Mining Association Rules in Long Sequences

Boris Cule and Bart Goethals

University of Antwerp,
Middelheimlaan 1, 2020 Antwerpen, Belgium
{boris.cule,bart.goethals}@ua.ac.be

Abstract. Discovering interesting patterns in long sequences, and finding confident association rules within them, is a popular area in data mining. Most existing methods define patterns as interesting if they occur frequently enough in a sufficiently cohesive form. Based on these frequent patterns, association rules are mined in the traditional manner. Recently, a new interestingness measure, combining cohesion and frequency of a pattern, has been proposed, and patterns are deemed interesting if encountering one event from the pattern implies with a high probability that the rest of the pattern can be found nearby. It is quite clear that this probability is not necessarily equally high for all the events making up such a pattern, which is why we propose to introduce the concept of association rules into this problem setting. The confidence of such an association rule tells us how far on average from a particular event, or a set of events, one has to look, in order to find the rest of the pattern. In this paper, we present an efficient algorithm to mine such association rules. After applying our method to both synthetic and real-life data, we conclude that it indeed gives intuitive results in a number of applications.

Key words: association rules, sequences, interesting itemsets

1 Introduction

Pattern discovery in sequences is a popular data mining task. Typically, the dataset consists of a single event sequence, and the task consists of analysing the order in which events occur, trying to identify correlation among events. In this paper, events are items, and we look for association rules between itemsets. Usually, an itemset is evaluated based on how close to each other its items occur (cohesion), and how often the itemset itself occurs (frequency).

Recently, we proposed to combine cohesion and frequency into a single measure [2], thereby guaranteeing that if we encounter an item from an itemset identified as interesting, there is a high probability that the rest of the itemset can be found nearby. The proposed algorithm suffered from long runtimes, despite the efficient pruning of candidates. We now propose relaxing the pruning function, but making it much easier to compute. As a result, we prune less, but the algorithm runs much faster.

We further propose to introduce the concept of association rules into this setting. We wish to find itemsets that imply the occurrence of other itemsets nearby. We present an efficient algorithm to mine such rules, taking advantage of two factors that lead to its efficiency — we can mine itemsets and rules in parallel, and we only need to compute the confidence of a simple class of rules, and use them to evaluate all other rules.

2 Related Work

The concept of finding patterns in sequences was first described by Mannila et al. [7]. Patterns are described as episodes, and can be parallel, where the order in which events occur is irrelevant, serial, where events occur in a particular order, or a combination of the two. We limit ourselves to parallel episodes containing no more than one occurrence of any single event type — in other words, itemsets. In this setting, Mannila et al. propose the Winepi method to detect frequent itemsets. The method slides a window of fixed length over the sequence, and each window containing the itemset counts towards its total frequency, which is defined as the proportion of all windows that contain it. The confidence of an association rule $X \Rightarrow Y$, denoted $c(X \Rightarrow Y)$, is defined as the ratio of the frequency of $X \cup Y$ and the frequency of X. Once the frequent itemsets have been found, rules between them are generated in the traditional manner [1].

Mannila et al. propose another method, Minepi, to search for frequent itemsets based on their minimal occurrences [7]. Here, however, association rules are of the form $X[win_1] \Rightarrow Y[win_2]$, meaning that if itemset X has a minimal occurrence in a window W_1 of size win_1, then $X \cup Y$ has a minimal occurrence in a window W_2 of size win_2 that fully contains W_1. As we plan to develop rules that tell us how likely we are to find some items nearby, we do not wish to use any fixed window lengths, so these are precisely the sort of rules we wish to avoid.

Generating association rules based on a maximum gap constraint, as defined by Garriga [4], was done by Méger and Rigotti [8], but only for serial episodes X and Y, where X is a prefix of Y. Most other related work has been based on the Winepi definitions, and only attempted to find the same rules (or a representative subset) more efficiently [3, 5].

