Discovering Interesting Patterns Through User's Interactive Feedback[*]

Dong Xin Xuehua Shen Qiaozhu Mei Jiawei Han
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL, 61801
{dongxin, xshen, qmei2, hanj}@uiuc.edu

ABSTRACT

In this paper, we study the problem of discovering interesting patterns through user's interactive feedback. We assume a set of candidate patterns (*i.e.*, frequent patterns) has already been mined. Our goal is to help a particular user effectively discover interesting patterns according to his specific interest. Without requiring a user to explicitly construct a prior knowledge to measure the interestingness of patterns, we learn the user's prior knowledge from his interactive feedback. We propose two models to represent a user's prior: the *log-linear model* and *biased belief model*. The former is designed for item-set patterns, whereas the latter is also applicable to sequential and structural patterns. To learn these models, we present a two-stage approach, *progressive shrinking* and *clustering*, to select sample patterns for feedback. The experimental results on real and synthetic data sets demonstrate the effectiveness of our approach.

Categories and Subject Descriptors: H.2.8 [Database Management]: Database Applications - Data Mining

General Terms: Algorithms

Keywords: Pattern Discovery, Interactive Feedback

1. INTRODUCTION

Discovering interesting patterns is an important task in data mining. However, a common problem in most pattern discovery algorithms is that there are too many patterns in the output while only a few of them is really interesting to a user. Moreover, the measure of interestingness is often subjective, and there is no consistent objective measure to represent user's interest. A pattern could be interesting to one user but not to another. Consider a scenario in the literature domain. There are often hundreds of papers published annually in a research area. To understand the research topics in the literature, one can mine frequently occurred terms, called *theme patterns*, from those papers. One such pattern from KDD conference papers is "associate pattern mining". A novice who wants to know the main research topics of KDD area will be interested in this pattern. However, an experienced researcher who would like to discover new emerging topics will probably not think it is an intriguing pattern at all.

Most existing pattern mining methods focus on efficiently computing patterns which satisfy a pre-specified criterion. The criterion can be simply a minimum support constraint or as complex as the unexpectedness with respect to a user-specified model, *e.g.*, the Bayesian Network [8]. In many cases, however, none of these pre-specified criteria can model the user's interestingness measure really well: the minimum support constraint is often too general to catch a user's prior knowledge and consequently too many common patterns are often generated in the results; the Bayesian network criterion requires users to construct a reasonably precise background knowledge explicitly, which is found to be difficult in many real applications.

In this paper, we propose to *discover interesting patterns through user's interactive feedback*. We assume a set of candidate patterns (*i.e.*, frequent patterns) has already been mined. Our goal is to help a particular user effectively discover interesting patterns according to his real interest. Instead of requiring the user to explicitly construct the prior knowledge precisely beforehand, we alleviate the user's burden by only asking him to rank a small set of sample patterns according to his interest.

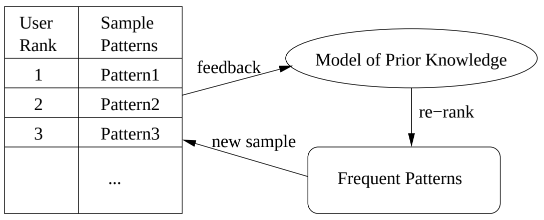

Figure 1: Discovering Interesting Patterns from Interactive Feedback

A framework of discovering interesting patterns is shown in Figure 1. Our system takes a set of candidate patterns as input. A model is created to represent a user's prior

[*]The work was supported in part by the U.S. National Science Foundation NSF IIS-03-08215/05-13678 and NSF BDI-05-15813. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

knowledge, and the entire procedure is to learn the model parameters. At each round, a small collection (*e.g.*, 10) of sample patterns are selected from the pattern collection and are asked for the user's preference. The user ranks the sample patterns, and the feedback information will be used to refine the model parameters. The system then re-ranks the patterns according to the intermediate learning result and decides which patterns to be selected for next feedback. The interaction continues for several rounds. Finally, the top-ranked patterns are output as interesting patterns.

