborhoods that a taxi drives through for each client, or the restaurants visited by registered customers of say Yelp® or Urbanspoon®;

3. the performance of different industries on the share market over different days of trading;
4. the sequence of actions performed by a surgeon over the course of different surgeries;
5. the tests performed on (and action taken about) patients from admission to discharge.

The datasets associated to all these applications are however extremely valuable and hence only rarely made available freely to the scientific community. In addition, assessing the quality of technologies for pattern discovery requires having knowledge about the patterns that are actually present in data. Therefore we start by evaluating the discovery with sequences that are sampled from known distributions with specific patterns. We can then compare the discovered interactions to the true structure from which the data was sampled. This is the first set of experiments we present in Section 4.2. Then, in Section 4.3, we assess the relevance and scalability of our approach on public domain literary works. We have selected several books which we believe have characteristics that are quite representative of many other types datasets.

It is important to note that the main objective of this paper is not to prove that extracting patterns from sequential databases is an important topic; this has in fact been largely motivated by the data mining community. Rather, we aim at demonstrating 1. that interestingness-based sequential pattern extraction is critical and 2. the relevance of our approach.

All the datasets used in this paper (as well as their "ground truths", when available) are provided for reviewing purposes at `https://github.com/fpetitjean/Skopus`.

4.2 Experiments on data with known patterns

We first assess our method on datasets that embed a known set of sequential patterns.

Details of the data generation process are as follows: we start by generating a dataset with 10,000 random sequences over a vocabulary $\mathcal{I}$ with 10 tokens; the probability of the tokens follows a flat Dirichlet distribution: $\mathcal{I} \sim \text{Dir}(\mathbf{1})$. We generate each random sequence in the dataset by choosing its length l_s distributed accordingly to a shifted[4] Poisson distribution: $l_s \sim \text{Pois}(9.0) + 1$. This makes the sequences to have an average length of 10 with standard deviation 3 ($E(l_p) = 10$, $\text{Var}(l_p) = 9$).

We then generate a set of k sequential patterns, each with an associated probability of occurrence in a sequence drawn uniformly in $[0.05, 0.2]$ and length l_p following $l_p \sim \text{Pois}(1.0) + 2$, *i.e.* with average length 3 and standard deviation 1 ($E(l_p) = 10$, $\text{Var}(l_p) = 1$). Then, each pattern is embedded sequentially according to its associated probability. Embedding is performed by uniformly at random selecting insertion points in the sequence.

We then extracted the top-20 sequential patterns according to *leverage* and compare the results with a support-based extraction[5], r_{prt} [33] and r_{ind}, a baseline

[4] Shifting ensures that the sequences have at least 2 elements.

[5] Note that we "help" the support-based extraction by not allowing it to test single items, otherwise it would only return those.

Table 2 Results on data with known embedded patterns

#patterns	Recall			
	Support	Leverage	r_{prt} [33]	r_{ind} [13]
1	0%	**100%**	**100%**	**100%**
2	0%	**100%**	**100%**	50%
3	0%	**100%**	66%	66%
4	25%	**100%**	**100%**	75%
5	0%	**100%**	40%	40%
6	33%	**67%**	**67%**	33%
7	14%	**86%**	43%	43%
8	0%	**25%**	12%	12%
9	0%	**78%**	44%	22%
10	0%	**40%**	20%	10%

used by [33] using the indepdence model proposed by [13]. Both baselines require a set of candidate patterns: here we used frequent patterns with a threshold of 500. Note that this threshold potentially "helps" these two methods, because none of the injected patterns have a support lower than 500; such a threshold has thus potential to prune a significant number of patterns that could have been ranked in the top-10. We report recall rates in Table 2, that is, proportions of all patterns that were embedded that are included in the top-20 patterns returned by each approach. As expected, the support-based method does not perform well. In fact, it only ever extracted patterns of length 2; which mainly correspond to sequential patterns that are frequent because their composing items are as well. This is a similar behaviour to the one that we have explained with *and*, *to* and *of* in the introduction: patterns composed with frequent items will appear frequently by chance, without being interesting. This means that top-k extraction based on support has difficulty extracting patterns of length $\geqslant 3$.

To further highlight this point, we shall mention that for the experiment with a single embedded pattern (#patterns = 1 in Table 2), $\langle c, b, c, a\rangle$ embedded in 19% of the sequences, this pattern is ranked 92^{nd} under support while it is the top pattern under our *leverage*, r_{prt} and r_{ind}. More generally, Table 2 shows that all approaches outperform support-based approaches, by recovering a much larger number of embedded patterns.