Consider sequence $s = cghefababcidjcgdlcmd$, that will be used as a running example throughout the paper. Assume that the time stamps associated with the items are integers from 1 to 20. This short sequence is enough to demonstrate the unintuitiveness of the Winepi method for evaluating association rules. We see that for each occurrence of an a, there is a b right next to it. Similarly, for each g, there is a c right next to it. Logically, association rules $a \Rightarrow b$ and $g \Rightarrow c$ should be equally confident (to be precise, their confidence should be equal to 1). Winepi, with a window size of 2 (the number of possible windows is thus 21, as the first window starts one time stamp before the sequence, and the last one ends one time stamp after the sequence), results in $fr(a) = \frac{4}{21}$, $fr(ab) = \frac{3}{21}$, $fr(g) = \frac{4}{21}$, $fr(cg) = \frac{2}{21}$. and therefore $c(a \Rightarrow b) = 0.75$, $c(g \Rightarrow c) = 0.5$. A larger window always gives similar results, namely $c(g \Rightarrow c) < c(a \Rightarrow b) < 1$.

Due to space restrictions, we only present related work on association rules. A more extensive discussion of related work on discovering patterns in sequences can be found in [2] and [6].

3 Problem Setting

As our work is based on an earlier work [2], we now reproduce some of the necessary definitions and notations that we will use here. An event is a pair (i, t), consisting of an item and a time stamp, where $i \in I$, the set of all possible items, and $t \in \mathbb{N}$. Two items can never occur at the same time. We denote a sequence of events by $\mathcal{S}$. For an itemset X, the set of all occurrences of its items is denoted by $N(X) = \{t \mid (i,t) \in \mathcal{S} \text{ and } i \in X\}$. The coverage of X is defined as the probability of encountering an item from X in the sequence, and denoted

$$P(X) = \frac{|N(X)|}{|\mathcal{S}|}.$$

The length of the shortest interval containing itemset X for each time stamp in $N(X)$ is computed as

$$W(X,\, t) = \min\{t_2 - t_1 + 1 \mid t_1 \leq t \leq t_2 \text{ and } \forall i \in X, \exists (i, t') \in \mathcal{S},\, t_1 \leq t' \leq t_2\}.$$

The average length of such shortest intervals is expressed as

$$\overline{W}(X) = \frac{\sum_{t \in N(X)} W(X,\, t)}{|N(X)|}.$$

The cohesion of X is defined as the ratio of the itemset size and the average length of the shortest intervals that contain it, and denoted

$$C(X) = \frac{|X|}{\overline{W}(X)}.$$

Finally, the interestingness of an itemset X is defined as $I(X) = C(X)P(X)$. Given a user defined threshold *min_int*, X is considered interesting if $I(X)$ exceeds *min_int*. An optional parameter, *max_size*, can be used to limit the output only to itemsets with a size smaller or equal to *max_size*.

We are now ready to define the concept of association rules in this setting. The aim is to generate rules of the form *if X occurs, Y occurs nearby*, where $X \cap Y = \emptyset$ and $X \cup Y$ is an interesting itemset. We denote such a rule by $X \Rightarrow Y$, and we call X the body of the rule and Y the head of the rule. Clearly, the closer Y occurs to X on average, the higher the value of the rule. In other words, to compute the confidence of the rule, we must now use the average length of minimal windows containing $X \cup Y$, but only from the point of view of items making up itemset X. We therefore define this new average as

$$\overline{W}(X, Y) = \frac{\sum_{t \in N(X)} W(X \cup Y,\, t)}{|N(X)|}.$$

The confidence of a rule can now be defined as

$$c(X \Rightarrow Y) = \frac{|X \cup Y|}{\overline{W}(X, Y)}.$$

A rule $X \Rightarrow Y$ is considered confident if its confidence exceeds a given threshold, *min_conf*.

We now return to our running example. Looking at itemset cd, we see that the occurrence of a c at time stamp 1 will reduce the value of rule $c \Rightarrow d$, but not of rule $d \Rightarrow c$. Indeed, we see that $W(cd, 1) = 12$, and the minimal window containing cd for the other three occurrences of c is always of size 3. Therefore, $\overline{W}(c, d) = \frac{21}{4} = 5.25$, and $c(c \Rightarrow d) = \frac{2}{5.25} = 0.38$. Meanwhile, the minimal window containing cd for all occurrences of d is always of size 3. It follows that $\overline{W}(d, c) = \frac{9}{3} = 3$ and $c(d \Rightarrow c) = \frac{2}{3} = 0.67$. We can conclude that while an occurrence of a c does not highly imply finding a d nearby, when we encounter a d we can be reasonably certain that a c will be found nearby. We also note that, according to our definitions, $c(a \Rightarrow b) = 1$ and $c(g \Rightarrow c) = 1$, as desired.

