RP-growth: Top-k Mining of Relevant Patterns with Minimum Support Raising[*]

Yoshitaka Kameya and Taisuke Sato[†]

Abstract

One practical inconvenience in frequent pattern mining is that it often yields a flood of common or uninformative patterns, and thus we should carefully adjust the minimum support. To alleviate this inconvenience, based on FP-growth, this paper proposes RP-growth, an efficient algorithm for top-k mining of discriminative patterns which are highly relevant to the class of interest. RP-growth conducts a branch-and-bound search using anti-monotonic upper bounds of the relevance scores such as F-score and χ^2, and the pruning in branch-and-bound search is successfully translated to minimum support raising, a standard, easy-to-implement pruning strategy for top-k mining. Furthermore, by introducing the notion called weakness and an additional, aggressive pruning strategy based on weakness, RP-growth efficiently finds k patterns of wide variety and high relevance to the class of interest. Experimental results on text classification exhibit the efficiency and the usefulness of RP-growth.

1 Introduction

Frequent pattern mining is now one of core techniques in knowledge discovery. On the flip side, there is a practical inconvenience in frequent pattern mining that it often yields a flood of common or uninformative patterns, and thus we should carefully adjust the threshold called the minimum support to have a manageable number of frequent patterns. One improvement against this inconvenience is *top-k mining* [11, 21], where k is the number of patterns to find, and the minimum support is automatically adjusted following k. Top-k mining has a clear merit that the parameter k is less sensitive to the statistics on the dataset and so is easier for human users to specify. Another direction is to find more informative patterns by *discriminative pattern mining* from the dataset where a class label is supplied for each transaction. Discriminative pattern mining extracts the patterns that distinguish the transactions in one class from those in the other(s), and has been studied under the names of subgroup discovery [8, 22], emerging pattern mining [4, 5, 20], contrast set mining [1], supervised descriptive rule discovery [15], cluster grouping [24], etc.

The purpose of this paper is to propose RP-growth, an efficient top-k mining algorithm for finding discriminative patterns called *relevant patterns*, based on FP-growth [10]. The relevance between a pattern $\boldsymbol{x}$ and the class c of interest is often quantified by a relevance score $R_c(\boldsymbol{x})$ (e.g. χ^2, F-score and so on), and the main difficulty in discriminative pattern mining is that most of popular relevance scores do not satisfy the anti-monotonicity w.r.t. set-inclusion among patterns, and hence a simple pruning based on the minimum support is not available. To get over the lack of anti-monotonicity, as is occasionally done [19, 21, 22, 24], RP-growth performs a branch-and-bound search for relevant patterns, in which we compute an *anti-monotonic upper bound* $\overline{R}_c(\boldsymbol{x})$ of the relevance score for each node (corresponding to a pattern $\boldsymbol{x}$) in an enumeration tree, and we prune the subtree below the node, according to the computed upper bound. One technical contribution of this paper is to show how the pruning strategy in this branch-and-bound search is translated into *minimum support raising* [11], a standard, easy-to-implement pruning strategy for top-k frequent pattern mining. Unlike previous approaches, this translation is applicable to non-convex relevance scores such as F-score, and is also possible in many of the existing mining algorithms for itemsets, sequences and trees. Furthermore, for filtering out the patterns that are still redundant and uninformative, we introduce the notion called *weakness* between patterns, and another contribution is a finding that the search tree used in FP-growth enables aggressive pruning based on weakness. As a result, RP-growth efficiently extracts k patterns of wide variety and high relevance to the class of interest, and they would lead to some crucial knowledge about the dataset.

The rest of this paper is outlined as follows. First, in Section 2, we explain the background of this paper. Section 3 then gives the details of RP-growth, our proposed method. In Section 4, we report the experimental results on text classification. Finally, Section 5 describes the related work, and Section 6 concludes this paper.

[*] This work is supported in part by Grant-in-Aid for Scientific Research (No. 20240016) from MEXT of Japan.

[†] Graduate School of Information Science and Engineering, Tokyo Institute of Technology, {kameya,sato}@mi.cs.titech.ac.jp

2 Background

2.1 Preliminaries. First of all, let us introduce some notations. We are given a dataset $\mathcal{D} = \{t_1, t_2, \ldots, t_N\}$ of size N, where t_i is a transaction, a set of items. The set of all items appearing in $\mathcal{D}$ is denoted by $\mathcal{X}$. Also we consider that each transaction belongs to one of predefined classes $\mathcal{C}$, and let c_i be the class of transaction t_i. A pattern is a subset of $\mathcal{X}$, and let $\mathcal{P}$ be the set of all possible patterns. We introduce some canonical order $\prec$ among items, and always place the items in a transaction/pattern following $\prec$. Transaction/patterns are also ordered in a lexicographical way based on $\prec$. Depending on the context, we interchangeably denote a pattern as a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, as a set $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$, or as a conjunction $\boldsymbol{x} = (x_1 \wedge x_2 \wedge \ldots \wedge x_n)$, and an item is regarded as one of these collections with a single element.

In addition, the probabilities treated in this paper are all empirical ones, i.e. they are computed from the statistics on the dataset $\mathcal{D}$. For example, a joint probability $p(c, \boldsymbol{x})$ is obtained as $N(c, \boldsymbol{x})/N$, where $N(c, \boldsymbol{x}) = \#\{i \mid c_i = c, \boldsymbol{x} \subseteq t_i, 1 \leq i \leq N\}$ and $\#S$ indicates the cardinality of a set S. Also we use a symbol $\neg$ for negation. For example, we have $p(c, \neg\boldsymbol{x}) = N(c, \neg\boldsymbol{x})/N = \#\{i \mid c_i = c, \boldsymbol{x} \not\subseteq t_i, 1 \leq i \leq N\}/N$, $p(\neg c, \boldsymbol{x}) = N(\neg c, \boldsymbol{x})/N = \#\{i \mid c_i \neq c, \boldsymbol{x} \subseteq t_i, 1 \leq i \leq N\}/N$, and so on. Using these joint probabilities, marginal probabilities and conditional probabilities are computed, e.g. we obtain $p(\boldsymbol{x}) = p(c, \boldsymbol{x}) + p(\neg c, \boldsymbol{x})$, $p(c) = p(c, \boldsymbol{x}) + p(c, \neg\boldsymbol{x})$ or $p(c \mid \boldsymbol{x}) = p(c, \boldsymbol{x})/p(\boldsymbol{x})$. Throughout the paper, we only consider practical cases where $0 < p(c) < 1$ and $0 < p(\boldsymbol{x}) < 1$ hold.

2.2 Relevant patterns. As mentioned before, we seek for k patterns that are relevant to the class c of interest. For that, we first adopt a relevance score R_c, which is defined as a function from $\mathcal{P}$ to $\mathbb{R}$, the set of real numbers. The relevance measured by $R_c(\boldsymbol{x})$ can also be regarded as interestingness of a class association rule $\boldsymbol{x} \Rightarrow c$, where $\boldsymbol{x}$ is a pattern. Many relevance scores have been proposed in the literature [9, 15], and from now on, we briefly explain a dozen of well-known ones, which are grouped into several categories:

- *Confidence* $p(c \mid \boldsymbol{x})$ is used in class association rule mining [16], and we call $p(\boldsymbol{x} \mid c)$ (resp. $p(\boldsymbol{x} \mid \neg c)$) the *positive* (resp. *negative*) *support* of $\boldsymbol{x}$ for class c. For brevity, 'support' means positive support unless explicitly noted. *Growth rate* $\mathrm{GR}_c(\boldsymbol{x}) = p(\boldsymbol{x} \mid c)/p(\boldsymbol{x} \mid \neg c)$ explicitly states that the transactions satisfying $\boldsymbol{x}$ are common in class c and uncommon in the class(es) other than c, and is mainly used to find emerging patterns [4]. *Pointwise mutual information* $\mathrm{PMI}_c(\boldsymbol{x}) = \log p(c, \boldsymbol{x}) - \log\{p(c)p(\boldsymbol{x})\}$ is adopted in text analysis [17], and its non-logarithmic version $p(c, \boldsymbol{x})/(p(c)p(\boldsymbol{x}))$ is called *lift*. Using the results by Kralj Novak et al. [15], it is shown that the relevance scores in this category except support give the same ranking over possible patterns $\mathcal{P}$ (and consequently, the same top-k patterns).
- *Leverage* $\mathrm{L}_c(\boldsymbol{x}) = p(c, \boldsymbol{x}) - p(c)p(\boldsymbol{x})$ is often used for finding interesting association rules [21]. $\mathrm{L}_c(\boldsymbol{x})$ is equivalent to the *weighted relative accuracy*, a score used in subgroup discovery [22]. A related score $\mathrm{SD}_c(\boldsymbol{x}) = p(\boldsymbol{x} \mid c) - p(\boldsymbol{x} \mid \neg c)$, often called *support difference*, is adopted in contrast set mining [1].[1] These scores also give the same ranking over $\mathcal{P}$ [15].
- *Precision* and *recall* are well-known measures in information retrieval [17] and evaluation of classifiers [12]. $p(c \mid \boldsymbol{x})$ and $p(\boldsymbol{x} \mid c)$ is also seen as precision and recall, respectively, of a pattern $\boldsymbol{x}$ for class c [9]. To balance their opposite behaviors, we often use the harmonic mean $\mathrm{F}_c(\boldsymbol{x}) = 2p(c \mid \boldsymbol{x})p(\boldsymbol{x} \mid c)/(p(c \mid \boldsymbol{x}) + p(\boldsymbol{x} \mid c)) = 2p(c, \boldsymbol{x})/(p(c) + p(\boldsymbol{x}))$ and call it the *F-score* or *Dice's index*. *Jaccard's index* $\mathrm{J}_c(\boldsymbol{x}) = p(c, \boldsymbol{x})/(p(c) + p(\boldsymbol{x}) - p(c, \boldsymbol{x}))$ can be rewritten as $\mathrm{F}_c(\boldsymbol{x})/(2 - \mathrm{F}_c(\boldsymbol{x}))$ and hence F_c and J_c give the same ranking over $\mathcal{P}$.
- χ^2 is a traditional measure indicating the correlation between two random variables. Noting that $\{c, \neg c\}$ and $\{\boldsymbol{x}, \neg\boldsymbol{x}\}$ are realizations of random variables C and X, respectively, the χ^2 value between C and X is regarded as a relevance score $\chi^2_c(\boldsymbol{x}) = \sum_{c' \in \{c, \neg c\}, \boldsymbol{x}' \in \{\boldsymbol{x}, \neg\boldsymbol{x}\}} \tau(c', \boldsymbol{x}')$ where

$$\tau(c', \boldsymbol{x}') = N\frac{(p(c', \boldsymbol{x}') - p(c')p(\boldsymbol{x}'))^2}{p(c')p(\boldsymbol{x}')}.$$