There are two basic research questions in discovering interesting patterns through interactive feedback. First, how does the system model a user's prior knowledge? Second, how does the system select sample patterns to maximize the learning benefits? In this paper, we propose two models, the *log-linear model* and *biased belief model*, to represent user priors. The former is designed only for item-set patterns whereas the latter can also be applied to sequential and structural patterns. For effective learning from user's interactive feedback, we develop a two-stage approach, *progressive shrinking* and *clustering*, to select interesting and novel patterns for feedback. Our experimental results show that the proposed approaches are effective in discovering user-specified interesting patterns.

The rest of the paper is organized as follows. Section 2 introduces our problem formulation. The models for prior knowledge are discussed in Section 3, followed by learning procedure from the user's interactive feedback in Section 4. We present the experimental results in Section 5. Section 6 concludes our work.

2. PROBLEM STATEMENT

We first discuss the formulations of interestingness and user feedback and then present the problem statement.

2.1 Interestingness Measure

We assume that a set of frequent patterns $\mathcal{P}$ has been mined and forms the input of our system. Each pattern $P \in \mathcal{P}$ consists of the composition of P (*i.e.*, the item-set, sequential items or graphical structure of P) and a set of transaction IDs which contain P.

The interestingness of a pattern P is evaluated by a *subjective* measure [11] that computes the difference between the observed frequency $f_o(P)$ and the expected frequency $f_e(P)$. Here frequency is the proportion of transactions which contain the pattern. We model the subjective interestingness measure using two components: a model of prior knowledge and a ranking function. The model of prior knowledge M is used to compute the expected frequency of P as follows:

$$f_e(P) = M(P, \theta)$$

i.e., M takes a pattern $P \in \mathcal{P}$ and a model parameter $\theta \in \Theta$ as inputs and returns $f_e(P)$ as the expected frequency of P.

The ranking function R is of the form:

$$R(f_o(P), f_e(P)) \to \mathbb{R}$$

which returns the degree of interestingness of the pattern according to two frequencies.

2.2 From Feedback to Constraints

In order to learn to rank patterns according to the subjective measure of user interestingness, we ask for user feedback interactively. A user feedback is formulated as a constraint on the model to be learned. We iteratively present the user a set of k sample patterns $\{P_1, P_2, \ldots, P_k\}$ and ask for the user's preferences. P_i is ranked above P_j (*i.e.*, $P_i >_f P_j$) if the user judges P_i is more interesting. A user can provide a fully ordered or partially ordered feedback. Every pair of relative order $P_i >_f P_j$ is formulated as a constraint as follows:

$$R(f_o(P_i), f_e(P_i) > R(f_o(P_j), f_e(P_j))$$

2.3 Problem Statement

The problem of *discovering interesting patterns through interactive feedback* can be stated as follows: Given a set of patterns, the system ranks patterns according to a user-specific interestingness measure, and at the same time minimizes the user's efforts in providing feedback.

The user-specific interestingness measure consists of a ranking function and a model of prior knowledge. Generally, it is relatively simple to determine the ranking function R: A user can either select a system-default function or define one by his own. However, it is difficult to determine the model of prior knowledge M and its parameters θ. In this paper, we mainly focus on the models and the methods of learning parameters from user feedback. Without loss of generality, we use the following *log-linear* ranking function:

$$R(f_o(P), f_e(P)) = \log f_o(P) - \log f_e(P) \qquad (1)$$

The above function measures the degree of how $f_o(P)$ is larger than $f_e(P)$. Our framework can accept other types of ranking functions such as *linear* (e.g., $f_o(P) - f_e(P)$) or *nonlinear* (e.g., $(f_o(P) - f_e(P))^2$). We will discuss the details in Section 3.3.

Given the ranking function, the problem of discovering user-specific interesting patterns is equivalent to learning the model of the user's prior knowledge.

