SFP-Rank: Significant Frequent Pattern Analysis for Effective Ranking

Yuanfeng Song, Wilfred Ng, Kenneth Leung, and Qiong Fang

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Hong Kong,
songyfxjtu@gmail.com,{wilfred,kwtleung,fang}@cse.ust.hk

Abstract. Ranking documents in terms of their relevance to a given query is fundamental to many real-life applications such as information retrieval and recommendation systems. Extensive study in these application domains has given rise to the development of many efficient ranking models. While most existing research focuses on developing *Learning To Rank* (LTR) models, the quality of the training features, which plays an important role in ranking performance, has not been fully studied. Thus, we propose a new approach that discovers effective features for the LTR problem.

In this paper, we present a theoretical analysis on which frequent patterns are potentially effective for improving the performance of LTR, and then propose an efficient method that selects frequent patterns for LTR. First, we define a new criterion, namely *feature significance* (or simply *significance*). Specifically, we use each feature's value to rank the training instances, and define the ranking effectiveness in terms of a performance measure as the significance of the feature. We show that the significance of an infrequent pattern is limited by using formal connection between pattern support and its significance. Then, we propose a methodology that sets the support value when performing frequent pattern mining. Finally, since frequent patterns are not equally effective for LTR, we further provide a coverage based significant pattern generation algorithm to discover effective patterns, and propose a new ranking approach called *SFP-Rank* (**S**ignificant **F**requent **P**attern based **Rank**ing), in which the ranking model is built upon the original features as well as the significant frequent patterns. Our experiments confirm that, by incorporating significant frequent patterns to train the ranking model, the performance of the ranking model can be substantially improved.

Keywords: Learning to rank; frequent patterns; feature selection; combined features; ranking performance

1 Introduction

Ranking is a well-recognized problem in the research area of *information retrieval*, since establishing an effective ranking model is central in many applica-

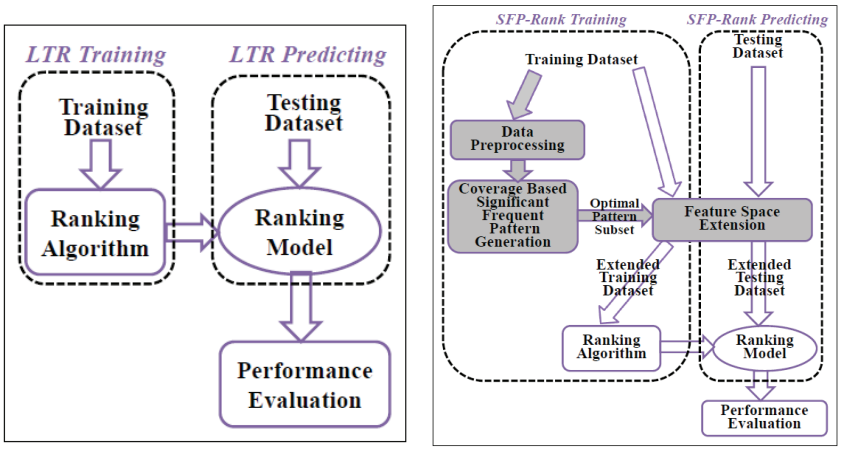
[0] A preliminary version of this paper appears as [33].

tions, such as advertising strategies, document retrieval systems, recommender systems, and many others [23]. For example, in a document retrieval system, given a query, there are usually a large number of documents that satisfy the query conditions. It is thus necessary to further rank documents according to their relevance to the query, in order that users are able to obtain the most relevant results.

An effective ranking method that guarantees the retrieval quality is significant for information retrieval systems. The related study has attracted a lot of researchers' attention in recent years [22, 5, 24, 30, 32]. Empirical ranking models, like vector space models and probabilistic models, have been applied to solve the ranking problem. However, existing models usually suffer high cost for tuning their parameters. Other advanced approaches, like RankSVM [22] and RankNet [5], have been derived from machine learning techniques, to automatically learn effective ranking functions, and they are generally regarded as the *learning to rank (LTR)* methods. The LTR methods solve the ranking problem in the following way. First, they take a training dataset as the input, which consists of the records expressed in a triplet $\langle q, \boldsymbol{d}, y\rangle$ with q as the query, $\boldsymbol{d}$ as the document (represented as a vector of features), and y as the known relevance score of $\boldsymbol{d}$ to q, and, based on the training dataset, a ranking model is then constructed. The testing dataset contains records which take the same form as those in the training dataset, except that the relevance scores y are unknown. Then, the ranking model is applied to the testing dataset to estimate the relevance score of a record. Finally, the records in the testing dataset are sorted in terms of their estimated relevance scores. By representing the documents as a large number of features and making use of advanced machine learning techniques, current LTR methods like RankSVM and RankNet achieve good performance for the ranking problem compared to those empirical ranking methods.