  Information gain (IG) is also a popular correlation measure between C and X. In [18, 19], both χ^2 and IG are shown to be convex w.r.t. the pair of $p(c, \boldsymbol{x})$ and $p(\boldsymbol{x})$, and have been adopted in branch-and-bound search [19, 24].

2.3 Minimum support raising. In this subsection, we describe minimum support raising [11], a standard, easy-to-implement pruning strategy in top-k pattern mining. First, let us consider an *enumeration tree*, illustrated in Fig. 1.[2] One may find in Fig. 1 that the parent of each node is its immediate prefix, and

[1] For simplicity, we modified the original definition $|p(\boldsymbol{x} \mid c) - p(\boldsymbol{x} \mid \neg c)|$ [1], since for our purpose, it seems reasonable to have patterns $\boldsymbol{x}$ such that $p(\boldsymbol{x} \mid c) > p(\boldsymbol{x} \mid \neg c)$.

[2] We only consider enumeration trees which has a one-to-one map between the nodes in a tree and the possible patterns in $\mathcal{P}$. So in the sequel, we refer to a node by its corresponding pattern.

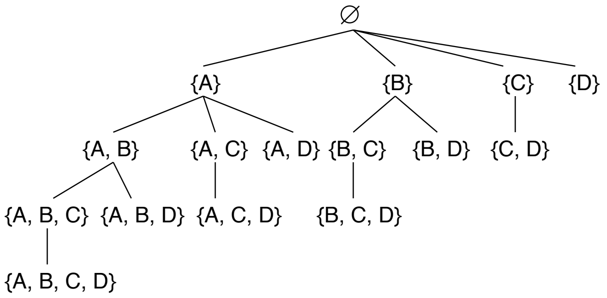

Figure 1: An example of a prefix enumeration tree.

such a tree is hereafter called a *prefix enumeration tree*. Although various styles on the traversal order of nodes are possible in an enumeration tree, in this paper, we focus on depth-first styles.

Now let us assume that, for a class c of interest, any pattern $\boldsymbol{x}$ should have a support $p(\boldsymbol{x} \mid c)$ no less than the minimum support $\sigma_{\min}$. Then, if $\boldsymbol{x}$ has the support $p(\boldsymbol{x} \mid c)$ smaller than $\sigma_{\min}$, we can safely prune the subtree rooted by $\boldsymbol{x}$, thanks to the anti-monotonicity of the support. Furthermore, consider the case of top-k *frequent* pattern mining. In top-k mining, we usually use an ordered list of candidate patterns, called the *candidate list* in this paper, and insert the patterns found in the middle of the search into the list, following the descending order of support. Suppose further that the candidate list already contains k patterns in the middle of the search. Then, letting $\boldsymbol{z}$ be the pattern with the k-th greatest support, it is obvious that any pattern $\boldsymbol{x}$ to be visited later should have the support no less than $p(\boldsymbol{z} \mid c)$, and so once observing $p(\boldsymbol{x} \mid c) < p(\boldsymbol{z} \mid c)$, we can prune the subtree rooted by $\boldsymbol{x}$. This operation is equivalent to raising the minimum support up to $p(\boldsymbol{z} \mid c)$ (i.e. $\sigma_{\min} := \max\{p(\boldsymbol{z} \mid c), \sigma_{\min}\}$) when $\boldsymbol{z}$ is found to be the pattern with the k-th greatest support at the time. The patterns with the $(k+1)$-th greatest or smaller support are usually removed automatically. Starting with a small value (typically $\sigma_{\min} := 1/N$ where N is the number of transactions), the minimum support is iteratively updated during the search. Section 3.1 describes how this minimum support raising is achieved in our case.

3 Top-k mining of relevant patterns

In this section, we explain the details of RP-growth, our proposed method. First of all, we define the task for top-k mining of relevant patterns.[3]

Definition 3.1. *Suppose a set $\mathcal{D}$ of transactions with classes, and the user-specified parameters k (≥ 1), $\sigma_{\min}$ (> 0), $\beta_{\min}$ (> 0) are given. Then, top-k mining of relevant patterns in $\mathcal{D}$ for class c is to find a set $\mathcal{P}^*$ of patterns ($\mathcal{P}^* \subseteq \mathcal{P}$) that satisfies:*

1. *$R_c(\boldsymbol{z}) > R_c(\boldsymbol{x}')$ for $\forall \boldsymbol{x}' \in \mathcal{P} \setminus \mathcal{P}^*$, where $\boldsymbol{z}$ is the pattern with the k-th greatest relevance score in $\mathcal{P}^*$,*
2. *$p(\boldsymbol{x} \mid c) \geq \sigma_{\min}$ for $\forall \boldsymbol{x} \in \mathcal{P}^*$,*
3. *$p(c \mid \boldsymbol{x}) \geq \beta_{\min}$ for $\forall \boldsymbol{x} \in \mathcal{P}^*$,*
4. *For $\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{P}^*$, $\boldsymbol{x}$ is not weaker than $\boldsymbol{x}'$, i.e. it does not hold that $\boldsymbol{x}' \subset \boldsymbol{x}$ and $R_c(\boldsymbol{x}) \leq R_c(\boldsymbol{x}')$.*

Here, k is the only mandatory user-specified parameter in this task. Typically we specify $\sigma_{\min} = 1/|\mathcal{D}|$ in the second condition, and in the third condition, we give $\beta_{\min} = p(c)$ or 0.5. The latter case indicates that each $\boldsymbol{x} \in \mathcal{P}^*$ works as a (weak) classifier, and we may specify a higher $\beta_{\min}$ for having more discriminative patterns. The last condition can be optional, and we will explore the notion called weakness in Section 3.3 and 3.4.

3.1 Branch-and-bound pruning translated into minimum support raising. As is mentioned before, branch-and-bound search is often adopted when the relevance score in use does not satisfy anti-monotonicity [19, 22, 24]. In a basic form of branch-and-bound search for top-k mining, for a class c of interest, we compute an anti-monotonic upper bound $\overline{R}_c(\boldsymbol{x})$ of the relevance score $R_c(\boldsymbol{x})$ of $\boldsymbol{x}$, and if it is found that $\overline{R}_c(\boldsymbol{x}) < R_c(\boldsymbol{z})$, where $\boldsymbol{z}$ is the pattern with the k-th greatest score, then we safely prune the subtree rooted by $\boldsymbol{x}$. This pruning exploits the anti-monotonicity of $\overline{R}_c$, which guarantees $R_c(\boldsymbol{x}') \leq \overline{R}_c(\boldsymbol{x}') \leq \overline{R}_c(\boldsymbol{x}) < R_c(\boldsymbol{z})$ for any super-pattern $\boldsymbol{x}'$ of $\boldsymbol{x}$.

In most of previous work, upper bounds are obtained by considering a common optimistic case. For example, in AprioriSMP [19], one upper bound of $\chi^2_c(\boldsymbol{x})$ is obtained by considering the case with $p(\boldsymbol{x}) = p(c, \boldsymbol{x})$ or $p(\neg c, \boldsymbol{x}) = 0$. Indeed, these conditions are equivalent to the conditions $p(c \mid \boldsymbol{x}) = 1$, $p(\boldsymbol{x} \mid \neg c) = 0$, $p(\neg \boldsymbol{x} \mid \neg c) = 1$, $p(\boldsymbol{x}) = p(c)p(\boldsymbol{x} \mid c)$, $p(\neg c, \neg \boldsymbol{x}) = p(\neg c)$, and so on. So a general heuristic for obtaining $\overline{R}_c(\boldsymbol{x})$ is to substitute simultaneously $p(c \mid \boldsymbol{x}) := 1$, $p(\boldsymbol{x} \mid \neg c) := 0$, $p(\neg \boldsymbol{x} \mid \neg c) := 1$, etc. into the definition of $R_c(\boldsymbol{x})$. Of course, after deriving $\overline{R}_c(\boldsymbol{x})$, we need to be sure that $\overline{R}_c(\boldsymbol{x})$ is indeed an upper bound of $R_c(\boldsymbol{x})$ and is anti-monotonic. For example, F-score is defined as $\mathrm{F}_c(\boldsymbol{x}) = 2p(c \mid \boldsymbol{x})p(\boldsymbol{x} \mid c)/(p(c \mid \boldsymbol{x}) + p(\boldsymbol{x} \mid c))$, so we obtain its upper bound by substituting $p(c \mid \boldsymbol{x}) := 1$ as:

$$\overline{\mathrm{F}}_c(\boldsymbol{x}) \quad = \quad \frac{2p(\boldsymbol{x} \mid c)}{1 + p(\boldsymbol{x} \mid c)}.$$

[3] A special treatment is needed for two exceptional cases in Def. 3.1. First, in the first condition, ties are appropriately broken depending on the application or at random. Second, if there is no $\mathcal{P}^*$ satisfying $|\mathcal{P}^*| \geq k$ and conditions 2–4, then we return the maximal set satisfying conditions 2–4.