3. MODELING PRIOR KNOWLEDGE

We discuss two models to represent user's prior knowledge. One is *log-linear model* which works for item-set patterns only, and the other is *biased belief model* which can also be applied to structural patterns.

3.1 Log-linear Model

We introduce the fully independent *log-linear model* and its constrained formulation using ranking SVM[9].

3.1.1 Log-linear Model

Statisticians have been using *log-linear model* [2] to study the frequency of an item-set comprising n-items: $f(x_1, x_2, \ldots, x_n)$. More formally, the saturated *log-linear model* for f is expressed as:

$$\log f(x_1, \ldots, x_n) = u + \sum_j u(x_j) + \sum_{j \neq k} u(x_j, x_k) + \ldots + u(x_1, \ldots, x_n) \qquad (2)$$

where the u summands capture the interactions between the items, and x_j is chosen from $\{i_j, \overline{i_j}\}$. For example, $u(\overline{i_1}, i_2)$ denotes the interaction between $\overline{i_1}$ and i_2.

The saturated *log-linear model* depicted above is the most general model for n items. Simpler *log-linear model* reduces the number of parameters by specifying certain interaction effects to be zero, which can typically be interpreted as a form of *independence* between the corresponding dimensions. The *fully independent model* assumes the values of u over two or more items vanish. Furthermore, since we are not interested in the generalized patterns, we do not need the value of $u(\overline{i_j})$. As a result, there are only $n+1$ variables: u and $u(i_j)$ $(j = 1, \ldots, n)$. We simplify the notation $u(i_j)$ as u_j. Given an item-set pattern $P = (i_1, \ldots, i_s)$, its expected frequency by a *fully independent log-linear model* is:

$$\log f_e(P) = u + \sum_{j=1,\ldots,s} u_j$$

3.1.2 *Constrained Formulation*

The relative ordering $P_1 >_f P_2$ provided by the user feedback gives the constraint:

$$\log f_o(P_1) - \log f_e(P_1) > \log f_o(P_2) - \log f_e(P_2)$$

Here, $\log f_o(P_1)$ and $\log f_o(P_2)$ are constants. $\log f_e(P_1)$ and $\log f_e(P_2)$ are both linear combination of variables u and u_j $(j = 1, \ldots, n)$. Basically, we want to learn these variables so that the number of violated constraints is minimized. This problem is known as NP-hard [3] and a practical approach is to use ranking SVM formulations [9].

We introduce a weighting vector as learning variables:

$$w = [v, -u, -u_1, \ldots, -u_n]^T$$

where $u, u_1, \ldots, u_n$ are *log-linear model* parameters and v is a scale variable associated to $\log f_o(P)$. Each pattern P can be represented by a vector:

$$v(P) = [\log f_o(P), x_1, \ldots, x_n]^T$$

where $x_j = 1$ if and only if $i_j \in P$. The feedback constraint can be rewritten as:

$$w^T \cdot v(P_1) > w^T \cdot v(P_2)$$

The ranking SVM formulates the problem as the following constrained optimization problem:

$$\begin{aligned} \textit{minimize}: \ & \|w\|^2 + C \sum_{P_i >_f P_j} \xi_{i,j} \\ \textit{subject to}: \ & w^T \cdot v(P_i) - w^T \cdot v(P_j) > 1 - \xi_{i,j} \\ & \xi_{i,j} \geq 0 \qquad (\forall\ P_i >_f P_j) \end{aligned}$$

By minimizing $\|w\|^2$, the formulation tries to maximize the generalizability of the model. C is the parameter which controls the trade-off between the learning error and the generalizability. Algorithms have been developed to solve the problem efficiently. Detailed introductions can be found in [9].

3.2 Biased Belief Model

The *log-linear model* is an item-set based model which cannot be applied to more complicated patterns (*e.g.*, sequential patterns or structural patterns). Here we introduce a more general *biased belief model*.