4 Improved Interesting Itemsets Algorithm

The algorithm proposed in [2] and given in Algorithm 1, finds interesting itemsets as defined in Section 3 by going through the search space (a tree) in a depth-first manner, pruning whenever possible. The first call to the algorithm is made with X empty, and Y equal to the set of all items.

Algorithm 1 INIT($\langle X, Y \rangle$) finds interesting itemsets

```
if $UBI(\langle X, Y \rangle) \geq min\_int$ and $size(X) \leq max\_size$ then
  if $Y = \emptyset$ then
    output $X$
  else
    Choose $a$ in $Y$
    INIT($\langle X \cup \{a\}, Y \setminus \{a\} \rangle$)
    INIT($\langle X, Y \setminus \{a\} \rangle$)
  end if
end if
```

The algorithm uses the UBI pruning function, that returns an upper bound of the interestingness of all itemsets Z such that $X \subseteq Z \subseteq X \cup Y$. If this upper bound is lower than the chosen min_int, the subtree rooted at $\langle X, Y \rangle$ can be pruned. The UBI function is defined as

$$UBI(\langle X, Y \rangle) = \frac{|N(X \cup Y)|^2 \times |X \cup Y|}{\sum_{t \in N(X)} W(X, t) \times |\mathcal{S}|}.$$

This pruning function prunes a large number of candidates, but the algorithm still suffers from long runtimes, due to the fact that each time a new itemset X is considered for pruning, $W(X, t)$ needs to be computed for almost each $t \in N(X)$. For large itemsets, this can imply multiple dataset scans just to decide if a single candidate node can be pruned.

We propose a new pruning function in an attempt to balance pruning a large number of candidates with the effort needed for pruning. As the main problem with the original function was computing the exact minimal windows for each candidate, we aim to estimate the length of these windows using a much simpler computation. To do this, we first compute the exact sum of the window lengths for each pair of items, and we then use these sums to come up with a lower bound of the sum of the window lengths for all other candidate nodes.

We first note that

$$\sum_{t \in N(X)} W(X,\, t) = \sum_{x \in X} \sum_{t \in N(\{x\})} W(X,\, t).$$

We then note that each window around an item $x \in X$ must be at least as large as the largest such window containing the same item x and any other item $y \in X$. It follows that

$$\sum_{t \in N(\{x\})} W(X,\, t) \geq \sum_{t \in N(\{x\})} \max_{y \in X \setminus \{x\}}(W(xy,\, t)).$$

Naturally, it also holds that

$$\sum_{t \in N(\{x\})} W(X,\, t) \geq \max_{y \in X \setminus \{x\}}(\sum_{t \in N(\{x\})} W(xy,\, t)).$$

To simplify our notation, from now on we will denote $\sum_{t \in N(\{x\})} W(xy,\, t)$ by $s(x, y)$. Finally, we see that

$$\sum_{t \in N(X)} W(X,\, t) \geq \sum_{x \in X} \max_{y \in X \setminus \{x\}}(s(x, y)),$$

giving us a lower bound for the sum of windows containing X around all occurrences of items of X. This gives us a new pruning function:

$$NUBI(\langle X, Y \rangle) = \frac{|N(X \cup Y)|^2 \times |X \cup Y|}{\sum_{x \in X} \max_{y \in X \setminus \{x\}}(s(x,y)) \times |\mathcal{S}|}.$$

This new pruning function is easily evaluated, as all it requires is that we store $s(x, y)$, the sum of minimal windows containing x and y over all occurrences of x, for each pair of items (x, y), so we can look them up when necessary.