Since F-score is the harmonic mean of $p(c \mid \boldsymbol{x})$ and $p(\boldsymbol{x} \mid c)$, which both range from 0 to 1, $\overline{\mathrm{F}}_c(\boldsymbol{x})$ is surely no less than $\mathrm{F}_c(\boldsymbol{x})$. Also for $0 \leq p(\boldsymbol{x} \mid c) \leq 1$, $\overline{\mathrm{F}}_c(\boldsymbol{x})$ is monotonically increasing according to $p(\boldsymbol{x} \mid c)$ and hence satisfy the anti-monotonicity w.r.t. set-inclusion among patterns. Similarly, the upper bounds of support difference and χ^2 are obtained as follows:[4]

$$\begin{aligned} \overline{\mathrm{SD}}_c(\boldsymbol{x}) &= p(\boldsymbol{x} \mid c) \\ \overline{\chi}^2_c(\boldsymbol{x}) &= N \frac{p(\boldsymbol{x} \mid c)p(\neg c)}{1 - p(c)p(\boldsymbol{x} \mid c)}. \end{aligned}$$

The above heuristic is applicable to various non-convex relevance scores, but for some scores whose upper bounds are always high, there would be no chance of pruning (e.g. growth rate $\mathrm{GR}_c(\boldsymbol{x}) = \infty$ and precision $p(c \mid \boldsymbol{x}) = 1$ when substituting $p(\boldsymbol{x} \mid \neg c) := 0$).

Here let us see how to translate branch-and-bound pruning into minimum support raising. As described before, $\boldsymbol{x}$ cannot stay in the final top-k patterns if $\overline{\mathrm{F}}_c(\boldsymbol{x}) < \mathrm{F}_c(\boldsymbol{z})$, where $\boldsymbol{z}$ is the pattern with the k-th greatest relevance score. Using $\overline{\mathrm{F}}_c(\boldsymbol{x})$ derived above, this condition is rewritten as:[5]

$$p(\boldsymbol{x} \mid c) < \frac{\mathrm{F}_c(\boldsymbol{z})}{2 - \mathrm{F}_c(\boldsymbol{z})}.$$

Then, noting that the right hand side works as a threshold for the support of $\boldsymbol{x}$, we have the following updating rule that realizes minimum support raising:

$$\sigma_{\min} \quad := \quad \max\left\{\frac{\mathrm{F}_c(\boldsymbol{z})}{2 - \mathrm{F}_c(\boldsymbol{z})}, \sigma_{\min}\right\}.$$

In a similar way, respectively for support difference and χ^2, we have the following updating rules:

$$\begin{aligned} \sigma_{\min} &:= \max\left\{\mathrm{SD}_c(\boldsymbol{z}), \sigma_{\min}\right\} \\ \sigma_{\min} &:= \max\left\{\frac{\chi^2_c(\boldsymbol{z})}{p(c)\chi^2_c(\boldsymbol{z}) + p(\neg c)N}, \sigma_{\min}\right\}. \end{aligned}$$

In what follows, we may use a generic form of these update rules: $\sigma_{\min} := \max\{U_c(\boldsymbol{z}), \sigma_{\min}\}$.

Here are a couple of crucial remarks on branch-and-bound search. First, we can have upper bounds of F-score and support difference with a simple operation, while AprioriSMP and the CG algorithm [24] assume the convexity of the relevance score. Second, we only need to change the updating rules for minimum support raising in existing implementations of top-k frequent pattern mining. Third and lastly, we can still benefit from the efficiency brought by FP-growth due to the dynamic shrinking of conditional patterns, which is achieved by removing infrequent items according to the minimum support at the time [10]. Also in our case, thanks to the minimum support translated from the upper bound, irrelevant items are removed individually during the search. The details of this dynamic shrinking will be illustrated in Section 3.5.

3.2 Difficulties in dealing with minimality in top-k mining. Even with top-k mining, there will be uninformative or redundant patterns. For example, suppose that a pattern $\{\mathsf{A}\}$ is significantly relevant to a class c of interest. Then, the patterns including A, such as $\{\mathsf{A},\mathsf{B}\}$, $\{\mathsf{A},\mathsf{C}\}$ and $\{\mathsf{A},\mathsf{B},\mathsf{C}\}$, also tend to be relevant to c, and would occupy the list of the final top-k patterns. To filter out these redundant patterns, some authors introduced the notion of *minimality* w.r.t. set-inclusion [5, 6, 20]. That is, for two candidate patterns $\boldsymbol{x}$ and $\boldsymbol{x}'$, we prefer $\boldsymbol{x}$ over $\boldsymbol{x}'$ and remove $\boldsymbol{x}'$ as a redundant one if $\boldsymbol{x} \subset \boldsymbol{x}'$. In the literature on emerging pattern (EP) mining, a set of minimal EPs is included in the 'left border' in border representation [4] of emerging patterns [5], but as pointed out in [6, 20], it is inefficient to conduct an exhaustive search for all EPs just for having minimal EPs. In top-k mining, there seem to be two following difficulties in dealing with minimality.

First, let us consider two patterns $\boldsymbol{x} = \{\mathsf{A},\mathsf{C}\}$ and $\boldsymbol{x}' = \{\mathsf{A},\mathsf{C},\mathsf{D}\}$ in the prefix enumeration tree in Fig. 1, where $\boldsymbol{x}$ is already stored in the candidate list, and now we are visiting $\boldsymbol{x}'$. Also we use the relevance score R_c, where c is the class of interest. Then, in a fortunate case that $R_c(\boldsymbol{x}) \geq R_c(\boldsymbol{x}')$, we can immediately throw away $\boldsymbol{x}'$. We can see this by imagining two cases. If $\boldsymbol{x}$ is included in the final k patterns, $\boldsymbol{x}'$ should be removed anyway since $\boldsymbol{x}'$ is not minimal; If $\boldsymbol{x}$ is not included in the final k patterns, then $\boldsymbol{x}'$ is not, either, since $R_c(\boldsymbol{x}) \geq R_c(\boldsymbol{x}')$. Unfortunately, when $R_c(\boldsymbol{x}) < R_c(\boldsymbol{x}')$, there arises a difficulty: we cannot immediately decide whether $\boldsymbol{x}'$ should be thrown away or not, since it depends on whether $\boldsymbol{x}$ is included in the final k patterns.

Besides, a depth-first (left-to-right) search over a prefix enumeration tree in Fig. 1 causes another difficulty: even $\boldsymbol{x} = \{\mathsf{A},\mathsf{C}\}$ might be removed when $\boldsymbol{x}'' = \{\mathsf{C}\}$, a sub-pattern of $\boldsymbol{x}$ visited later, is found to be more relevant than $\boldsymbol{x}$, i.e. $R_c(\boldsymbol{x}'') \geq R_c(\boldsymbol{x})$. This dependency propagates from patterns to patterns, and eventually makes it complicated to maintain the candidate list. What is worse is that, if we add the patterns that may be removed later, due to the violation of minimality, into the extended candidate list, the size of the list could be much larger than k, and hence the

[4] Due to the space limit, the derivation of $\overline{\chi}^2_c(\boldsymbol{x})$ is omitted and will appear in the full paper.

[5] For many relevance scores, this rewriting is possible in closed form. A typical exception is the case with information gain (IG), since IG includes the logarithm function in its definition.

effect of minimum support raising should be limited.

The former difficulty comes from a fact that minimality is usually determined given a *fixed* set of patterns, and does not harmonize with the dynamic behavior of the candidate list. On the other hand, a straightforward solution for the latter difficulty is to conduct a breadth-first (level-wise) search, where longer patterns are always visited after shorter patterns, but in general, breadth-first styles have a problem in memory consumption. Next, in Section 3.3, instead of minimality, we introduce a light-weight notion, called *weakness*, which is defined for a *pair* of patterns, for filtering out redundant patterns. We then show that, based on the enumeration trees originally used in FP-growth, it is still possible in depth-first search to check the weakness between patterns efficiently, and furthermore in Section 3.4, an additional, aggressive pruning is available, that exploits an extended notion of weakness.

3.3 Weak relevant patterns. Instead of minimality, now we introduce a notion called *weakness* for a pair of patterns. This notion itself was originally introduced in Yuan et al.'s framework called *most relevant explanation* [23] for explanatory analysis of Bayesian networks. The definition of weakness is given as follows:

DEFINITION 3.2. *Let c be a class of interest. Then, for a pair of patterns $\boldsymbol{x}$ and $\boldsymbol{x}'$ in $\mathcal{P}$, $\boldsymbol{x}'$ is weaker than $\boldsymbol{x}$ if and only if $\boldsymbol{x}' \supset \boldsymbol{x}$ and $R_c(\boldsymbol{x}') \le R_c(\boldsymbol{x})$.*

We can say first that weakness is intuitive and reasonable. For example, if $R_c(\{\mathsf{A},\mathsf{C},\mathsf{D}\}) > R_c(\{\mathsf{A},\mathsf{C}\})$, there would be something meaningful in the combination of A, C and D on comparison with $\{\mathsf{A},\mathsf{C}\}$, and it is natural to keep $\{\mathsf{A},\mathsf{C},\mathsf{D}\}$ as a candidate. On the other hand, if $R_c(\{\mathsf{A},\mathsf{C},\mathsf{D}\}) \le R_c(\{\mathsf{A},\mathsf{C}\})$, adding D into $\{\mathsf{A},\mathsf{C}\}$ would be redundant, or even harmful.