While most of the current research has been done to design and develop effective ranking models as a fundamental part in LTR methods, not many studies were carried out to improve the quality of the document features, which have a high impact on ranking quality. Besides the effectiveness of training the ranking models, the performance of the ranking models is also highly related to the quality of the features used in the ranking. The studies in [33] show that, by incorporating a set of properly selected frequent patterns as features, the performance of the ranking model such as RankSVM can be substantially improved. Accordingly, a frequent pattern based ranking approach called FP-Rank was proposed.

In this paper, we propose a new method that improves the quality of the ranking features, which eventually improves the accuracy of a ranking method. Basically, we first define the notion of *feature significance* (or simply *significance*) that measures the effectiveness of only one single feature for training ranking model. Significance is based on the Kendall Tau distance [13] between the ranked list generated by the model with one single feature, and the ground truth ranked list. Compared with conventional performance metrics [16], this criterion has the



(a) Conventional LTR Approach (b) SFP-Rank Approach

Fig. 1. Convectional LTR Approach vs. SFP-Rank Approach

advantage of handling ties in the ranked list as well as accurately reflecting the effectiveness of a feature.

A challenge in our approach is to handle combined features, since a large portion of frequent patterns used in ranking are combined features. The combined features have the ability to reflect more semantics of the data. For example, when extracting features to represent documents in information retrieval system, using phrases (combined features) can accurately differentiate documents compared to only using single words (single features) extracted from the documents. Although combined features are useful for LTR, the number of combined features is exponential with respect to the original features. To tackle this problem, we provide a theoretical analysis showing that the significance of a pattern with low support is limited due to the pattern's limited coverage on the dataset. This fundamental result connects a pattern's frequency with its significance and provides a means to set the support value for mining frequent patterns. Our approach is then able to filter out a large amount of insignificant patterns in the mining process.

Since frequent patterns are generated solely based on frequency, not every frequent pattern is equally significant for LTR. Furthermore, including lots of frequent patterns in the model training not only increases the model training time, but also degrades the ranking performance. Thus, we adopt a coverage based significant frequent pattern generation approach to solve this problem. Based on the connection between a pattern's frequency and its significance, we filter out all the insignificant patterns by setting some appropriate support. Intuitively, we recursively find the significant patterns from the uncovered instances, until we have enough patterns to cover all the training instances.

To illustrate the underlying techniques and the main issues clearly in this work, we depict our proposed SFP approach in Figure 1(b) and also the conventional LTR approach in Figure 1(a). After data preprocessing, the coverage based significant frequent pattern generation method is adopted to generate significant frequent patterns. The generated patterns are used as new features to extend the feature space of the original data. Finally, the LTR model is trained based on the extended feature space.

In summary, our contributions concerning our proposed *SFP-Rank* approach are as follows:

- We define *feature significance* as a new criterion for evaluating the feature used in the LTR problem. This criterion enjoys the benefits of handling ties in the ranked list as well as accurately reflecting a feature's effectiveness.
- We present a formal analysis about the relationship between a pattern's significance and its frequency and show that frequent patterns have the potential to improve the performance of LTR. We also develop a method that sets the support threshold of frequent pattern mining.
- We propose a ranking approach called *SFP-Rank*. It includes a coverage based significant pattern generation algorithm that efficiently discovers significant frequent patterns for LTR.
- We evaluate our proposed algorithms. By incorporating the selected patterns as new features for ranking, the ranking performance of commonly used LTR models, such as RankSVM, is shown to be greatly improved.
- We apply *SFP-Rank* in a scenario of search engine query suggestion ranking, and show that *SFP-Rank* is more effective in identifying useful suggestion candidates for search refinement compared to the RankSVM baseline.

The rest of this paper is organized as follows. The notations and basic concepts related to *SFP-Rank* are introduced in Section 2. In Section 3, the framework *SFP-Rank* is presented. Extensive experiments on real datasets have been conducted in Section 4. A real case application example of *SFP-Rank* is given in Section 5. The related work is discussed in Section 6, followed by the conclusions given in Section 7.

2 Preliminaries

In this section, we introduce the notations and basic concepts that are used throughout the paper.

Let $\boldsymbol{B} = \{f_1, \ldots, f_m\}$ be a set of features, where each feature f_i is associated with a set of values R_i. A pattern α is a subset of features in B where each feature f_i takes a specific value in R_i. For example, suppose we have a set of features $\boldsymbol{B} = \{f_1, f_2, f_3\}$, and $R_1 = \{a, b\}$, $R_2 = \{x, y, z\}$ and $R_3 = \{u, v\}$. $\langle f_1 = a, f_3 = u\rangle$ is a pattern, which includes two features f_1 and f_3 with their respective values a and u. Further, there are two kinds of patterns as follows:

- Patterns with single features (i.e. pattern length = 1).