3.2.1 *Belief on Transactions*

Instead of modelling user prior on patterns directly, we model user prior on data. The intuition is that the expectation of a pattern is determined by user's belief in the underlining data. If a user has high belief in the subset of the data that supports the pattern, the user will have high expectation of this pattern; on the other hand, if a user knows little about the data, the expectation will be low.

The *biased belief model* derives the expected frequency of a pattern from the belief in transactions containing the pattern. To measure user's belief in the data, we assign a belief probability to each transaction. A higher probability means the user is more familiar with this transaction, whereas a lower one indicates that this transaction is novel to the user. Let us assume that each transaction is independent. The set of transaction data forms a *multiple-Bernoulli distribution*, where each binary trial corresponds to the event whether the user knows the transaction or not. Therefore, the user's prior knowledge can be represented by a vector $[p_1, \ldots, p_m]$, where p_k is the belief probability for transaction k, and m is the total number of transactions. Given a pattern P, the value of $f_e(P)$ is proportional to the expected number of occurrences of P:

$$\sum_{k=1,\ldots,m} p_k \times x_k(P),$$

where $x_k(P) = 1$ if transaction k contains pattern P, otherwise, it is 0.

The learning task is to determine the parameters p_k. Given a user feedback $P_i >_f P_j$, we have:

$$\log f_o(P_i) - \log f_e(P_i) > \log f_o(P_j) - \log f_e(P_j)$$

that is,

$$\frac{\sum_{k=1,\ldots,m} p_k \times x_k(P_j)}{f_o(P_j)} > \frac{\sum_{k=1,\ldots,m} p_k \times x_k(P_i)}{f_o(P_i)}$$

Similar to our formulation in *log-linear model*, we introduce a weighting vector:

$$w = [p_1, p_2, \ldots, p_m]^T$$

and each pattern P is represented by a vector:

$$v(P) = \frac{1}{f_o(P)} [x_1(P), \ldots, x_m(P)]^T$$

We again use the ranking SVM method and formulate the optimization problem as follows:

$$\begin{aligned} \textit{minimize}: \ & \|w\|^2 + C \sum_{P_i >_f P_j} \xi_{i,j} \\ \textit{subject to}: \ & w^T \cdot v(P_j) - w^T \cdot v(P_i) > 1 - \xi_{i,j} \\ & \xi_{i,j} \geq 0 \qquad (\forall\ P_i >_f P_j) \\ & w_k \geq 0 \qquad (k = 1, \ldots, m) \end{aligned}$$

The additional constraints $w_k \geq 0$ indicate that the belief probability for each transaction cannot be less than 0. We do not apply the constraint $w_k \leq 1$ here because all w_k's can be scaled to no larger than 1.

3.2.2 *Model Relaxation*

Unfortunately, the constrained formulation presented above does not give satisfactory results in experiments. One important reason is that we assume the transactions in user's prior are similar to the observed transactions, which is not always true. Consider an extreme situation where an observed data set contains a set of transactions supporting both patterns P_1 and P_2, and also contains another set of transactions supporting P_2 only. But there is no transaction containing P_1 only. A user may have high expectation of pattern P_1 but not P_2. The constraints $w_k \geq 0$ are too rigid so that the expectation of P_2 cannot be less than P_1. This violates the user's belief. To improve the learning power of the model, we remove the constraints $w_k \geq 0$ to allow some transactions to be assigned to negative weights. In this example, the transactions containing pattern P_2 only will have *negative* weights so that they can downgrade the high belief (of P_2) brought by the transactions containing both P_1 and P_2. Hence the model is more amenable to fit in the user's prior knowledge.