Furthermore, to see a procedural aspect of weakness, let us consider the situation where we wish to have non-weak relevant patterns for class c, and two patterns $\boldsymbol{x} = \{\mathsf{A},\mathsf{C}\}$ and $\boldsymbol{x}' = \{\mathsf{A},\mathsf{C},\mathsf{D}\}$ are under study. In this situation, if $R_c(\boldsymbol{x}) \ge R_c(\boldsymbol{x}')$, then $\boldsymbol{x}'$ is weaker than $\boldsymbol{x}$, and hence we immediately throw away $\boldsymbol{x}'$, as in the case with minimality. On the other hand, if $R_c(\boldsymbol{x}) < R_c(\boldsymbol{x}')$, then $\boldsymbol{x}'$ is *not* weaker than $\boldsymbol{x}$, and hence the removal of $\boldsymbol{x}'$ does not depend on whether $\boldsymbol{x}$ remains in the final top-k patterns. So, to ensure a safe addition of $\boldsymbol{x}' = \{\mathsf{A},\mathsf{C},\mathsf{D}\}$ into the candidate list, it is sufficient to know that the relevance score of $\boldsymbol{x}'$ is higher than those of its all sub-patterns $\{\mathsf{A}\}$, $\{\mathsf{C}\}$, $\{\mathsf{D}\}$, $\{\mathsf{A},\mathsf{C}\}$, $\{\mathsf{A},\mathsf{D}\}$ and $\{\mathsf{C},\mathsf{D}\}$. Now the first difficulty with minimal relevant patterns can be mitigated by using weakness.

The second difficulty in checking the minimality in a depth-first search will be eliminated by considering enumeration trees, hereafter called *suffix enumeration trees*, illustrated in Fig. 2. In a suffix enumeration tree, the parent of each node is its immediate suffix, and siblings are placed following the lexicographical order based on $\prec$ (i.e. left-to-right in Fig. 2). Interestingly, FP-growth implicitly uses suffix enumeration trees when the items in each transactions are sorted according to $\prec$. Here, a key fact is that, *in a depth-first search over a suffix enumeration tree, all of $\boldsymbol{x}$'s sub-patterns have been visited when $\boldsymbol{x}$ itself is visited.* For example, in Fig. 2, before visiting $\{\mathsf{A},\mathsf{C},\mathsf{D}\}$, we visit $\{\mathsf{A}\}$, $\{\mathsf{C}\}$, $\{\mathsf{D}\}$, $\{\mathsf{A},\mathsf{C}\}$, and so on. To summarize, together with weakness and suffix enumeration trees, for adding $\boldsymbol{x}$ into the candidate list safely, it is sufficient to check whether $\boldsymbol{x}$ is not weaker than the existing patterns in the list.

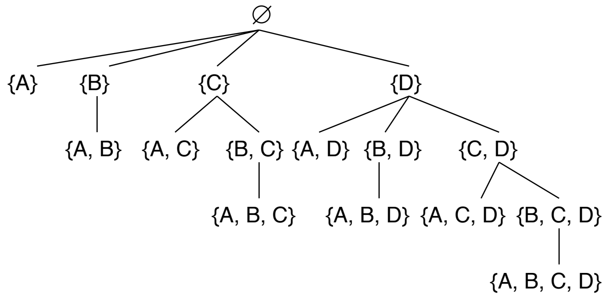

Figure 2: An example of a suffix enumeration tree.

3.4 Pruning based on weakness. Moreover, we can introduce an additional pruning based on the notion called *prunable weakness*, defined as follows:

DEFINITION 3.3. *Let c be a class of interest. Then, for a pair of patterns $\boldsymbol{x}$ and $\boldsymbol{x}'$ in $\mathcal{P}$, $\boldsymbol{x}'$ is prunably weaker than $\boldsymbol{x}$ if and only if $\boldsymbol{x}' \supset \boldsymbol{x}$ and $\overline{R}_c(\boldsymbol{x}') \le R_c(\boldsymbol{x})$.*

Prunable weakness implies weakness in the original sense. Let us suppose that we are visiting $\boldsymbol{x}'$ which is prunably weaker than $\boldsymbol{x}$ existing in the candidate list at the moment. Then, since the upper bound $\overline{R}_c(\boldsymbol{x}')$ of the relevance score of $\boldsymbol{x}'$ is no greater than $R_c(\boldsymbol{x})$, the relevance scores of the super-patterns of $\boldsymbol{x}'$ are never greater than $R_c(\boldsymbol{x})$ either, and hence these super-patterns are all weaker than $\boldsymbol{x}$. So we can safely prune the subtree rooted by $\boldsymbol{x}'$. This pruning is aggressive in that there are (exponentially) many chances to prune long patterns. For example, $\{\mathsf{A},\mathsf{C},\mathsf{D}\}$ should not be prunably weaker than any of its all (up-to-six) sub-patterns remaining in the current candidate list.

Besides, in a suffix enumeration tree, the nodes in a path from the root to a pattern $\boldsymbol{x}$ are all sub-patterns of $\boldsymbol{x}$. For this 'built-in' relation, we can save the effort further, by skipping the check on set-

Table 1: Example transactions (left) and F-scores of items (right).

Class c	Transaction
$+$	$\{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}$
$+$	$\{\mathsf{A}, \mathsf{B}, \mathsf{D}, \mathsf{F}\}$
$+$	$\{\mathsf{B}, \mathsf{C}, \mathsf{E}\}$
$-$	$\{\mathsf{A}, \mathsf{C}\}$
$-$	$\{\mathsf{B}, \mathsf{C}, \mathsf{D}\}$
$-$	$\{\mathsf{A}, \mathsf{C}, \mathsf{E}, \mathsf{F}\}$

Item x	$N(+, x)$	$N(-, x)$	$F_+(x)$
B	3	1	0.857
D	2	1	0.667
A	2	2	0.571
C	2	3	0.500
E	1	1	0.400
F	1	1	0.400

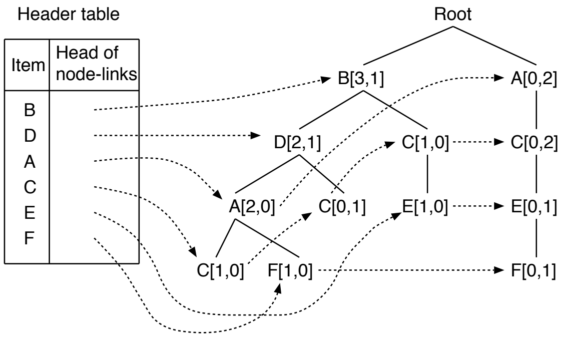

Figure 3: The initial RP-tree and its header table for example transactions.

inclusion in judging the prunable weakness. To be more specific, we first prepare a threshold called a *local minimum support* $\sigma^\pi_{\min}$ for each path π from the root to a leaf in the enumeration tree. Then, after visiting a pattern $\boldsymbol{x}$, for each path π that includes $\boldsymbol{x}$, $\sigma^\pi_{\min}$ is updated by $\sigma^\pi_{\min} := \max\{U_c(\boldsymbol{x}), \sigma^\pi_{\min}\}$, where $U_c(\boldsymbol{x})$ is obtained by the translation described in Section 3.1. As a result, each pattern is doubly qualified by the original minimum support and the local minimum support. Each local minimum support is easily implemented as an argument variable of a recursive call realizing a depth-first search (the details are given in Section 3.6).

3.5 RP-trees. *RP-trees* (relevant-pattern trees) are a simple extension of FP-trees, used in FP-growth [10].[6] An RP-tree is a prefix tree that compactly stores the transactions (or their subsets) in the dataset. Hereafter we simply refer to the class of interest by $+$, and suppose that the transactions in the other classes are merged into one class $-$. Now let us consider an example[7] of a dataset of six transactions, shown in Table 1 (left). Each transaction belongs to one of two classes $+$ and $-$, and let N^+ and N^- be the numbers of transactions in class $+$ and $-$, respectively. In this example, $N^+ = N^- = 3$.

In advance of constructing the initial RP-tree, similarly to the case with FP-tree, we scan the dataset and obtain the counts for computing the relevance scores of items. For the dataset in Table 1 (left), we obtain $N(+, x)$ (resp. $N(-, x)$) as the counts of each item x in the transactions in class $+$ (resp. $-$), and its F-score $F_+(x)$ as shown in Table 1 (right). For example, $F_+(\mathsf{B})$, the F-score of item B, is computed as the harmonic mean of $p(+ \mid \mathsf{B}) = N(+, \mathsf{B})/(N(+, \mathsf{B}) + N(-, \mathsf{B})) = 3/4$ and $p(\mathsf{B} \mid +) = N(+, \mathsf{B})/N^+ = 1$. Then, we define the canonical order $\prec$ among the items by firstly the magnitude of their F-scores and secondly their alphabets (i.e. $\mathsf{B} \prec \mathsf{D} \prec \mathsf{A} \prec \mathsf{C} \prec \mathsf{E} \prec \mathsf{F}$ in Table 1), and reorder the items in each transaction based on $\prec$.