We can also observe that our method — SkOPUS with leverage — outperforms r_{prt} and r_{ind} by obtaining a significantly higher recall of the embedded patterns. It is also interesting to note that these experiments confirm the ones in [33] by showing the poorer performance of r_{ind} compared to r_{prt}.

It can be observed that, as the number of patterns embedded in the data increases, the recall decreases. This is due to three main factors:

1. because we keep the number of patterns extracted constant ($k = 20$), it is normal that recall diminishes as the number of actual patterns increases;
2. our approach considers the subsequences of an embedded pattern to also be interesting patterns: this means even if we were to extract an actual pattern $\langle d, m, k, d\rangle$, we are also prone to extract subsequences of this patterns, such as

Table 3 Details on the datasets embedding 7 patterns.

Injected patterns	Support	Leverage
$\langle i, g, d\rangle$	17.1%	3.36%
$\langle d, e\rangle$	15.9%	0.85%
$\langle a, e, j\rangle$	14.6%	2.95%
$\langle i, b\rangle$	14.2%	4.26%
$\langle j, e, j, g\rangle$	10.5%	2.11%
$\langle j, d, c, i, a\rangle$	7.8%	3.58%
$\langle j, h, i\rangle$	7.1%	2.34%

$\langle d, m, k\rangle$ or $\langle d, m, d\rangle$. There is no counterpart of *independent productivity* from self sufficient itemset mining whereby subpatterns can be discarded [40].

3. some patterns can overlap; for example if we have two (actual) patterns $\langle a, e, e, i\rangle$ and $\langle e, f, i, d\rangle$ that are independently embedded in the data, then, the pattern $\langle e, i\rangle$ is going to be even more frequent and can be extracted; even though it is not one of the directly embedded patterns.

Note that the two last points actually complicate the extraction of patterns within the top-20 as well. These three elements are best exemplified with the top-20 sequential patterns corresponding to the dataset containing 7 patterns, which we illustrate in Table 3 and Table 4, where we report the exact matches and subpatterns that are discovered, adopting the elements appropriate to sequential patterns from the approach pioneered by Zimmermann [49] for non-sequential patterns.

First, this table further illustrates the consistency of our method. We have depicted the extracted patterns that are actual injected patterns in with a $\bullet$ next to them. While our method can extract 6 out the 7 actual patterns within the top-20, we can see that the support-based method only extracts one pattern (corresponding to a pair), and the two other methods only 3 actual patterns. It is also interesting to note that the similarity between the 4 first patterns extracted by our method and by r_{prt}.

It is informative to explain why the pattern $\langle d, e\rangle$ is not part of our top-20 while it is part of the top-20 on the *support*: it it is quite a frequent pattern (support of 63%), but this is actually mostly due to the support of d and e. This pattern was introduced with a 16% probability; the difference is explained by the fact that both tokens d and e were sampled with a relatively high probability $\geqslant 15\%$. This means that the support method only extracts it in the top-20 because d and e are frequent to start with; had they been infrequent tokens, the support wouldn't have ranked the pattern that high. This is exactly what happens with pattern $\langle i, b\rangle$, for which token b has much lower probability ($p(\langle b\rangle) = 0.02$). Because it does not appear **by chance** frequently, the support-based approach ranks the pattern very low. It is also interesting to see that under *support*, pattern $\langle e, d\rangle$ is only slightly less interesting, which clearly shows its inconsistency. Conversely, we can see the consistent behaviour of the leverage-based approach, which ranks $\langle i, b\rangle$ with much higher value than $\langle d, e\rangle$. Before this pattern was embedded into the data it already appeared in about 61% of the sequences. Observing it in 63% is thus not a very strong effect and our method ranks it accordingly. On the other hand, before introduction of $\langle i, b\rangle$, the pattern only appeared in about 13% of

Table 4 Top 20 results on the dataset embedding 7 patterns. Exact matches are represented in blue with a • next to it; subsequences of such exact matches are depicted in boldface.