3.3 Discussion of Ranking Functions

With respect to the ranking function, we have demonstrated both models using a *log-linear* formulation. The *biased belief model* can also take the *linear* ranking functions (*e.g.*, $f_o(P) - f_e(P)$). Assigning $v(P) = [f_o(P), -x_1(P), \ldots, -x_m(P)]$ and $w = [u, p_1, p_2, \ldots, p_m]$ (u is a scale variable) leads to a linear ranking SVM formulation. Extending to nonlinear constraint formulations, both models can use arbitrary ranking functions. Let $R(f_o(P), f_e(P)) = K(v(P), w)$, if K satisfies Mercer's condition [4], it can be used as SVM kernel; otherwise, the formulated nonlinear problem can be solved by existing mathematical programming package (*e.g.*, SNOPT [5]). Recent progress on nonlinear programming [5] shows that problems up to $40,000$ variables and constraints are solvable within reasonable time.

4. LEARNING FROM FEEDBACK

In this section, we discuss the strategy to select sample patterns for effective learning. The importance of selecting best samples for user feedback has been recognized by some previous work [13, 10]. Ideally, the system should collaborate with the user in the whole interactive process to improve the ranking accuracy and reduce the number of interactions. In this section, we first discuss the selection criteria and then propose our sample selection method. Finally, we present the complete algorithm as a summary.

4.1 Selection Criteria

We first propose two criteria to select sample patterns. The first is that the selected sample patterns should not be redundant to each other. We say there is redundancy between two patterns if they are close in both pattern composition (*i.e.*, the set of items) and frequency. Since redundant patterns naturally rank close to each other, presenting redundant patterns for feedback does not maximize the learning benefit and increases user overhead. The second criterion is that the selected patterns should help to learn a user's knowledge about the ranking of interesting patterns since a user generally has preference over higher ranked patterns, and the relative ranking among uninteresting patterns is not important.

4.2 Progressive Shrinking and Clustering

Our two criteria are also discussed in active feedback with information retrieval [10], where the authors proposed to first cluster top-N documents, and then select the k centroids from each cluster as feedback samples. This method was shown to be effective in document query. However, the method cannot be directly applied here. This is because in information retrieval, the documents ranking is query guided. Interesting results are generally congregated around the query, and the initial top-N results are good enough to guarantee that most interesting documents are included. While in pattern ranking, the interesting patterns scatter over the whole pattern collections. At the beginning, the system has no idea which parts of patterns are interesting. Concentrating the initial top-N results will delay the discovery of interesting patterns.

We extend the method as follows. Instead of fixing the number of N for clustering, we adopt a two-stage approach, *progressive shrinking* and *clustering*, to select sample patterns at each iteration. We define a shrinking ratio α ($0 < \alpha < 1$). At the beginning, the candidate set size N is equal to the size of the complete pattern collection. It gradually decreases to focus more on the highly ranked patterns. At each iteration, we update $N = \alpha N$, and the pattern set for clustering is the top-N patterns w.r.t. the current ranking.

Suppose a user agrees to examine k patterns at each iteration, we then cluster these top-N patterns into k clusters. We use the Jaccard distance [7] for clustering: given a pattern P_1 and P_2, the distance between P_1 and P_2 is defined as:

$$D(P_1, P_2) = 1 - \frac{|T(P_1) \cap T(P_2)|}{|T(P_1) \cup T(P_2)|}$$

where $T(P)$ is the set of transactions which contain pattern P. This measure is applicable to all kinds of patterns mined from transaction database, and also a valid distance metric [12].

We adopt a clustering criterion of minimizing the maximum distance of each pattern to the nearest sample pattern. This is a typical k-center problem in graph algorithms and approximatable within 2 using a greedy algorithm. The algorithm first picks an arbitrary pattern. While the number of picked patterns is less than k, the algorithm continues to pick a pattern which has the maximal distance to the nearest picked patterns. The complexity of this algorithm is $O(kn)$. Due to the limited space, we do not introduce the details here, interested readers can refer to [6].