The exact windows will still have to be computed for the leaves of the search tree that have not been pruned, but this is unavoidable. Even for the leaves, it pays off to first check the upper bound, and then, only if the upper bound exceeds the threshold, compute the exact interestingness. The new algorithm uses $NUBI$ instead of UBI, and is the same as the one given in Algorithm 1, with

if $I(X) \geq min\_int$ **then output** X

replacing line 3.

Let us now return to our running example, and examine what happens if we encounter node $\langle \{a, b, c\}, \{d, e\} \rangle$ in the search tree. We denote $X = \{a, b, c\}$ and $Y = \{d, e\}$. With the original pruning technique, we would need to compute the exact minimal windows containing X for each occurrence of a, b or c. After a fair amount of work scanning the dataset many times, in all necessary directions, we would come up with the following: $\sum_{t \in N(X)} W(X,\, t) = 7 + 5 + 4 + 3 + 3 + 3 + 7 + 11 = 43$. The value of the UBI pruning function is therefore:

$$\frac{|N(X \cup Y)|^2 \times |X \cup Y|}{\sum_{t \in N(X)} W(X,\, t) \times |\mathcal{S}|} = \frac{144 \times 5}{43 \times 20} = 0.84.$$

Using the new technique, we would first compute $s(x, y)$ for all pairs (x, y). The relevant pairs are $s(a, b) = 4$, $s(a, c) = 8$, $s(b, a) = 4$, $s(b, c) = 6$, $s(c, a) = 27$, $s(c, b) = 25$. We can now compute the minimal possible sum of windows for each item, giving us

$$\max_{y\in X\setminus\{a\}}(s(a,y)) = 8,\ \max_{y\in X\setminus\{b\}}(s(b,y)) = 6,\ \max_{y\in X\setminus\{c\}}(s(c,y)) = 27$$

resulting in a sum of $\sum_{x\in X}\max_{y\in X\setminus\{x\}}(s(x,y)) = 41$. The value of the NUBI pruning function is therefore

$$\frac{|N(X\cup Y)|^2\times|X\cup Y|}{\sum_{x\in X}\max_{y\in X\setminus\{x\}}(s(x,y))\times|\mathcal{S}|} = 0.88$$

We see that by simply looking up a few precomputed values instead of scanning the dataset a number of times, we get a very good estimate of the sum of the window lengths.

5 Association Rules Algorithm

Unlike the traditional approaches, which need all the frequent itemsets to be generated before the generation of association rules can begin, we are able to generate rules in parallel with the interesting itemsets. When finding an interesting itemset X, we compute the sum of all minimal windows $W(X,\, t)$ for each $x \in X$ apart, before adding them up into the overall sum needed to compute $I(X)$. With these sums still in memory, we can easily compute the confidence of all association rules of the form $x \Rightarrow X\setminus\{x\}$, with $x \in X$, that can be generated from itemset X. In practice, it is sufficient to limit our computations to rules of precisely this form (i.e., rules where the body consists of a single item). To compute the confidence of all other rules, we first note that

$$\sum_{t\in N(X)} W(X\cup Y,\, t) = \sum_{x\in X}\sum_{t\in N(\{x\})} W(X\cup Y,\, t).$$

A trivial mathematical property tells us that

$$\sum_{t\in N(\{x\})} W(X\cup Y,\, t) = \overline{W}(x, Y\cup X\setminus\{x\})|N(x)|.$$

As a result, we can conclude that

$$\overline{W}(X,Y) = \frac{\sum_{x\in X}\overline{W}(x,Y\cup X\setminus\{x\})|N(x)|}{|N(X)|},$$

which in turn implies that

$$c(X\Rightarrow Y) = \frac{|X\cup Y||N(X)|}{\sum_{x\in X}\overline{W}(x,Y\cup X\setminus\{x\})|N(x)|}.$$

Meanwhile, we can derive that

$$c(x\Rightarrow Y\cup X\setminus\{x\}) = \frac{|X\cup Y|}{\overline{W}(x,Y\cup X\setminus\{x\})},$$

and it follows that

$$c(X\Rightarrow Y) = \frac{|N(X)|}{\sum_{x\in X}\frac{|N(x)|}{c(x\Rightarrow Y\cup X\setminus\{x\})}}. \qquad (1)$$