#	Top-k			
	Leverage (value)	Support	r_{prt} [33]	r_{ind} [13]
1	$\langle\mathbf{i,g}\rangle$ (4.3%)	$\langle\mathbf{j,d}\rangle$	•$\langle\mathbf{j,d,c,i,a}\rangle$	•$\langle\mathbf{j,d,c,i,a}\rangle$
2	•$\langle\mathbf{i,b}\rangle$ (4.3%)	$\langle j,e\rangle$	•$\langle\mathbf{i,b}\rangle$	$\langle d,d,c,i,a\rangle$
3	•$\langle\mathbf{j,d,c,i,a}\rangle$ (3.6%)	$\langle d,d\rangle$	•$\langle\mathbf{i,g,d}\rangle$	$\langle j,j,c,i,a\rangle$
4	•$\langle\mathbf{i,g,d}\rangle$ (3.4%)	$\langle\mathbf{j,j}\rangle$	$\langle\mathbf{i,g}\rangle$	$\langle\mathbf{j,c,i,a}\rangle$
5	$\langle\mathbf{d,c,i,a}\rangle$ (3.2%)	•$\langle\mathbf{d,e}\rangle$	$\langle\mathbf{j,c,i,a}\rangle$	$\langle\mathbf{d,c,i,a}\rangle$
6	$\langle\mathbf{j,c,i,a}\rangle$ (3.2%)	$\langle d,j\rangle$	$\langle j,j,c,i,a\rangle$	$\langle j,e,c,i,a\rangle$
7	•$\langle\mathbf{a,e,j}\rangle$ (2.9%)	$\langle\mathbf{e,j}\rangle$	$\langle d,d,c,i,a\rangle$	$\langle j,e,e,j,g\rangle$
8	$\langle i,g,g\rangle$ (2.8%)	$\langle i,d\rangle$	$\langle i,g,g\rangle$	$\langle d,j,c,i,a\rangle$
9	$\langle i,e\rangle$ (2.7%)	$\langle e,e\rangle$	$\langle\mathbf{d,c,i,a}\rangle$	$\langle j,d,c,e,a\rangle$
10	$\langle\mathbf{j,d,c,a}\rangle$ (2.6%)	$\langle\mathbf{j,g}\rangle$	$\langle i,g,g,d\rangle$	$\langle i,i,b\rangle$
11	$\langle\mathbf{j,d,c,i}\rangle$ (2.6%)	$\langle\mathbf{i,d}\rangle$	$\langle j,e,c,i,a\rangle$	$\langle\mathbf{j,d,c,a}\rangle$
12	•$\langle\mathbf{j,h,i}\rangle$ (2.3%)	$\langle\mathbf{g,d}\rangle$	$\langle j,h,i\rangle$	•$\langle\mathbf{i,g,d}\rangle$
13	$\langle a,e,e,j\rangle$ (2.3%)	$\langle d,g\rangle$	$\langle\mathbf{j,d,c,a}\rangle$	$\langle\mathbf{j,d,c,i}\rangle$
14	$\langle a,e,e\rangle$ (2.3%)	$\langle i,j\rangle$	$\langle d,j,c,i,a\rangle$	•$\langle\mathbf{j,e,j,g}\rangle$
15	$\langle\mathbf{j,d,i,a}\rangle$ (2.2%)	$\langle i,e\rangle$	$\langle j,j,e,j,g\rangle$	$\langle i,g,g,d\rangle$
16	•$\langle\mathbf{j,e,j,g}\rangle$ (2.1%)	$\langle\mathbf{e,g}\rangle$	$\langle j,e,e,j,g\rangle$	$\langle j,d,e,j,g\rangle$
17	$\langle\mathbf{j,e,g}\rangle$ (2.1%)	$\langle g,j\rangle$	$\langle i,e,d\rangle$	$\langle j,e,j,j,g\rangle$
18	$\langle a,e,g\rangle$ (2.1%)	$\langle\mathbf{j,i}\rangle$	$\langle i,g,d,d\rangle$	$\langle j,d,c,i,g\rangle$
19	$\langle i,g,g,d\rangle$ (2.1%)	$\langle\mathbf{d,i}\rangle$	$\langle i,c,i,a\rangle$	$\langle j,d,c,j,a\rangle$
20	$\langle i,e,e\rangle$ (2.0%)	$\langle g,e\rangle$	$\langle j,d,c,e,a\rangle$	$\langle j,d,d,i,a\rangle$

the sequences, observing it 22% is thus more significant and our method correctly ranks it accordingly.

Moreover, we have also depicted in boldface (with no •) the patterns extracted by our approach that correspond to subsequences of actual patterns that we have also extracted. This mainly illustrates the two first elements that we have noted, by having 8 slots in our top-20 "consumed" by sub-patterns of high leverage. Although this falls out of the scope of this first attempt at extracting the top-k sequential patterns with leverage, this is naturally echoing the work that has been done on filtered-top-k association discovery [40,46]. More generally, these results call for a reflection on the evaluation of sequential pattern mining procedures, similarly to the work that has been performed in this area for non-sequential pattern mining [49].