We call this clustering procedure *online-clustering*. To improve the scalability of this step, we perform *micro-clustering* on the input pattern set at the beginning and only keep the representative patterns in each micro-cluster. Basically, a pattern P is absorbed by a representative pattern P_r if P is a subpattern of P_r and $D(P, P_r) < \epsilon$. Here ϵ is a small value (*e.g.*, 0.1). We start with a subset of maximal patterns (a pattern is maximal if its super pattern is not in the collection), and remove all the patterns that can be absorbed by one of those maximal patterns. Since maximal patterns contain more information about the pattern content, they are kept as representative patterns. This procedure repeats on the remaining patterns until there is no pattern left.

4.3 The Complete Algorithm

We have discussed the user prior models and the sample selection method. As a summary, we outline the complete algorithm for pattern ranking with user's interactive feedback in Algorithm 1.

The algorithm takes the entire collection of patterns as input. User can also specify the number of iterations he/she would like to provide feedback and how many patterns he/she would like to judge at each round. The algorithm works as follows. First, a micro-clustering is conducted on the input patterns set $\mathcal{P}$, and only the representative patterns in $\mathcal{P}_r$ are used thereafter. Variable N indicates the number of top

Algorithm 1 Discover Interesting Pattern

```
Input: A set of $n$ patterns, $\mathcal{P}$
       Number of sample patterns for feedback, $k$
       Number of iterations of feedback, $iter$
       Shrinking ratio, $\alpha$
       Micro Clustering Parameter, $\epsilon$
Output: Ranked Pattern List

1: $\mathcal{P}_r$ = Micro-Clustering($\mathcal{P}$, $\epsilon$);
2: $N = |\mathcal{P}_r|$;
3: for $(i = 0; i < iter; i++)$
4:     $\mathcal{C}$ = top-$N$ patterns in $\mathcal{P}_r$;
5:     Online-Clustering of $\mathcal{C}$;
6:     Present $k$ sample patterns to user for feedback;
7:     Formulate constraints according to the user feedback;
8:     Refining the model parameters by Ranking SVM;
9:     Re-Rank Patterns in $\mathcal{P}_r$ with the refined model;
10:    Remove the $k$ selected patterns from $\mathcal{P}_r$;
11:    $N = \alpha N$;
12: Rank Patterns in $\mathcal{P}$ with the learned model;
13: return
```

patterns (with the current ranking) to be considered for sample selection (line 4). This value is progressively reduced by a factor α (line 11). Line 5 conducts the online-clustering to find k sample patterns for user feedback (see previous subsection). Line 6 asks user to provide feedback on the sample patterns. Lines 7 and 8 formulate the feedback as constrained problem, which is further solved by the ranking SVM. Note the constraints from the previous feedback are accumulated in the formulation. The model parameters are updated by the solutions of the ranking SVM and the patterns are re-ranked according to the updated model. Line 10 removes the selected sample patterns so that they will not be selected again for user judgment. After the algorithm repeats *iter* times, the ranking results are output.

5. EXPERIMENTAL RESULTS

5.1 Experiment Setting

We set up an experiment environment to simulate user feedback. The intuition is that a user may consider a pattern interesting because he was previously exposed to data that has different flavor. Thus the user's prior knowledge can be approximated by a background data set, and the expected frequency of a pattern is the frequency computed from this background data.

Using this methodology, we partition each data set in our experiments into two subsets: one for observed data and the other for background data. The background data is used to simulate a user's prior knowledge. For each pattern P, $f_e(P)$ is the smoothed frequency in the background data. The background data provides a target pattern ranking which is used to simulate user feedback. Since our goal is to discover interesting patterns, our accuracy measure favors higher ranked patterns. Let $top^l(k)$ be the top-k% results reported by the ranking *learned* from the interactive feedback and $top^t(k)$ be the top-k% results in the *target* ranking constructed by our simulation. The accuracy measure is defined as follows.

$$Accurary = \frac{top^t(k) \cap top^l(k)}{k}$$

We assume that the user provides fully ordered feedback. To minimize the user's effort, at most 10 patterns are presented to the user at each iteration and at most 10 iterations are allowed. Since the number of constraints is upper bounded by 450, the scalability is not an important issue here because most constrained problems can be solved by ranking SVM within seconds. In all experiments, we use *closed* item-set (or sequential) patterns as input and set $\epsilon = 0.1$ for micro-clustering and $\alpha = 0.1$ for shrinking ratio. The ranking SVM [9] is called with its default parameter setting.