As a result, once we have evaluated all the rules of the form $x \Rightarrow X\setminus\{x\}$, with $x \in X$, we can then evaluate all other rules $Y \Rightarrow X\setminus Y$, with $Y \subset X$, without

Algorithm 2 AR($\langle X, Y \rangle$) finds interesting itemsets and confident association rules within them

```
if $NUBI(\langle X, Y \rangle) \geq min\_int$ and $size(X) \leq max\_size$ then
  if $Y = \emptyset$ then
    if $I(X) \geq min\_int$ then
      output $X$
      for all $x \in X$ do
        compute and store $c(x \Rightarrow X \setminus \{x\})$
        if $c(x \Rightarrow X \setminus \{x\}) \geq min\_conf$ then output $x \Rightarrow X \setminus \{x\}$
      end for
      for all $Y \subset X$ with $|Y| \geq 2$ do
        if $c(Y \Rightarrow X \setminus Y) \geq min\_conf$ then output $Y \Rightarrow X \setminus Y$
      end for
    end if
  else
    Choose $a$ in $Y$
    AR($\langle X \cup \{a\}, Y \setminus \{a\} \rangle$)
    AR($\langle X, Y \setminus \{a\} \rangle$)
  end if
end if
```

further dataset scans. The algorithm for finding both interesting itemsets and confident association rules is given in Algorithm 2.

Looking back at our running example, let us compute the confidence of rule $ab \Rightarrow c$. First we compute $c(a \Rightarrow bc) = \frac{3}{4} = 0.75$ and $c(b \Rightarrow ac) = \frac{3}{3.5} = 0.86$. From Eq. 1, it follows that $c(ab \Rightarrow c) = \frac{4}{\frac{2}{0.75}+\frac{2}{0.86}} = \frac{4}{5} = 0.8$. It is easy to check that this corresponds to the value as defined in Section 3.

6 Experiments

In our experiments, we aim to show three things:

1. our algorithm for finding interesting itemsets works faster than the one given in [2] and can handle much longer sequences.
2. our algorithm for finding association rules gives meaningful results, without generating spurious output.
3. our algorithm for finding association rules runs very efficiently, even with a confidence threshold set to 0.

To do this, we designed a dataset that allowed us to demonstrate all three claims. To generate a sequence in which some association rules would certainly stand out, we used the Markov chain model given in Table 1. Our dataset consisted of items a, b, c, and 22 other items, randomly distributed whenever the transition table led us to item x. We fine tuned the probabilities in such a way that all items apart from c had approximately the same frequency, while c appeared approximately twice as often. The high probability of transitions from a

to b and c, and from b to a and c should result in rules such as $a \Rightarrow c$ and $b \Rightarrow c$ having a high confidence. However, given that c appears more often, sometimes without an a or a b nearby, we would expect rules such as $c \Rightarrow a$ and $c \Rightarrow b$ to be ranked lower. Later on we show that our algorithm gave the expected results for all these cases.

	a	b	c	x
a	0	0.45	0.45	0.1
b	0.45	0	0.45	0.1
c	0	0	0.1	0.9
x	0.025	0.025	0.05	0.9

Table 1. A transition matrix defining a Markov model

First, though, let us examine our first claim. We used our Markov model to generate ten sequences of 10 000 items and ran the two algorithms on each sequence, varying *min_int*, at first choosing *max_size* of 4. Figures 1(a) and 1(c) show the average runtimes and number of evaluated candidate nodes for each algorithm, as well as the actual number of identified interesting itemsets. While our algorithm pruned less (Figure 1(c)), it ran much faster, most importantly at the thresholds where the most interesting itemsets are found (see Figure 1(b) for a zoomed-in version). Naturally, if $min\_int = 0$, the algorithms take equally long, as all itemsets are identified as interesting, and their exact interestingness must be computed. In short, we see that the runtime of the original algorithm is proportional to the number of candidates, while the runtime of our new algorithm is proportional to the number of interesting itemsets.