4.3 Experiments on literary works

We compared the top patterns from several works of literature and the JMLR abstract dataset. For brevity, we only discuss the patterns obtained from JMLR. For more information about other results, see our section Supplementary material at the end of the manuscript.

We study in detail the results of the different methods on the JMLR dataset [35], which represents the abstracts of the papers published in the *Journal of Machine Learning Research*. This dataset holds 788 abstracts (hence sequences), which use 3,844 words (items). The average length of abstracts is 96 words with

Table 5 Top 10 results on abstracts of JMLR papers

	Top-k patterns		
#	Support	Leverage	r_{prt} [33]
1	⟨*algorithm, algorithm*⟩	⟨*paper, show*⟩	⟨*support, vector*⟩
2	⟨*learn, learn*⟩	⟨*paper, result*⟩	⟨*support, vector, machin*⟩
3	⟨*learn, algorithm*⟩	⟨*support, vector, machin*⟩	⟨*support, machin*⟩
4	⟨*algorithm, learn*⟩	⟨*paper, algorithm*⟩	⟨*real, world*⟩
5	⟨*data, data*⟩	⟨*support, vector*⟩	⟨*vector, machin*⟩
6	⟨*learn,data*⟩	⟨*base, result*⟩	⟨*state, art*⟩
7	⟨*model, model*⟩	⟨*paper, method*⟩	⟨*reproduc, hilbert*⟩
8	⟨*problem, problem*⟩	⟨*learn,result*⟩	⟨*high, dimension*⟩
9	⟨*learn, result*⟩	⟨*paper, propos*⟩	⟨*first, second*⟩
10	⟨*problem, algorithm*⟩	⟨*vector,machin*⟩	⟨*larg, scale*⟩

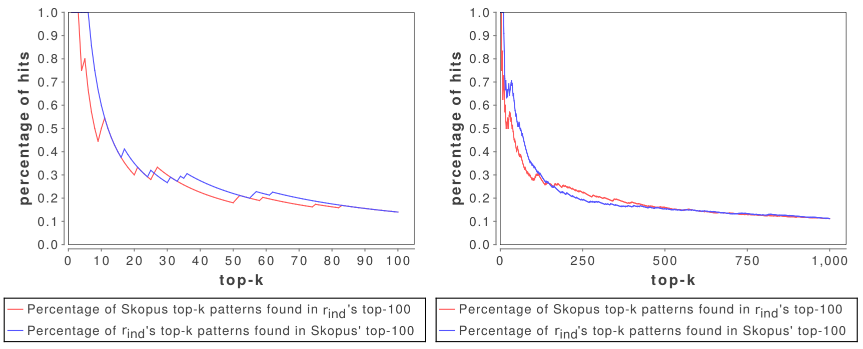
a maximum length of 231. Since r_{prt} requires a candidate set, we use frequent patterns with a threshold of 10. For the other literature works we used a threshold of 5.

We first present the results and detail the computation times in the next subsection. The top-10 patterns are presented in Table 5 for our method (leverage), support for reference, and the results from r_{prt}; having shown in the previous section that our method and r_{prt} outperform r_{ind}, we focus on other methods and keep top-support patterns for reference.

The first critical observation echoes the ones we made in the introduction about the inconsistency of support-based extraction: when using the support, most of the patterns correspond in repetitions of very frequent words, such as *algorithm*, *learn*, *data*, *model* and *problem*. Moreover we can see that when using the support as the measure of interest, the pattern $\langle learn, algorithm\rangle$ and its reversed version $\langle algorithm, learn\rangle$ appear with very similar scores (resp. supports 36% and 29%). In contrast, we can observe that our method and r_{prt} present patterns that seem of higher general interest than support. Interestingly, the patterns extracted seem to differ significantly between the methods. Beyond highlighting the importance of synthetic data experiments, this also re-confirms the subjectivity of interestingness. We find then interesting to examine two elements:

1. The overlap of the different methods.
2. The patterns that crystallize the differences between the methods.

Overlap between SkOPUS and r_{prt} For each method, we examine what percentage of its top-k is found in the top-k of the other. Figure 1 presents this overlap analysis for SkOPUS and r_{prt}: in (a) their top-100 and (b) their top-100. For example, a point at coordinates (x, y) for the top-100 tells us that y percent of the top-x of the first method was found in the top-100 of the other. Two similar results would then have a slowly decreasing rate. In this case we observe quite the opposite: there is only little overlap between the two methods with less than 20% overlap within the top-100 and top-1000. Interestingly, SkOPUS seem to find more of the top patterns ranked by r_{prt} than the opposite, which supports the relevance of our method.