5.2 Experimental Results

We conduct a series of experiments to examine the ranking accuracy w.r.t. different criteria: item-set patterns, sequential patterns and different sample selection strategies.

5.2.1 Item-set Patterns

Using the simulation environment discussed above, our first task is to examine the ranking accuracy on the item-set patterns. The first experiment is run on a real data set *pumsb* [1], which consists of $49,046$ transactions with $2,113$ distinct items. The average length of transactions is 74. We extract the first $1,000$ transactions from the original data set as observed data and the rest transactions are used as background data. By setting minimum support as 88%, we mined $8,234$ closed frequent item-sets. The micro-clustering with $\epsilon = 0.1$ reduces them to 769 representative patterns. The accuracy of top 10% results of the *log-linear model* and *biased belief model* with different feedback size (*i.e.*, 5 and 10) is shown in Figure 2. We observed that both models achieve higher than 80% (70%) accuracy with feedback size 10 (5).

5.2.2 Sequential Patterns

In this subsection, we examine the ranking performance on *sequential patterns*. We use the 2001-2002 KDD paper abstracts as the observed data and the 1999-2000 KDD paper abstracts as the background data. All abstracts are decomposed into sentences. There are $1,609$ sentences in 2001-2002 data set. Using minimum support 0.8%, we mine 967 closed sequential patterns. The *log-linear model* is used by treating the sequential patterns as item-set patterns. The accuracy of the top k percent ($k = 1, \ldots, 10$) ranking after 10 iterations is shown in Figure 3.

Not surprisingly, the *biased belief model* works better than the *log-linear model*. We further compute the *limiting behavior* of both models by assuming that the *fully ordered feedback* on all 967 patterns is available for constraint formulation. This can be seen as the upper bound case of the ranking accuracy. An interesting observation is the ranking accuracy of *log-linear model* with the fully ordered feedback is even worse than that with 10 iterations. This indicates that treating sequential patterns as item-set patterns actually introduces noise information to the *log-linear model*. For example, sequential patterns *ab* and *ba* are considered same as item-set patterns. However, they might have totally different frequencies. Hence, the fully ordered feedback does not help to learn a finer *log-linear model*. On the other hand, the *biased belief model* gets 80% for top 10 percent rankings with fully ordered feedback.

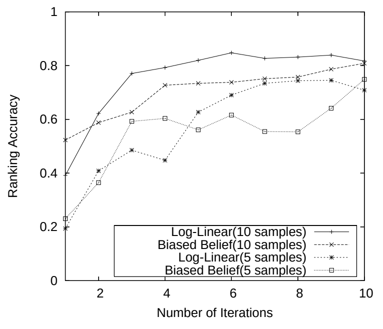

Figure 2: Top-10% ranking accuracy: k iterations.

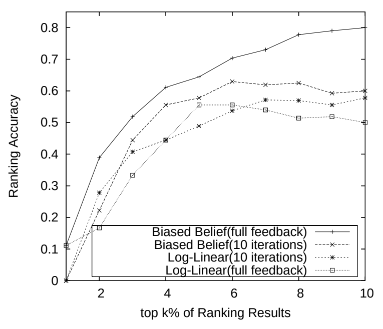

Figure 3: Top-k% ranking accuracy: 10 iterations.

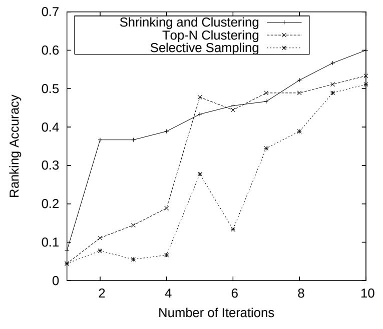

Figure 4: Top-10% ranking accuracy: sample pattern selection.