The initial RP-tree is then constructed in the same way as FP-tree except that we separately count the occurrences of patterns in class $+$ and those in class $-$. For example, Fig. 3 illustrates the initial RP-tree for the dataset in Table 1, and the path from the root to the node labeled by $\mathsf{A}[2, 0]$ indicates that a pattern $\{\mathsf{B}, \mathsf{D}, \mathsf{A}\}$ occurs twice in the transactions of class $+$, and never occurs in the transactions of class $-$. Also as in Fig. 3, we construct the header table where an item x is associated with the header of node-links connecting all nodes with the labels of the form $x[n^+, n^-]$. Hereafter n^+ and n^- are called the *positive count* and the *negative count* of the corresponding node.

Also similarly to FP-growth, RP-growth traverses a suffix enumeration tree, recursively constructing RP-trees called *conditional RP-trees*. At beginning, the initial RP-tree in Fig. 3 contains all transactions, and let us consider the case where we work on pattern $\{\mathsf{E}\}$ by inserting E to the initial pattern $\emptyset$, and the initial minimum support is set to $1/N^+ = 1/3$. Then, Fig. 4 (above) shows how to construct the RP-tree conditioned E from the initial RP-tree. That is, starting from the head of node-links associated with E, we first find two nodes labeled by $\mathsf{E}[1, 0]$ and $\mathsf{E}[0, 1]$ in the initial RP-tree. Then, we obtain a *counted pattern* $\{\mathsf{B}, \mathsf{C}\}[1, 0]$ from the set of nodes *between* the root and the node labeled by $\mathsf{E}[1, 0]$. The pair $[1, 0]$ of positive counts and negative counts associated with $\{\mathsf{B}, \mathsf{C}\}$ is inherited from $\mathsf{E}[1, 0]$. Similarly we obtain another counted pattern $\{\mathsf{A}, \mathsf{C}\}[0, 1]$ using $\mathsf{E}[0, 1]$. Note here that a pair of positive/negative counts $[N(+, \{\mathsf{E}\}), N(-, \{\mathsf{E}\})]$ of pattern $\{\mathsf{E}\}$, which we are working on, can be computed as the sum of the counts in these counted patterns, i.e. $[1, 0] + [0, 1] = [1, 1]$, and the F-score of pattern $\{\mathsf{E}\}$ is computed by the harmonic mean of $N(+, \{\mathsf{E}\})/(N(+, \{\mathsf{E}\}) + N(-, \{\mathsf{E}\})) = 1/(1 + 1) = 1/2$ and $N(+, \{\mathsf{E}\})/N^+ = 1/3$. In the next step, we count

[6] In the literature of emerging pattern mining, some authors use CP-trees (contrast-pattern trees) [6, 20], which have a similar structure to RP-trees. The difference from our method is that we do not perform the merging operation introduced in [6].

[7] This example was originally introduced in [20].

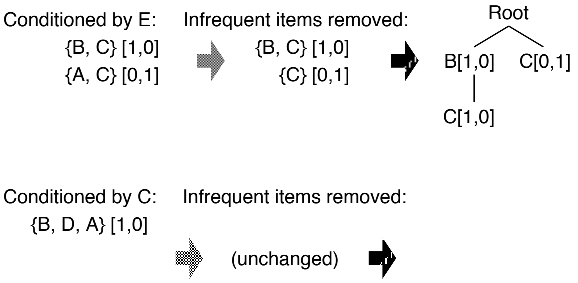

Figure 4: Construction of an RP-tree conditioned on E (above) and C (below).

the individual items in these counted patterns, and shrink them by removing infrequent items. In Fig. 4 (above), the pairs of positive/negative counts of A, B and C are $[0, 1]$, $[1, 0]$ and $[1, 1]$, respectively. Since the minimum support is set to $1/N^+ = 1/3$, we remove item A, whose positive count is 0, from the counted patterns. Note here that this shrinking step corresponds to pruning based on the current minimum support. Finally, an RP-tree conditioned on E is constructed as in the right hand side of Fig. 4 (above). Fig. 4 (below) illustrates a construction of an RP-tree conditioned on C in the case where we work on pattern $\{\mathsf{C}\}$.

As seen above, dynamic shrinking of RP-trees is realized by removing infrequent items based on the current minimum support, and accelerates the search by narrowing the enumeration tree. For example, in Fig. 4, we only have to extend $\{\mathsf{E}\}$ into $\{\mathsf{B}, \mathsf{E}\}$ and $\{\mathsf{C}, \mathsf{E}\}$, since there are only B and C in the conditional RP-tree. As mentioned in Section 3.1, even in top-k mining of relevant patterns, we can benefit from the mechanism of dynamic shrinking, thanks to the translation of branch-and-bound pruning into minimum support raising.

3.6 RP-growth. Based on the descriptions in the previous subsections in Section 3, RP-growth is now presented as Algorithm 1 and 2, and works for the top-k mining task defined at the beginning of Section 3.[8] RP-Growth (Algorithm 1) is the main routine, and Grow (Algorithm 2) is a recursive routine called from RP-Growth. For brevity, in these procedures, we introduce five global variables c, k, $\sigma_{\min}$, $\beta_{\min}$ and L, which stand for the class of interest, the number of relevant patterns to return, the minimum support, the minimum precision and the candidate list, respectively.

Algorithm 1 RP-Growth(c_0, k_0, $\mathcal{D}$)

```
Require: $c_0$: the class of interest, $k_0$: the number of non-weak relevant patterns, $\mathcal{D}$: the dataset
$c := c_0$ and $k := k_0$
$L :=$ an empty candidate list
$\boldsymbol{x} :=$ an empty pattern
$\sigma_{\min} := 1/|\mathcal{D}|$
$\beta_{\min} := 0.5$
$T :=$ the initial RP-tree constructed from $\mathcal{D}$
Call Grow($T$, $\boldsymbol{x}$, $1/|\mathcal{D}|$)
Output the patterns in $L$
```

First, in RP-Growth, we initialize the variables, construct the initial RP-tree as described in Section 3.5, and call Grow. After the call, top-k patterns are stored in the candidate list L, and hence we output L as a final result. In each call of Grow$(T_0, \boldsymbol{x}_0, \sigma_0)$, we consider the key items in the header table H_0 (lines 1–2). Such key items are the items appearing in T_0.[9] Inside the loop for each key item x, we first work for a construction of a RP-tree conditioned on x as described in Section 3.5 (line 3), where the pruning based on the original/local minimum support ($\sigma_{\min}$ and σ_0) is embedded.[10] We then extend the current pattern by x (line 4) and compute the relevance score $R_c(\boldsymbol{x})$ of the extended pattern $\boldsymbol{x}$ (line 5). If the extended pattern satisfies the constraint on precision (line 6), we raise the local minimum support (line 7) using the computed $R_c(\boldsymbol{x})$. For example, with F-score, we use $U_c(\boldsymbol{x}) = \mathrm{F}_c(\boldsymbol{x})/(2 - \mathrm{F}_c(\boldsymbol{x}))$. Then, if $\boldsymbol{x}$ is *not* weaker than any patterns in the candidate list L (line 8), we add $\boldsymbol{x}$ and then if L is full, we remove the low-ranked patterns (line 11), and raise the minimum support (line 12). If $\boldsymbol{x}$ is weaker than some pattern in L, we do not add $\boldsymbol{x}$ into L, but try to continue the search for longer patterns. In contrast, if $\boldsymbol{x}$ is prunably weaker than some pattern in L, we stop the search. For $\boldsymbol{x}$ having passed the filter on prunable weakness, we call Grow recursively (line 17).

4 Experiments

4.1 Basic settings. We conducted three experiments using the 20 newsgroups dataset. Following the discussions in Sections 2.2 and 3.1, we chose three relevance scores in these experiments: χ^2, F-score and support difference. The purpose of the first experiment is to confirm the intuitiveness of the relevant patterns obtained by RP-growth. In the second one, we make a comparison among the three relevance scores

[8] For allowing weak patterns, some modifications are needed, i.e. we remove the third argument of Grow and lines 7, 8, 14, 16 and 18 in Grow.

[9] For example, B, D, A, C, E and F are such key items in Fig. 3.

[10] As a further optimization, one may plug the tasks in lines 5–7 into the construction step at line 3, to enable pruning based on $\sigma_{\min}$ and σ, where σ is the updated local minimum support.

Algorithm 2 GROW(T_0, $\boldsymbol{x}_0$, σ_0)

```
Require: $T_0$: the current RP-tree, $\boldsymbol{x}_0$: the current pattern, $\sigma_0$: the current local minimum support
$H_0 :=$ the header table associated with $T_0$
for all $x$ in the key items of $H_0$ do
    Construct an RP-tree $T$ conditioned on $x$ from $T_0$ (pruning based on $\sigma_{\min}$ and $\sigma_0$ is embedded)
    $\boldsymbol{x} := \{x\} \cup \boldsymbol{x}_0$
    Compute $R_c(\boldsymbol{x})$ and $p(c \mid \boldsymbol{x})$ from the positive/negative counts stored in $T$
    if $p(c \mid \boldsymbol{x}) \geq \beta_{\min}$ then
        $\sigma := \max\{U_c(\boldsymbol{x}), \sigma_0\}$
        if $\boldsymbol{x}$ is not weaker than any patterns in $L$ then
            Insert $\boldsymbol{x}$ into $L$ following the descending order of $R_c$
            if $|L| \geq k$ then
                Remove the patterns with the $(k+1)$-th greatest or smaller relevance scores (if any) from $L$
                $\sigma_{\min} := \max\{U_c(\boldsymbol{z}), \sigma_{\min}\}$, where $\boldsymbol{z}$ is the $k$-th pattern in $L$
            end if
        end if
    end if
    if $\boldsymbol{x}$ is not prunably weaker than any patterns in $L$ then
        Call GROW($T$, $\boldsymbol{x}$, $\sigma$)
    end if
end for
```

above from the viewpoint of the size of the search space. The last experiment explores the usefulness of relevant patterns by seeing whether the new features constructed from relevant patterns can enhance the predictive power of a classifier. We expect relevant patterns to be *combined features* which are more discriminative than the original (singleton) features. Previously, frequent patterns [3, 16] and discriminative patterns [5, 6, 20] were used for a similar objective. The 20 newsgroups dataset is originally a collection of approximately 20,000 articles from 20 different newsgroups, each of which is seen as a pre-defined class. We used a preprocessed dataset available from `http://people.csail.mit.edu/jrennie/20Newsgroups/` and made further preprocessing.[11] The dataset was finally converted into 17,930 articles containing 5,666 words. Each article was then transformed into a transaction t, where each item $w \in t$ is a word in the article. From now, we report the results of three experiments explained above in turn.