(a) Top-100 (b) Top-1000

Fig. 1 Overlap between the top-k given by SkOPUS and r_{prt}

Analysis of representative differences Significant differences exist even in the very top patterns, which clearly appears in Figure 1(a), which shows that patterns as early as in SkOPUS' third position are not part of the top-100 of the patterns ranked by r_{prt}. We detail below a few such example patterns that are highly contrasting the differences between SkOPUS and r_{prt}:

- Patterns SkOPUS ranks high but not r_{prt}: $\langle paper, algorithm\rangle$ is extracted as the 4th pattern with highest leverage while it is ranked 7946 by r_{prt}. This pattern is extracted by SkOPUS for its leverage because this succession appears 174 times in the dataset, while its reversed pattern $\langle algorithm, paper\rangle$ appears only 80 times. It seems reasonable to be highly rank a pattern that occurs with more than twice the frequency of its reverse. r_{prt} ranks this pattern much lower because *paper* and *algorithm* are both probable individually (resp. support of 43% and 58%). A similar phenomenon is seen for $\langle base, result\rangle$ — which SkOPUS ranks 6 vs. 650 for r_{prt} — and for $\langle learn, result\rangle$— which SkOPUS ranks 8 vs. $25,810$ for r_{prt}.
- Patterns r_{prt} ranks high but not SkOPUS: $\langle reproduc, hilbert\rangle$ is extracted as the 7th pattern by r_{prt} while it is ranked 696 by SkOPUS. This pattern is extracted by r_{prt} because it consists of two relatively rare items, *reproduc* occurs 32 times and *hilbert* occurs 36 times. On the other hand, SkOPUS ranks this pattern relatively low because it appears in only 28 abstracts. While under our measure of expected support this pattern has high lift, it has low leverage, the latter being a measure that favors patterns that appear the greatest number of times in excess of the expected. It is also interesting to note that SkOPUS ranks the pattern $\langle reproduc, hilbert, spac\rangle$ much higher than $\langle reproduc, hilbert\rangle$ (390 vs. 696). It seems natural that if all abstracts that include $\langle reproduc, hilbert\rangle$ are followed by *space*, then the pattern of length 3 should be more interesting. SkOPUS' rank respects this ordering while r_{prt} does not in this case, having the pattern $\langle reproduc, hilbert, spac\rangle$ ranked 42nd. Note also that the pattern $\langle hilbert, spac\rangle$ occurs less frequently than $\langle reproduc, hilbert\rangle$, as some papers mention "reproduc[ing] the Hilbert **norm**", but not space (see e.g. the paper by F. Bach at `http://www.jmlr.org/papers/v9/bach08b.html`).

4.4 Execution time

We finish the experiments by showing the running time of our approach; all experiments are performed using a standard desktop computer.

Results are reported in Table 6. Not surprisingly, extracting the top-20 under support with SkOPUS is extremely fast, because the most frequent elements are encountered very early in the exploration of the search space;[6] no top-20 were actually presenting patterns of more than length 2 regardless of the experiments.

For other methods, it can be observed than synthetic data are extremely challenging, mostly because we included only a few patterns with reasonable probability. Results are obtained in a few hours for SkOPUS under leverage and less than a minute for r_{prt}. The main reason why r_{prt} is faster is due to a high mining threshold of 500 without which extraction cannot be performed, because if we lower this threshold then the mining step will take a considerate time due to a frequent pattern explosion. On the other hand, SkOPUS does not require any threshold.

Finally, our approach exhibits extremely competitive running time compared to support on real-world datasets where interesting patterns are more present. It only takes SkOPUS 37s to extract the top-20 patterns in the JMLR dataset; only slightly more than twice the time taken for support-extraction, and significantly faster than r_{prt} with minimum support set to 10. On other literary works, SkOPUS finishes in less than a few minutes, while r_{prt} is generally finishes in seconds but it required to set a minimum support threshold of 5.

