

# A new constraint for mining sets in sequences<sup>1</sup>

Boris Cule <sup>a</sup>

Bart Goethals <sup>a</sup>

Céline Robardet <sup>b</sup>

<sup>a</sup> *University of Antwerp, Middelheimlaan 1, B-2020 Antwerpen, Belgium*

<sup>b</sup> *Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, F-69621, France*

Discovering interesting episodes is a popular area in temporal or sequential data mining, examples of which are mining text or protein sequences. In such data, the order in which the events appear is being analysed and the user's goal is to identify the regularities that may appear in the dataset, consisting of one or more sequences. The usual approach to episode discovery is to look for episodes consisting of events that frequently appear close to each other. Most of the current state-of-the-art methods first use a window of fixed length to find sufficiently cohesive episodes and then retrieve those that occur in more windows (or sequences) than a given minimum threshold. The frequency of an itemset  $X$ ,  $fr(X)$ , is thus defined as the number of windows  $X$  appears in divided by the total number of possible windows. The use of a window of fixed length is a major limitation of such approaches as no episodes longer than this window can ever be discovered.

A different method that increases the window length proportionally to the size of the candidate set has been proposed in order to remove this limitation. Still, in this proposal, the window length remains fixed for a particular candidate when counting its frequency in the sequence. Hence, when the episode occurs in the sequence, but in a time frame larger than the window size, then such occurrences will be disregarded. The high frequency of a set of events appearing close together gives no guarantee that a subset of that set will not sometimes appear far away from the rest of the set.

Take, for example, the following sequence:

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Stamps																				
Sequence	c	g	h	e	f	a	b	a	b	c	i	d	j	c	k	d	l	c	m	d

Using a window of fixed length will either not discover itemset  $cd$  at all, or will consider it more interesting than itemset  $ab$ , even though  $a$  and  $b$  always occur right next to each other. For example, if the chosen window size is 3, we will get the following frequencies:

$$fr(cd) = \frac{5}{22}, fr(ab) = \frac{4}{22}$$

where 22 is the number of all possible windows of size 3. Furthermore, the occurrence of a  $c$  right at the beginning of the sequence, far away from any  $d$ , is simply ignored and has no effect on the value of itemset  $cd$ , whereas it clearly should reduce it.

In this paper we propose a new constraint to select interesting sets of events. We focus on parallel episodes which are unordered sets of events, and can thus be considered as itemsets. We therefore consider an event in a sequence to be a couple consisting of an item and a time stamp  $(i, t)$  where  $i \in I$ , the set of all possible items, and  $t \in \mathbb{N}$ . We assume that two items in a sequence can never occur at the same time. We denote a sequence of such events by  $\mathcal{S}$ .

We define the interestingness of an itemset based on a combination of how often the items in the set appear in the sequence and how close to each other they appear on average. Hence, this approach does not use a fixed window length but instead also takes the occurrences of the items far from the rest of the set into account. It is precisely such occurrences that might sufficiently lower the cohesion of an itemset to render it uninteresting.

---

<sup>1</sup>This is an extended abstract of a paper published in the Proceedings of the 2009 SIAM DM conference; Sparks, Nevada, USA

When looking at one sequence, we first define the *coverage* of an itemset  $X$  as the probability of encountering an item from that itemset in the sequence:  $P(X) = \frac{|N(X)|}{|\mathcal{S}|}$ , where  $N(X) = \{t \mid (i, t) \in \mathcal{S} \text{ and } i \in X\}$ , i.e. the set of all time stamps at which an item in  $X$  occurs. We then look for a minimal window containing the whole itemset  $X$  around each occurrence of one of its items:  $W(X, t) = \min\{t_2 - t_1 + 1 \mid t_1 \leq t \leq t_2 \text{ and } \forall i \in X, \exists(i, t') \in \mathcal{S}, t_1 \leq t' \leq t_2\}$ .

We define the *cohesion* of an itemset as the ratio between the itemset’s length and the average length of all such minimal windows:  $C(X) = \frac{|X|}{\overline{W}(X)}$ , where  $\overline{W}(X) = \frac{\sum_{t \in N(X)} W(X, t)}{|N(X)|}$ . Finally, we define the *interestingness* of an itemset as the product of its coverage and its cohesion:  $I(X) = C(X)P(X)$ .

Itemsets are considered interesting if their interestingness is higher than a threshold chosen by the user. As both the coverage and the cohesion are values between 0 and 1, the same is true for the interestingness, which makes it possible to apply the same interestingness threshold to any kind of dataset. Given these definitions, we can claim the following: if we encounter an item from an interesting itemset, we can be reasonably certain that the rest of the itemset can be found nearby.

Looking back at the example above, we can again compare the values of itemsets  $ab$  and  $cd$ . The relevant results are summed up in the following table:

$X$	$ N(X) $	$P(X)$	$\overline{W}(X)$	$C(X)$	$I(X)$
$ab$	4	0.2	2	1	0.2
$cd$	7	0.35	4.29	0.47	0.16

We can see that the fact that  $ab$  is fully cohesive (i.e.  $C(ab) = 1$ ), while  $cd$ ’s cohesion is far lower, results in  $cd$  having a lower interestingness despite its higher coverage. Our method thus gives the desired results.

The introduced interestingness measure has useful properties that allow us to develop an efficient algorithm to search for interesting itemsets, depending on a user-defined threshold. Our algorithm generates candidates using a divide-and-conquer method, traversing the search tree, and uses a pruning technique based on an upper bound of the interestingness in order to reduce, as soon as possible, the number of candidate itemsets. This upper bound is constructed in such a way that, when evaluating a node in the search tree, it holds for all candidates within the subtree rooted at that node, so when the upper bound is smaller than the interestingness threshold, we can safely prune the whole subtree.

We present a similar definition for datasets consisting of many sequences. Here, the coverage of an itemset is defined as the probability of finding the whole itemset in a single sequence within the set of sequences. We now look for a minimal window containing the whole itemset within each sequence that contains it. Cohesion of an itemset is then defined as the ratio between the itemset’s length and the average length of all such minimal windows. Finally, the interestingness of an itemset is defined as the product of its coverage and its cohesion. Once again, these definitions allow us to claim the following: if we encounter a sequence containing all items from an interesting itemset, we can be reasonably certain that these items can be found near each other. Here, too, we present an efficient algorithm for mining such interesting itemsets, based on a similar pruning technique.

After applying our methods to various datasets consisting of both a single sequence and of multiple sequences, we could see that they give intuitive results. We used synthetic datasets to show that we get expected results and no spurious output, and to test the efficiency of our algorithms. We used real-life datasets (music, DNA and text sequences) to show that our method produced output of interest to various types of users. For example, when we ran our algorithm for multiple sequences on the dataset consisting of the abstracts of papers accepted at the 2008 ECML PKDD conference (preprocessed to remove stop words), these were the most interesting pairs of (stems of) words we found:

$X$	$ N(X) $	$P(X)$	$\overline{W}(X)$	$C(X)$	$I(X)$
$\{real, world\}$	9	0.16	2	1	0.16
$\{dimension, reduct\}$	6	0.11	2	1	0.11
$\{semi, supervis\}$	6	0.11	2	1	0.11
$\{state, art\}$	6	0.11	2	1	0.11
$\{experiment, result\}$	6	0.11	2	1	0.11
$\{learn, task\}$	7	0.13	2.57	0.78	0.1
$\{dimension, low\}$	5	0.09	2	1	0.09