5.2.3 Sample Patterns Selection

Finally, we test the learning effectiveness by various strategies to select sample patterns for feedback. We compare our proposed strategy with the selective sampling approach [13] and the top-N clustering approach [10]. We use the same KDD abstract data set in the previous experiment and set the feedback size as 10. Since selective sampling does not consider pattern redundancy, it is very likely that the method will pick 10 very similar patterns. To make a fair comparison, we first cluster representative patterns into 100 groups and select 10 representative patterns using selective sampling criteria. For top-N clustering approach, the value of N is set as 100, which is 10% of the total patterns. The shrink ratio of our method is set as 0.1. As shown in Figure 4, the selective sampling approach is comparatively worse because it does not favor higher ranked patterns. The accuracy of the top-N clustering approach is worse than that of the shrinking and clustering method until the 5-*th* iteration. This confirms that our proposed approach is more effective to locate interesting patterns, and the top-N clustering approach may miss some interesting patterns in the early iterations.

6. CONCLUSIONS

In many data mining approaches, the algorithm or the measure is pre-designed and the user accepts the results passively. This paper introduces a new problem setting where the mining system interacts with the user. To discover interesting patterns for a particular user, we propose a framework to learn user's prior knowledge from interactive feedback. We study two model formulations, the *log-linear model* and the *biased belief model*, and discuss the strategy to select sample patterns for user feedback. The performance of proposed approaches are tested on various data sets. Experiment results show that both models are able to learn user's background knowledge. Moreover, the *biased belief model* also works for sequential and structural patterns.

The current work can be extended in several ways. First, we will study new models for sparse data set. As seen from the experiments, data sparsity may cause learning difficulty. Second, we will further exploit active feedback strategies to maximize the learning benefit. Finally, we will explore the interactive feedback on other personalized data mining applications such as user-specific summarization and clustering.

7. REFERENCES

[1] R. Bayardo, B. Goethals, and M. Zaki. Fimi 2004 workshop. *Proc. ICDM Workshop on Frequent Itemset Mining Implementations (FIMI)*, 2004.

[2] Y. M. M. Bishop, S. E. Fienberg, and P. W. Holland. *Discrete Multivariate Analysis.* The MIT Press, 1975.

[3] W. W. Cohen, R. R. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

[4] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods.* Cambridge Univ. Press, 2000.

[5] P. E. Gill, W. Murray, and M. A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM Review*, 47:99–131, 2005.

[6] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Comput. Sci.*, 38:293–306, 1985.

[7] A. Jain and R. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.

[8] S. Jaroszewicz and T. Scheffer. Fast discovery of unexpected patterns in data, relative to a bayesian network. *Proc. of 2005 ACM Int. Conf. on Knowledge Discovery in Databases (KDD'05)*, pages 118 – 127, 2005.

[9] T. Joachims. Optimizing search engines using clickthough data. *Proc. of 2002 ACM Int. Conf. on Knowledge Discovery in Databases (KDD'02)*, pages 133 – 142, 2002.

[10] X. Shen and C. Zhai. Active feedback in ad hoc information retrieval. *Proc. of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'05)*, pages 59 – 66, 2005.

[11] A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Trans. on Knowledge and Data Eng.*, 8:970–974, 1996.

[12] D. Xin, J. Han, X. Yan, and H. Cheng. Mining compressed frequent-pattern sets. *Proc. of 31st International Conference on Very Large Data Bases (VLDB'99)*, pages 709 – 720, 2005 .

[13] H. Yu. Svm selective sampling for ranking with application to data retrieval. *Proc. of 2005 ACM Int. Conf. on Knowledge Discovery in Databases (KDD'05)*, paegs 354 – 363, 2005.