It is finally interesting to note that the algorithmic complexity of SkOPUS and r_{prt} varies with different elements. For SkOPUS, it is a function of how interesting the patterns in the data are: data that holds patterns of high interest will prune significantly large parts of the search space (and the earlier they are found, the quicker the process). For r_{prt}, the interests of the patterns that the data holds is almost neutral to the complexity, which will mostly vary as a function of the number of frequent closed patterns.

5 Conclusion

This paper introduced a new general definition for the expected support of sequential patterns, which specifically focuses on the order within the pattern.

We have described the intuition behind our definitions, as well as efficient algorithms for both the computation of the expected support and the exact exploration of the search space. Put together, these contributions allowed us to introduce SkOPUS which constitutes, to the best of our knowledge, the first framework for the exact mining of the k sequential patterns with highest leverage from data.

Experiments on controlled data have validated the consistency and relevance of our framework, relevance that we have then confirmed on literary work.

This work naturally raised a number of questions and issues that, we anticipate, will be of great interest to the community in the future. These include:

[6] Although *support* is often not a good proxy for interestingness, it remains that SkOPUS can be very efficiently and exactly extract the most frequent sequential patterns.

Table 6 Running time on the different datasets for extracting the top-20.

	Runtimes		
Name	Support	Leverage	r_{prt} [33] (mining + ranking)
Synth-1	< 1s	1h41m57s	5s + 4s
Synth-2	< 1s	1h44m28s	5s + 5s
Synth-3	< 1s	53m19s	9s + 9s
Synth-4	< 1s	1h24m40s	6s + 6s
Synth-5	< 1s	1h34m41s	7s + 7s
Synth-6	< 1s	3h55m33s	10s + 11s
Synth-7	< 1s	6h5m31s	18s + 19s
Synth-8	< 1s	1h25m25s	16s + 14s
Synth-9	< 1s	2h28m30s	21s + 24s
Synth-10	< 1s	8h58m53s	8m15s + 6m16s
JMLR	16s	37s	6m47s + 1m3s
Lawyers	26s	3m51s	13s + 2s
Finn	6s	1m8s	5s + 2s
Sawyer	7s	1m23s	2s + 1s
NewPhysics	5s	48s	3s + 1s
TwoCities	14s	2m16s	6s + 2s
Animals	8s	1m46s	1s + 1s

1. finding heuristics to quickly fill the temporary top-k with patterns of high-interest, possibly by integrating some background knowledge;
2. refining upper bounds for the most common interestingness measures;
3. filtering the results to remove trivial subpatterns from those that are discovered (patterns highlighted in boldface in Table 3);
4. assessing the statistical significance of the extracted patterns, so that only patterns that have high-likelihood to be observed in future data would be extracted (as initiated by [9,3]);
5. using background knowledge to define a model of the joint distribution, and then extract the patterns that differ the most from it; such an approach has been investigated by S. Jaroszewicz on non-sequential patterns, and models the joint distribution with a Bayesian Network [18];
6. many domains hold patterns of the type “admission, then biopsy or blood-test, and then surgery”. Our work focuses on the extraction of chains of events (serial episodes); we believe that extension to the more general classes of episodes will be of major interest.

Acknowledgements We are grateful to the anonymous reviewers and to Matteo Riondato for valuable and constructive comments on this manuscript.

Supplementary material

For completeness, we make available the top-100 patterns extracted from six literary works (see Table 7) and for all the methods discussed at `https://github.com/fpetitjean/Skopus/`.

Table 7 Characteristics of the literary datasets

Name	$\|\mathcal{I}\|$	$\|\mathcal{D}\|$	Avg. length	Max. length
The Industries of Animals	5547	4931	12.8	107
A Book About Lawyers	8386	4787	21.9	211
Adventures of Huckleberry Finn	4504	6402	13.7	213
A Tale of Two Cities	6392	8584	12.4	150
The Adventures of Tom Sawyers	5073	5259	10.7	119
The New Physics and Its Evolution	3832	3081	19.4	81

Compliance with Ethical Standards

This material is based upon work supported by the Air Force Office of Scientific Research, Asian Office of Aerospace Research and Development (AOARD) under award number FA2386-15-1-4007. This research was also supported by China Special Fund for Meteorological Research in the Public Interest under grant GYHY201306070, Jiangsu Government Scholarship for Overseas Studies; and by the Australian Research Council under grant DP140100087.