4.2 Intuitiveness of the relevant patterns.

In the first experiment, we confirm the intuitiveness of the relevant patterns obtained by RP-growth. To find the patterns relevant to a class (newsgroup) c of interest, we binarize the dataset $\mathcal{D}$ according to c. That is, we rename class c as class $+$ and merge the transactions of the other 19 classes into class $-$. Table 2 lists top-25 non-weak relevant patterns for three classes: `comp.graphics`, `rec.sport.hockey` and `talk.politics.guns` (the other results are omitted due to the space limit), under F-score with the threshold $\beta_{\min}$ for precision $p(c \mid \boldsymbol{x})$ being 0.5. Table 2 shows that the firstly-ranked pattern for each newsgroup includes the part of the name of the newsgroup (the word suffixes have been replaced or removed by the stemmer), and most of the other patterns are also related to the newsgroup. Interestingly, for the newsgroup `comp.graphics`, the word graphic cannot be a relevant pattern itself, but becomes a relevant pattern by being combined with another word. Besides, each list of top-25 non-weak relevant patterns mixes well discriminative patterns (with high precision; such as {polygon} for `comp.graphics`) and supportive patterns (with high recall; such as {gun} for `talk.politics.guns`). Lastly we confirmed the importance of weakness. For example, when allowing weak patterns for `talk.politics.guns`, we have 16 patterns including the word gun in the top-25 list.

On the other hand, since we have 20 classes, the binarized class distribution $\{p(+), p(-)\}$ is basically skewed (e.g. $p(+) = 899/17930 \approx 0.05$ when `comp.graphics` is the class of interest), and thus in this dataset, the setting $\beta_{\min} = 0.5$ tends to produce the patterns with relatively low support. Presumably from this reason, for some newsgroups such as `alt.atheism`, the top-25 list contains many patterns of proper nouns, e.g. {keith, caltech}. In such a case, setting $\beta_{\min}$ between $p(+)$ and 0.5 would produce a more intuitive result.

[11] To be more specific, we removed stop words based on the list used by the SMART system, performed stemming by the Porter's algorithm [17], removed infrequent words (< 50 occurrences), and removed short articles (< 10 words).

Table 2: Top-25 non-weak relevant patterns in the 20 newsgroup dataset under F-score with $\beta_{\min} = 0.5$.

$c =$ `comp.graphics`

Pattern $\boldsymbol{x}$	$p(c \mid \boldsymbol{x})$	$p(\boldsymbol{x} \mid c)$	$\mathrm{F}_c(\boldsymbol{x})$
{graphic, program}	0.537	0.136	0.217
{gif}	0.552	0.119	0.196
{graphic, imag}	0.642	0.108	0.185
{imag, program}	0.516	0.110	0.181
{imag, file}	0.531	0.105	0.175
{graphic, find}	0.578	0.087	0.151
{imag, bit}	0.514	0.083	0.144
{graphic, code}	0.613	0.081	0.143
{graphic, bit}	0.545	0.080	0.140
{graphic, packag}	0.591	0.076	0.134
{format, convert}	0.588	0.075	0.132
{graphic, comp}	0.730	0.072	0.132
{imag, format}	0.613	0.072	0.129
{graphic, point}	0.573	0.070	0.125
{graphic, format}	0.670	0.068	0.123
{imag, convert}	0.596	0.066	0.118
{polygon}	0.915	0.060	0.113
{imag, softwar}	0.500	0.062	0.111
{graphic, ftp}	0.500	0.061	0.109
{graphic, algorithm}	0.852	0.058	0.108
{jpeg}	0.825	0.058	0.108
{graphic, group}	0.514	0.060	0.108
{graphic, site}	0.530	0.059	0.106
{graphic, comput, articl}	0.525	0.059	0.106
{code, algorithm}	0.500	0.059	0.105

$c =$ `rec.sport.hockey`

Pattern $\boldsymbol{x}$	$p(c \mid \boldsymbol{x})$	$p(\boldsymbol{x} \mid c)$	$\mathrm{F}_c(\boldsymbol{x})$
{hockei}	0.943	0.377	0.538
{team}	0.519	0.473	0.495
{playoff}	0.943	0.277	0.428
{game, plai}	0.506	0.273	0.354
{nhl}	0.990	0.206	0.341
{cup}	0.584	0.195	0.292
{player, plai}	0.575	0.190	0.286
{score}	0.510	0.194	0.281
{game, player}	0.561	0.186	0.280
{game, goal}	0.899	0.157	0.267
{game, win}	0.517	0.174	0.260
{game, fan}	0.622	0.164	0.260
{plai, goal}	0.852	0.144	0.246
{wing}	0.515	0.156	0.240
{leaf}	0.894	0.132	0.230
{bruin}	1.000	0.130	0.230
{pittsburgh}	0.567	0.142	0.226
{game, watch}	0.621	0.136	0.224
{detroit}	0.733	0.131	0.222
{penguin}	0.871	0.127	0.222
{game, season}	0.539	0.137	0.219
{game, night}	0.660	0.129	0.216
{ranger}	0.629	0.129	0.214
{plai, win}	0.529	0.134	0.214
{plai, fan}	0.603	0.128	0.211

$c =$ `talk.politics.guns`

Pattern $\boldsymbol{x}$	$p(c \mid \boldsymbol{x})$	$p(\boldsymbol{x} \mid c)$	$\mathrm{F}_c(\boldsymbol{x})$
{gun}	0.540	0.414	0.469
{weapon}	0.528	0.253	0.342
{fbi}	0.506	0.246	0.331
{firearm}	0.884	0.196	0.321
{batf}	0.662	0.155	0.252
{waco}	0.543	0.154	0.240
{assault}	0.587	0.124	0.205
{cdt, sw}	0.933	0.110	0.196
{cdt, stratu}	0.916	0.110	0.196
{handgun}	0.818	0.111	0.195
{cdt}	0.817	0.110	0.193
{stratu, sw}	0.700	0.110	0.190
{fire, compound}	0.698	0.109	0.188
{stratu}	0.570	0.110	0.184
{bd}	0.530	0.110	0.182
{sw}	0.521	0.110	0.181
{atf}	0.692	0.101	0.176
{arm, law}	0.527	0.086	0.148
{compound, dai}	0.598	0.082	0.144
{nra}	0.696	0.079	0.143
{rocket, special}	0.750	0.077	0.140
{rocket, speak}	0.840	0.076	0.139
{rocket, vo}	0.918	0.075	0.139
{vo, investor}	0.918	0.075	0.139
{vo, speak, todai}	0.918	0.075	0.139