To support the second claim, we ran our association rules algorithm with both *min_int* and *min_conf* equal to 0. We set *max_size* to 4. We now simply had to check which rules had the highest confidence. In repeated experiments, with various 500 000 items long datasets, the results were very consistent. The most interesting rules were $a \Rightarrow c$ and $b \Rightarrow c$, with a confidence of 0.52. Then followed rules $a \Rightarrow bc$, $b \Rightarrow ac$ and $ab \Rightarrow c$, with a confidence of 0.34. Rules $a \Rightarrow b$ and $b \Rightarrow a$ had a confidence of 0.29, while rules $ac \Rightarrow b$ and $bc \Rightarrow a$ had a confidence of 0.2. Rule $c \Rightarrow ab$ had a confidence of around 0.17, while rules not involving a, b or c had a confidence between 0.13 and 0.17. We can safely conclude that our algorithm gave the expected results.

To confirm this claim, we ran our algorithm on three text datasets: *English*[1], *Italian*[2] and *Dutch*[3]. In each text, we considered the letters of the alphabet and the space between words (denoted ␣) as items, ignoring all other symbols. In all three languages, rule $q \Rightarrow u$ had a confidence of 1, as q is almost always followed by u in all these languages. In all three languages, there followed a number of rules with ␣ in the head, but the body varied. In Italian, a space is often found near f, as f is mostly found at the beginning of Italian words, while the same is

[1] a selection from http://www.gutenberg.org/files/1999/1999.txt
[2] a selection from http://www.gutenberg.org/dirs/etext04/7clel10.txt
[3] a selection from http://www.gutenberg.org/files/18066/18066-8.txt

true for j in English. Rules involving two letters revealed some patterns inherent in these languages. For example, rule $h \Rightarrow c$ was ranked high in Italian (where h appears very rarely without a c in front of it), while rule $j \Rightarrow i$ scored very high in Dutch (where j is often preceded by an i). A summary of the results is given in Table 2.

	English	*Italian*	*Dutch*
two letters	$c(q \Rightarrow u) = 1$	$c(q \Rightarrow u) = 1$	$c(q \Rightarrow u) = 1$
	$c(z \Rightarrow i) = 0.61$	$c(h \Rightarrow c) = 0.75$	$c(j \Rightarrow i) = 0.75$
	$c(v \Rightarrow e) = 0.58$	$c(h \Rightarrow e) = 0.51$	$c(d \Rightarrow e) = 0.61$
	$c(x \Rightarrow e) = 0.53$	$c(z \Rightarrow a) = 0.5$	$c(g \Rightarrow e) = 0.57$
three letters	$c(q \Rightarrow eu) = 0.63$	$c(q \Rightarrow eu) = 0.6$	$c(q \Rightarrow iu) = 1$[5]
	$c(z \Rightarrow ai) = 0.52$	$c(h \Rightarrow ce) = 0.57$	$c(j \Rightarrow ei) = 0.46$
	$c(q \Rightarrow nu) = 0.4$	$c(q \Rightarrow au) = 0.52$	$c(g \Rightarrow en) = 0.44$
with $\sqcup$	$c(y \Rightarrow \sqcup) = 0.85$	$c(q \Rightarrow \sqcup) = 0.84$	$c(z \Rightarrow \sqcup) = 0.78$
	$c(w \Rightarrow \sqcup) = 0.84$	$c(f \Rightarrow \sqcup) = 0.7$	$c(w \Rightarrow \sqcup) = 0.74$
	$c(j \Rightarrow \sqcup) = 0.83$	$c(a \Rightarrow \sqcup) = 0.7$	$c(v \Rightarrow \sqcup) = 0.73$
	...	...	...
	$c(\sqcup \Rightarrow e) = 0.35$	$c(\sqcup \Rightarrow a) = 0.39$	$c(\sqcup \Rightarrow n) = 0.35$

Table 2. Some interesting rules found in three different languages

To prove our third claim, we ran the *AR* algorithm on ten Markov chain datasets (each 10 000 items long), using *min_int* of 0.025 and *max_size* of 4, each time varying *min_conf*. The average runtimes and number of found association rules are shown in Figure 1(d). We can see that the exponential growth in the number of generated rules had virtually no effect on the runtime of the algorithm.