References

1. The Oxford English Corpus: Facts about Language. In: Oxford Dictionaries. Oxford University Press (2015). http://www.oxforddictionaries.com/words/the-oec-facts-about-the-language
2. Achar, A., Laxman, S., Viswanathan, R., Sastry, P.: Discovering injective episodes with general partial orders. Data Mining and Knowledge Discovery **25**(1), 67–108 (2012)
3. Achar, A., Sastry, P.: Statistical significance of episodes with general partial orders. Information Sciences **296**(0), 175–200 (2015)
4. Aggarwal, C.C., Han, J. (eds.): Frequent pattern mining. Springer (2014)
5. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM-SIGMOD International Conference on Management of Data, pp. 207–216. Washington, DC (1993)
6. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the Eleventh International Conference on Data Engineering, March 6-10, 1995, Taipei, Taiwan, pp. 3–14. IEEE Computer Society (1995)
7. Bayardo Jr, R.J., Agrawal, R.: Mining the most interesting rules. In: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 145–154. ACM (1999)
8. Boley, M., Horváth, T., Wrobel, S.: Eficient discovery of interesting patterns based on strong closedness. Statistical Analysis and Data Mining **2**(5-6), 346–360 (2009)
9. Castro, N.C., Azevedo, P.J.: Significant motifs in time series. Statistical Analysis and Data Mining: The ASA Data Science Journal **5**(1), 35–53 (2012)
10. Fournier-Viger, P., Gomariz, A., Gueniche, T., Mwamikazi, E., Thomas, R.: TKS: efficient mining of top-k sequential patterns. In: Advanced Data Mining and Applications, 9th International Conference, ADMA 2013, Hangzhou, China, December 14-16, 2013, Proceedings, Part I, *Lecture Notes in Computer Science*, vol. 8346, pp. 109–120. Springer (2013)
11. Geng, L., Hamilton, H.J.: Choosing the right lens: Finding what is interesting in data mining. In: Quality Measures in Data Mining, pp. 3–24. Springer (2007)
12. Gwadera, R., Atallah, M.J., Szpankowski, W.: Markov models for identification of significant episodes. In: SIAM International Conference on Data Mining, pp. 404–414 (2005)
13. Gwadera, R., Atallah, M.J., Szpankowski, W.: Reliable detection of episodes in event sequences. Knowledge and Information Systems **7**(4), 415–437 (2005)

14. Gwadera, R., Crestani, F.: Ranking sequential patterns with respect to significance. In: Advances in Knowledge Discovery and Data Mining, 14th Pacific-Asia Conference, PAKDD 2010, Hyderabad, India, June 21-24, 2010. Proceedings. Part I, *Lecture Notes in Computer Science*, vol. 6118, pp. 286–299. Springer (2010)
15. Hämäläinen, W.: Efficient discovery of the top-k optimal dependency rules with Fisher's exact test of significance. In: ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010, pp. 196–205. IEEE Computer Society (2010)
16. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. Data Mining and Knowledge Discovery **15**(1), 55–86 (2007)
17. Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., Hsu, M.: Freespan: frequent pattern-projected sequential pattern mining. In: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, MA, USA, August 20-23, 2000, pp. 355–359. ACM (2000)
18. Jaroszewicz, S., Simovici, D.A.: Interestingness of frequent itemsets using bayesian networks as background knowledge. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 178–196. ACM (2004)
19. Lam, H.T., Moerchen, F., Fradkin, D., Calders, T.: Mining compressing sequential patterns. Statistical Analysis and Data Mining: The ASA Data Science Journal **7**(1), 34–52 (2014)
20. Low-Kam, C., Raïssi, C., Kaytoue, M., Pei, J.: Mining statistically significant sequential patterns. In: 2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, December 7-10, 2013, pp. 488–497. IEEE Computer Society (2013)
21. Mabroukeh, N.R., Ezeife, C.I.: A taxonomy of sequential pattern mining algorithms. ACM Computing Surveys (CSUR) **43**(1), 3 (2010)
22. Mampaey, M., Vreeken, J., Tatti, N.: Summarizing data succinctly with the most informative itemsets. ACM Transactions on Knowledge Discovery from Data **6**(4), 16 (2012)
23. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovering frequent episodes in sequences. In: Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95), Montreal, Canada, August 20-21, 1995, pp. 210–215. AAAI Press (1995)
24. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery **1**(3), 259–289 (1997)
25. Masseglia, F., Cathala, F., Poncelet, P.: The PSP approach for mining sequential patterns. In: Principles of Data Mining and Knowledge Discovery, Second European Symposium, PKDD '98, Nantes, France, September 23-26, 1998, Proceedings, *Lecture Notes in Computer Science*, vol. 1510, pp. 176–184. Springer (1998)
26. Mooney, C.H., Roddick, J.F.: Sequential pattern mining–approaches and algorithms. ACM Computing Surveys (CSUR) **45**(2), 19 (2013)
27. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Prefixspan: Mining sequential patterns by prefix-projected growth. In: Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany, pp. 215–224. IEEE Computer Society (2001)
28. Piatetsky-Shapiro, G.: Discovery, analysis, and presentation of strong rules. In: G. Piatetsky-Shapiro, J. Frawley (eds.) Knowledge Discovery in Databases, pp. 229–248. AAAI/MIT Press, Menlo Park, CA. (1991)
29. Raïssi, C., Calders, T., Poncelet, P.: Mining conjunctive sequential patterns. Data Mining and Knowledge Discovery **17**(1), 77–93 (2008)
30. Siebes, A., Vreeken, J., van Leeuwen, M.: Item sets that compress. In: Proceedings of the Sixth SIAM International Conference on Data Mining, April 20-22, 2006, Bethesda, MD, USA, pp. 395–406. SIAM (2006)
31. Tatti, N.: Significance of episodes based on minimal windows. In: IEEE International Conference on Data Mining, pp. 513–522 (2009)
32. Tatti, N.: Discovering episodes with compact minimal windows. Data Mining and Knowledge Discovery **28**(4), 1046–1077 (2014)
33. Tatti, N.: Ranking episodes using a partition model. Data Mining and Knowledge Discovery pp. 1–31 (2015)
34. Tatti, N., Mampaey, M.: Using background knowledge to rank itemsets. Data Mining and Knowledge Discovery **21**(2), 293–309 (2010)
35. Tatti, N., Vreeken, J.: The long and the short of it: Summarising event sequences with serial episodes. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 462–470 (2012)