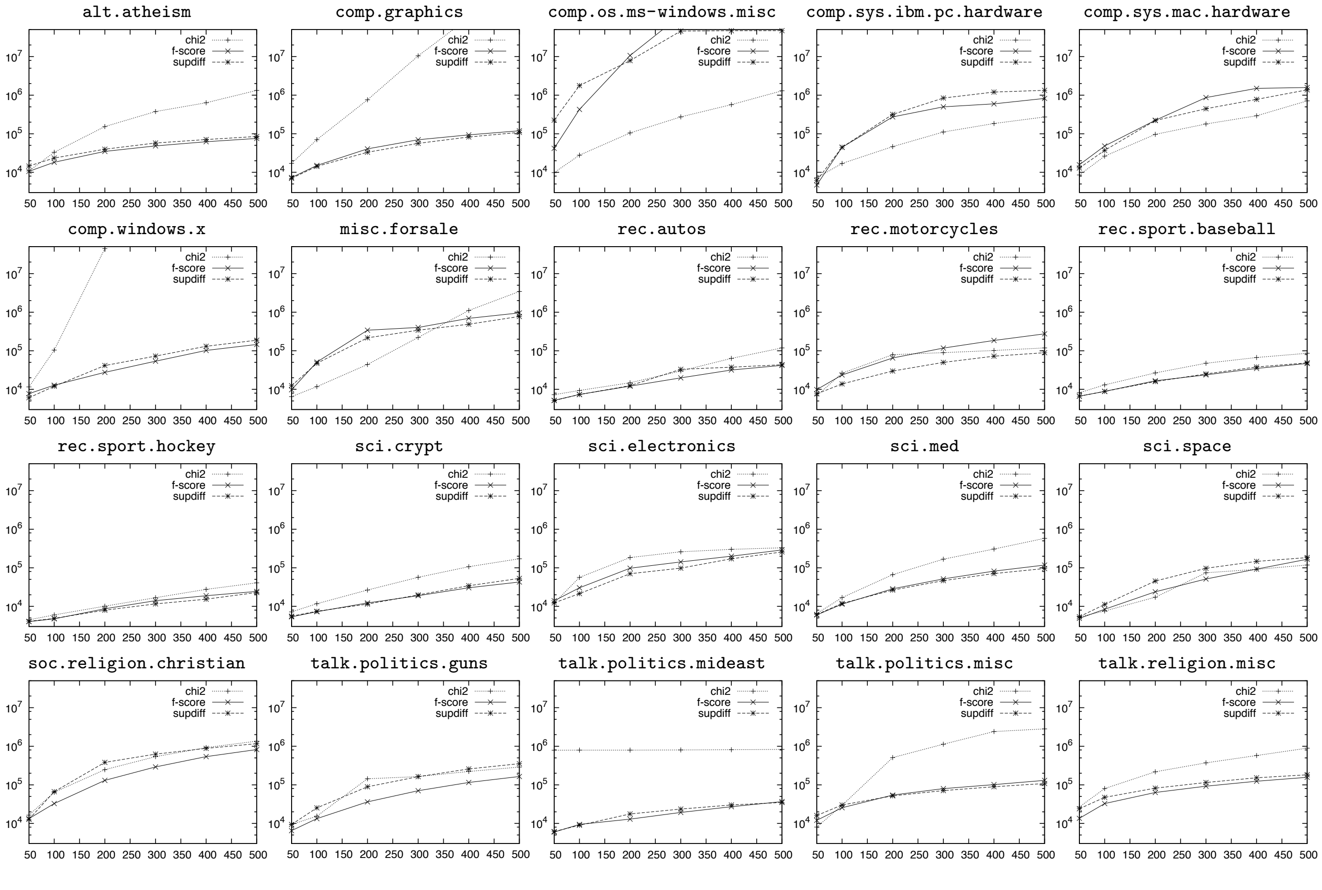

Figure 5: The numbers of visited nodes in the enumeration tree for each newsgroup (the y-axis), varying the number k of non-weak relevant patterns to find (the x-axis). In each graph, "chi2," "f-score" and "supdiff" respectively indicate χ^2, F-score and support difference.

4.3 Search space. The second experiment uses the same dataset as that in the first experiment, and among the relevance score χ^2, F-score and support difference, we compare the size of the search space, i.e. the number of visited nodes in an enumeration tree, for each newsgroup, varying the number k of patterns to be found. The graphs in Fig. 5 show the results under $\beta_{\min} = 0.5$. These graphs show that, when using χ^2, the size of the search space drastically varies according to the newsgroup. Indeed, for `comp.graphics` and `comp.windows.x` with large k, RP-growth with χ^2 did not finish the search within two hours (implementation language: Java, CPU: Core i7 2.66GHz). Contrastingly, F-score (resp. support difference) finished the search within a minute for all newsgroups except `comp.os.ms-windows.misc`, and for this newsgroup, it took about 17 minutes (resp. 5 minutes) even with $k = 500$.

4.4 Classification performance. In the third experiment, we aim to build an accurate classifier that predicts the newsgroup of a given article. We first describe additional settings for this experiment. In the 20 newsgroup dataset, each transaction t is converted into a binary vector, where each feature corresponds to a word w and its value is set as 1 if $w \in t$ and 0 otherwise. Furthermore, from the relevant patterns obtained by RP-growth, we construct combined features. That is, each combined feature corresponds to a relevant pattern $\boldsymbol{x}$ and its value is set as 1 if $\boldsymbol{x} \subseteq t$ and 0 otherwise. The base classifiers are support vector machines (SVMs). For evaluation of trained SVMs, we conduct stratified 10-fold cross-validation [12], i.e. we first split the entire dataset into 10 partial, equally-sized ones keeping the class distribution. Then, in each fold, in a rotating way, we use eight partial datasets (80%) for training, one partial dataset (10%) for testing, and the remainder (10%) as a held-out dataset for tuning the user-specified parameters of SVMs and RP-growth (k and $\beta_{\min}$). LIBSVM[12] is used as an implementation of SVMs, and we use two types of kernels: linear kernels whose parameter is usually referred to by C, and RBF kernels with parameters C and γ.[13] In this experiment, we compare SVMs under three configurations:

1. Linear kernel with singleton (original) features
2. RBF kernel with singleton features
3. Linear kernel with singleton/combined features

These configurations are referred to by L+S, RBF+S and L+SC, respectively. First, L+S gives the baseline in this comparison, and RBF+S shows the performance with an advanced kernel. Meanwhile, our attention is mainly paid to L+SC, since it uses combined features constructed from relevant patterns. The combined features above are created from the relevant patterns of length two or longer, obtained with three relevance scores: χ^2, F-score and support difference.

The results of evaluation on classification performance in the three configurations above are presented in Table 3.[14] In this table, each row except the last one corresponds to a class (newsgroup) c, and contains the average F-scores on classification for the dataset binarized according to the class c. The last row ("*All*") contains average accuracies on classification for the original, multi-class dataset where we add all combined features obtained by RP-growth. In each cell, the figure after ± is the standard error in cross-validation, and for each row, the best (resp. the second best) F-score/accuracy is marked by '$**$' (resp. by '$*$').

From Table 3, we first see that the configurations RBF+S and L+SC outperform the baseline L+S. Although RBF+S works best under F-score in half of 20 newsgroups, for multi-class classification (the last row), the performance of L+SC is close to that of RBF+S. Also we would like to add that the advantage of the use of combined features over RBF kernels is that the combined features are interpretable and would make further analyses easy. For example, one may explain why the classification for `comp.graphics` is difficult while the classification for `rec.sport.hockey` is easy, by showing the F-scores of top-25 relevant patterns in Table 2. Besides, in L+SC, we could not observe a significant deference among χ^2, F-score and support difference.

5 Related work

In this section, we focus on the difference between RP-growth and several existing works closely related to RP-growth. First, as far as we know, KORD [21] firstly explored top-k mining of discriminative patterns. One advantage of RP-growth over KORD should be its simplicity. Indeed, an upper bound of leverage, KORD's primary relevance score, is easily derived, and therefore KORD that seeks for class association rules $\boldsymbol{x} \Rightarrow c$ is realized as RP-growth.[15] Besides, KORD only considers constraints on the form of each individual

[12] `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

[13] Based on the available held-out data, we conducted a grid search with refining for user-specified parameters, i.e. C for linear kernels was chosen from $\{2^i \mid i = -7, -6, \ldots, 4, 5\}$, C for RBF kernels from $\{2^i \mid i = -7, -6, \ldots, 8, 9\}$, γ from $\{2^i \mid i = -13, -12, -11, -10, -9, -8, -7, -6, -5, -3, -1, 1, 3\}$, k from $\{50, 100, 200, 300, 400, 500, 600, 700\}$, and $\beta_{\min}$ from $\{0.2, 0.3, 0.4, 0.5\}$.

[14] The combined features with χ^2 are not available for the newsgroups `comp.graphics`, `comp.windows.x` and `misc.forsale`, since RP-growth did not finish the search within two hours (Fig. 5 clearly shows the behavior for such newsgroups).

[15] KORD is generally designed to find $\boldsymbol{x} \Rightarrow y$, where y is an arbitrary item not appearing in $\boldsymbol{x}$.

Table 3: Average F-scores and accuracies (%) on classification by SVMs.

Class of interest	Singleton		Singleton/Combined		
	Linear	RBF	Linear		
			χ^2	F-score	Support diff.
`alt.atheism`	**83.97±1.39	*83.78±1.37	83.68±1.14	83.68±1.42	83.76±1.41
`comp.graphics`	71.18±0.97	**72.96±1.13	N/A	*72.02±0.79	71.98±0.87
`comp.os.ms-windows.misc`	75.73±1.28	**77.57±1.03	75.84±1.20	76.17±1.31	*76.25±1.28
`comp.sys.ibm.pc.hardware`	68.75±1.89	*68.93±1.34	68.45±1.32	**69.26±1.60	68.92±1.80
`comp.sys.mac.hardware`	*79.69±1.23	**80.85±1.29	79.63±1.60	78.72±1.37	78.68±1.43
`comp.windows.x`	80.65±1.04	*80.68±1.20	N/A	80.63±1.15	**81.06±1.23
`misc.forsale`	79.59±1.17	**80.08±0.96	N/A	*79.85±1.23	79.83±1.12
`rec.autos`	81.20±1.16	**82.29±1.38	81.15±1.29	81.24±1.12	*81.47±1.24
`rec.motorcycles`	91.22±0.53	91.06±0.48	*91.70±0.46	91.67±0.57	**91.70±0.53
`rec.sport.baseball`	90.14±0.51	90.49±0.54	**90.72±0.52	90.39±0.48	*90.51±0.45
`rec.sport.hockey`	94.35±0.50	94.44±0.50	94.68±0.47	*94.89±0.53	**94.95±0.54
`sci.crypt`	91.27±0.60	91.31±0.64	**91.63±0.58	*91.61±0.56	91.39±0.61
`sci.electronics`	71.27±0.89	**73.65±1.58	71.56±1.00	73.21±0.99	*73.35±0.86
`sci.med`	85.98±0.78	*86.59±0.70	86.42±0.72	**86.60±0.70	86.42±0.74
`sci.space`	89.75±0.55	90.22±0.57	**90.85±0.43	*90.54±0.44	90.49±0.47
`soc.religion.christian`	85.41±0.65	**86.12±0.64	*86.12±0.66	86.03±0.55	85.90±0.58
`talk.politics.guns`	83.19±0.93	**84.62±1.10	*84.21±1.01	83.89±1.15	83.82±1.17
`talk.politics.mideast`	*92.65±0.72	**92.70±0.72	92.33±0.65	92.22±0.66	92.10±0.67
`talk.politics.misc`	76.34±1.23	**77.28±1.13	76.10±1.13	*76.96±1.25	76.62±1.50
`talk.religion.misc`	62.61±1.44	*62.75±1.50	**63.50±1.84	61.89±2.00	62.52±1.71
All	83.88±0.20	**84.95±0.22	84.48±0.13	84.73±0.22	*84.73±0.23

rule, and does not consider constraints *between* rules like weakness. Terlecki and Walczak [20] later proposed top-k minimal jumping EPs, which are highly discriminative patterns satisfying $p(\boldsymbol{x} \mid c) > 0$ and $p(\boldsymbol{x} \mid \neg c) = 0$ (these conditions jointly imply $p(c \mid \boldsymbol{x}) = 1$). Jumping EPs are a special case of relevant patterns targeted in this paper, and as discussed in Section 3.2 and 3.3, weakness is easier to handle than minimality and is still meaningful.