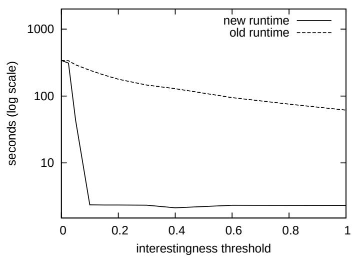
7 Conclusion

In this paper we presented a new way of identifying association rules in sequences. We base ourselves on interesting itemsets and look for association rules within them. When we encounter a part of an itemset in the sequence, our measure of the rule's confidence tells us how likely we are to find the rest of the itemset nearby.

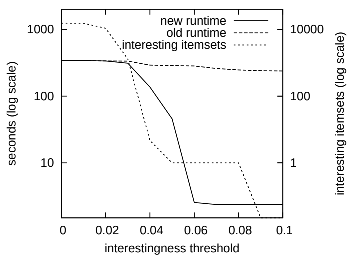
On our way to discovering association rules, we found a way to improve the runtime of the algorithm that finds the interesting itemsets, too. By relaxing the upper bound of an itemset's interestingness, we actually prune fewer candidates, but our new algorithm runs much faster than the old one. To be precise, the runtime of the new algorithm is proportional to the number of identified itemsets, while the runtime of the old algorithm was proportional to the number of evaluated nodes in the search tree. Due to being able to generate association rules while mining interesting itemsets, the cost of finding confident association rules is negligible.

Experiments demonstrated the validity of our central claims — that our algorithm for finding interesting itemsets runs much faster than the original one, particularly on long datasets, and that our algorithm for finding association rules gives meaningful results very efficiently.

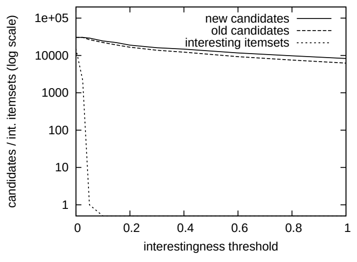
[5] This is due to a short dataset, rather than an actual rule in the Dutch language.



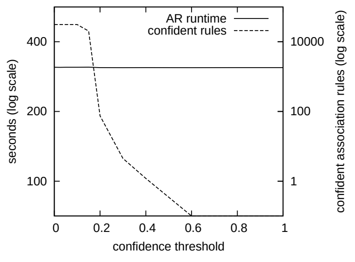
(a)



(b)



(c)



(d)

Fig. 1. (a) Runtime comparison of the two itemset algorithms with *max_size* set to 4. (b) Zoomed-in version of Figure 1(a). (c) Pruning comparison with *max_size* set to 4. (d) Runtime of the *AR* algorithm with a varying confidence threshold.

References

1. R. Agrawal, T. Imielinski and A. Swami, *Mining Association Rules between Sets of Items in Large Databases*, Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 207-216, 1993
2. B. Cule, B. Goethals and C. Robardet, *A new constraint for mining sets in sequences*, Proc. SIAM Int. Conf. on Data Mining (SDM), pp. 317-328, 2009
3. G. Das, K-I. Lin, H. Mannila, G. Renganathan and P. Smyth, *Rule discovery from time series*, Proc. Int. Conf. on Knowledge Discovery and Data Mining (KDD), pp. 16-22, 1998
4. G. C. Garriga, *Discovering Unbounded Episodes in Sequential Data*, Proc. 7th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD), pp. 83–94, 2003
5. S. K. Harms, J. Saquer and T. Tadesse, *Discovering Representative Episodal Association Rules from Event Sequences Using Frequent Closed Episode Sets and Event Constraints*, Proc. IEEE Int. Conf. on Data Mining (ICDM), pp. 603-606, 2001
6. S. Laxman and P. S. Sastry, *A survey of temporal data mining*, SADHANA, Academy Proceedings in Engineering Sciences, volume 31, part 2, pp. 173–198, 2006
7. H. Mannila, H. Toivonen and A. I. Verkamo, *Discovery of Frequent Episodes in Event Sequences*, Data Mining and Knowledge Discovery, volume 1(3), pp. 259–289, 1997
8. N. Méger and C. Rigotti, *Constraint-Based Mining of Episode Rules and Optimal Window Sizes*, Proc. 8th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD), pp. 313–324, 2004