36. Tew, C.V., Giraud-Carrier, C.G., Tanner, K.W., Burton, S.H.: Behavior-based clustering and analysis of interestingness measures for association rule mining. Data Mining and Knowledge Discovery **28**(4), 1004–1045 (2014)
37. Tucker, A.: Applied Combinatorics. John Wiley & Sons, Inc., New York, NY, USA (2006)
38. Tzvetkov, P., Yan, X., Han, J.: TSP: mining top-k closed sequential patterns. In: Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), 19-22 December 2003, Melbourne, Florida, USA, pp. 347–354. IEEE Computer Society (2003)
39. Vreeken, J., van Leeuwen, M., Siebes, A.: Krimp: mining itemsets that compress. Data Mining and Knowledge Discovery **23**(1), 169–214 (2011)
40. Webb, G.: Self-sufficient itemsets: An approach to screening potentially interesting associations between items. Transactions on Knowledge Discovery from Data **4**, 3:1–3:20 (2010)
41. Webb, G.I.: OPUS: an efficient admissible algorithm for unordered search. Journal of Artificial Intelligence Research (JAIR) **3**, 431–465 (1995)
42. Webb, G.I.: Efficient search for association rules. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 99–107. ACM (2000)
43. Webb, G.I.: Discovering significant patterns. Machine Learning **68**(1), 1–33 (2007)
44. Webb, G.I.: Layered critical values: A powerful direct-adjustment approach to discovering significant patterns. Machine Learning **71**(2-3), 307–323 (2008)
45. Webb, G.I.: Self-sufficient itemsets: An approach to screening potentially interesting associations between items. ACM Transactions on Knowledge Discovery from Data (TKDD) **4**(1), 3 (2010)
46. Webb, G.I.: Filtered-top-k association discovery. Wiley Interdisc. Rev.: Data Mining and Knowledge Discovery **1**(3), 183–192 (2011)
47. Webb, G.I., Vreeken, J.: Efficient discovery of the most interesting associations. ACM Transactions on Knowledge Discovery from Data **8**(3), 1–31 (2014)
48. Yan, X., Han, J., Afshar, R.: Clospan: Mining closed sequential patterns in large databases. In: Proceedings of the Third SIAM International Conference on Data Mining, San Francisco, CA, USA, May 1-3, 2003, pp. 166–177. SIAM (2003)
49. Zimmermann, A.: Objectively evaluating interestingness measures for frequent itemset mining. In: J. Li, L. Cao, C. Wang, K. Tan, B. Liu, J. Pei, V. Tseng (eds.) Trends and Applications in Knowledge Discovery and Data Mining, *Lecture Notes in Computer Science*, vol. 7867, pp. 354–366. Springer Berlin Heidelberg (2013)