Branch-and-bound search is a generic technique in pattern mining where the relevance score does not satisfy the anti-monotonicity w.r.t. set-inclusion among patterns. Unlike AprioriSMP [19] and the CG algorithm [24], RP-growth can deal with non-convex relevance score such as F-score. An upper bound of weighted relative accuracy $p(\boldsymbol{x})(p(c \mid \boldsymbol{x}) - p(c))$, a relevance score equivalent to leverage, was introduced as $p(\boldsymbol{x})(1 - p(c))$ [22], but in our translation, the upper bound can be tighter: $p(c)p(\boldsymbol{x} \mid c)(1 - p(c))$. Additionally in RP-growth, the translation of branch-and-bound pruning into minimum support raising leads to effective dynamic shrinking of RP-trees, by removing unpromising items based on the current minimum support.

Recently, Fang et al. introduced an anti-monotonic relevance score called SupMax$K$ for finding low-support discriminative patterns [7]. SupMax$K$ is written as $\text{SupMax}K_c(\boldsymbol{x}) = p(\boldsymbol{x} \mid c) - \max_{\boldsymbol{x}' \subset \boldsymbol{x}, |\boldsymbol{x}'| = K} p(\boldsymbol{x}' \mid \neg c)$ in our notation. SupMax$K$ is attractive since it is anti-monotonic itself, but its computation seems costly, i.e. to compute $\text{SupMax}K_c(\boldsymbol{x})$ for each $\boldsymbol{x}$, we need to perform set-inclusion check with all the patterns of size K. In RP-growth, on the other hand, we only have to check the weakness using the patterns in the current candidate list, whose size k is usually small. Besides, Fang et al. did not propose top-k mining, so some appropriate threshold for SupMax$K$ should be given. One interesting approach is to perform a depth-first search according to SupMax$K$ using suffix enumeration trees, while a breadth-first style is originally taken.

The original version of weakness is due to Yuan et al.'s recent framework called *most relevant explanation* (MRE) [23] for explanatory analysis of Bayesian networks. That is, weakness in this paper is the converse relation of *strong dominance* in [23], i.e. $\boldsymbol{x}'$ is weaker than $\boldsymbol{x}$ iff $\boldsymbol{x}$ strongly dominates $\boldsymbol{x}'$. However, neither branch-and-bound search nor any pruning mechanism exploiting weakness has not been explored in MRE. For their relevance score called generalized Bayes factor (GBF) $p(c \mid \boldsymbol{x})/p(c \mid \neg \boldsymbol{x})$, we can conduct branch-and-bound pruning via minimum support raising, and it is a future work to examine the tightness of GBF's upper bound. Besides, Bayardo et al. defined that a pattern $\boldsymbol{x}$ is *productive* iff the improvement $R_c(\boldsymbol{x}) - \max_{\boldsymbol{x}' \subset \boldsymbol{x}} R_c(\boldsymbol{x}')$ is positive [2], and hence non-weak patterns in this paper are also said to be productive ($R_c(\boldsymbol{x})$ is fixed to be confidence/precision $p(c \mid \boldsymbol{x})$ in the original definition).

6 Conclusion

In this paper, based on FP-growth, we proposed RP-growth, an efficient algorithm for top-k mining of discriminative patterns which are highly relevant to the class of interest. These top-k relevant patterns are considered to be manageable and more informative than frequent patterns. RP-growth conducts a branch-and-bound search using anti-monotonic upper bounds of the relevance scores such as χ^2 and F-score, and the pruning in branch-and-bound search is successfully translated to minimum support raising, a standard, easy-to-implement pruning strategy for top-k mining. Furthermore, by introducing the notion of weakness and an additional, aggressive pruning strategy based on weakness, RP-growth efficiently finds k relevant patterns of wide variety. Experimental results on text classification exhibit the efficiency and the usefulness of RP-growth. In future, it is necessary to conduct more extensive experiments for demonstrating the usefulness of top-k relevant patterns. In addition to classification tasks, we are interested in applying RP-growth to the problems such as sensitivity or explanatory analysis of Bayesian networks [13, 23], automatic labeling of the clustering results [14, 17] (in which clustering and discriminative pattern mining work complementarily, e.g. the former can give class labels to the latter), and so on.

References

[1] S. Bay and M. Pazzani. Detecting group differences: mining contrast sets. *Data Mining and Knowledge Discovery*, 5:213–246, 2001.

[2] R. Bayardo, R. Agrawal and D. Gunopulos. Constraint-based rule mining in large, dense databases. *Data Mining and Knowledge Discovery*, 4:217–240, 2000.

[3] H. Cheng, X. Yan, J. Han, and C.-W. Hsu. Discriminative frequent pattern analysis for effective classification. *Proc. of the 23rd IEEE Int'l Conf. on Data Engineering (ICDE-07)*, pages 716–725, 2007.

[4] G. Dong and J. Li. Efficient mining of emerging patterns: discovering trends and differences. *Proc. of the 5th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD-99)*, pages 43–52, 1999.

[5] H. Fan and K. Ramamohanarao. A Bayesian approach to use emerging patterns for classification. *Proc. of the 14th Australasian Database Conf. (ADC-03)*, pages 39–48, 2003.

[6] H. Fan and K. Ramamohanarao. Fast discovery and the generalization of strong jumping emerging patterns for building compact and accurate classifiers. *IEEE Trans. on Knowledge and Data Engineering*, 18:721–737, 2006.

[7] G. Fang, G. Pandey, W. Wang, M. Gupta, M. Steinbach, and V. Kumar. Mining low-support discriminative patterns from dense and high-dimensional data. *IEEE Trans. on Knowledge and Data Engineering*, 24:279–294, 2012.

[8] D. Gamberger, N. Lavrač, F. Železný, and J. Tolar. Induction of comprehensible models for gene expression datasets by subgroup discovery methodology. *J. of Biomedical Informatics*, 37:269–284, 2004.

[9] L. Geng and H. Hamilton. Interestingness measures for data mining: a survey. *ACM Computing Surveys*, 38(3):1–32, 2006.

[10] J. Han, J. Pei, and Y. Yin. Mining frequent patterns witout candidate generation. *Proc. of the 2000 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD-00)*, pages 1–12, 2000.

[11] J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining top-K frequent closed patterns without minimum support. *Proc. of the 2002 IEEE Int'l Conf. on Data Mining (ICDM-02)*, pages 211–218, 2002.

[12] N. Japkowicz and M. Shah. *Evaluating Learning Algorithms: A Classification Prespective.* Cambridge U. Press, 2011.

[13] F. V. Jensen. *An Introduction to Bayesian Networks.* UCL Press, 1996.

[14] Y. Kameya, S. Nakamura, T. Iwasaki, and T. Sato. Verbal characterization of probabilistic clusters by minimal discriminative propositions. *Prof. of the 23rd IEEE Int'l Conf. on Tools with Artificial Intelligence (ICTAI-11)*, pages 873–875, 2011.

[15] P. Kralj Novak, N. Lavrač, and G. I. Webb. Supervised descriptive rule discovery: a unifying survey of contrast set, emerging pattern and subgroup mining. *J. of Machine Learning Research*, 10:377–403, 2009.

[16] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. *Proc. of the 4th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD-98)*, pages 80–86, 1998.

[17] C. Manning, P. Raghavan, and H. Shütze. *Introduction to Information Retrieval.* Cambridge U. Press, 2008.

[18] S. Morishita. On classification and regression. *Proc. of the 1st Int'l Conf. on Discovery Science (DS-98)*, pages 40–57, 1998.

[19] S. Morishita and J. Sese. Traversing itemset lattices with statistical metric pruning. *Proc. of the 19th ACM-SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS-00)*, pages 226–236, 2000.

[20] P. Terlecki and K. Walczak. Efficient discovery of top-k minimal jumping emerging patterns. *Proc. of the 6th Int'l Conf. on Rough Sets and Current Trends in Computing (RSCTC-08)*, pages 438–447, 2008.

[21] G. Webb and S. Zhang. k-optimal rule discovery. *Data Mining and Knowledge Discovery*, 10:39–79, 2005.

[22] S. Wrobel. An algorithm for multi-relational discovery of subgroups. *Proc. of the 1st European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD-97)*, pages 78–87, 1997.

[23] C. Yuan, H. Lim, and T.-C. Lu. Most relevant explanation in Bayesian networks. *J. of Artificial Intelligence Research*, 42:309–352, 2011.

[24] A. Zimmermann and L. De Raedt. Cluster grouping: from subgroup discovery to clustering. *Machine Learning*, 77:125–159, 2009.