- Patterns with combined features (i.e. pattern length > 1).

We denote by $\mathcal{D}$ the training dataset. Each record in $\mathcal{D}$ is a triplet of $\langle q, \boldsymbol{d}, y\rangle$, where q is a query, $\boldsymbol{d}$ is a document, and y is the relevance score of the document $\boldsymbol{d}$ to the query q. A document $\boldsymbol{d}$ is represented by a vector $\langle r_1, r_2, r_3, ..., r_m\rangle$, where $r_i \in R_i$. The relevance score y is an integer in the range $[0, K]$, where 0 means no relevance between the query and the document and the (maximum) value K means "maximal" relevance. Given a pattern α, $\mathcal{D}^\alpha \subseteq \mathcal{D}$ is the set of records such that the feature values are equal to the corresponding ones appearing in α. Continuing the above example, if we have a record instance $\langle q = q_0, \langle r_1 = a, r_2 = x, r_3 = u\rangle, y = 0\rangle$, the record is said to belong to $\mathcal{D}^\alpha$ with $\alpha = \langle f_1 = a, f_3 = u\rangle$. Given a frequency threshold θ_0, a pattern α is said to be *frequent* if its frequency $\theta_\alpha = \frac{\|D^\alpha\|}{\|D\|} \geq \theta_0$. We use F to denote a set of frequent patterns where θ_0 is satisfied. A pattern α is a *closed* frequent pattern in a data set $\mathcal{D}$ if α is frequent in $\mathcal{D}$ and there exists no proper super-pattern β such that β has the same support as α in $\mathcal{D}$ (cf. [34]).

We now discuss how to evaluate the effectiveness of a pattern when it is used as the single feature to train ranking models. We establish the notion of significance as follows.

Feature Significance *Feature significance* (or simply *Significance*), denoted by $S(\alpha)$, is defined to gauge the effectiveness of a feature α for the LTR model training. To calculate $S(\alpha)$, we solely use this feature to train a ranking model and then use this model to predict a ranked list. The distance between the predicted ranked list and the ground truth reflects α's effectiveness for the ranking problem.

Now, we provide a significance evaluation method based on Kendall Tau ranking distance in order to illustrate the idea, since Kendall Tau ranking distance is widely-used in measuring the distance between ranked lists [16]. However, the evaluation method is flexible, in the sense that other distance measures such as Spearman's footrule, rank correlation can also be chosen.

$$S(\alpha) = \frac{\sum_{q \in Q} \tau(l_q^\alpha, g_q)}{\|Q\|}, \tag{1}$$

where g_q is the ground truth ranked list given by a query q, and l_q^α is the ranked list on query q given by the model trained using feature α. $S(\alpha)$ is the overall significance values of a pattern, and is determined by averaging each query's Kendall tau value on the query set Q. For each query, the Kendall tau value is given by:

$$\tau(l_q^\alpha, g_q) = \frac{n_c(l_q^\alpha, g_q) - n_d(l_q^\alpha, g_q)}{\sqrt{n(q) - n_t(l_q^\alpha)}\sqrt{n(q) - n_t(g_q)}}, \tag{2}$$

where

$$n_c(l_q^\alpha, g_q) = \left\|\text{Concordant pairs between } l_q^\alpha \text{ and } g_q\right\|,$$

$$n_d(l_q^\alpha, g_q) = \left\|\text{Discordant pairs between } l_q^\alpha \text{ and } g_q\right\|,$$

$$n(q) = \frac{\|D_q\|\left(\|D_q\|-1\right)}{2},$$

$$n_t(l_q^\alpha) = \sum_i \frac{t_i(l_q^\alpha)\left(t_i(l_q^\alpha)-1\right)}{2},$$

$$t_i(l_q^\alpha) = \left\|\text{Tied value in } i^{th} \text{ ties group}(\alpha\text{'s predicted list } l_q^\alpha)\right\|$$

and, D_q and $\|A\|$ denote respectively the set of documents retrieved using query q and the size of A.

Ranking Model Evaluation Criteria Next, we introduce the evaluation criteria, which are commonly used parameters [3].

- *Discounted Cumulative Gain (DCG) and Normalized Discounted Cumulative Gain (NDCG).*
  For a query q, a ranking method returns a document list $l_q = [\boldsymbol{d}_1, \ldots \boldsymbol{d}_t]$ of length t, in which the documents are sorted in terms of the estimated document relevance to q. Suppose y_i is the ground-truth relevance score of document d_i. The DCG score of the ranked list l_q at n, denoted by $DCG@n(l_q)$, measures the ranking accuracy of the top-n documents in l_q, and is given by:

$$DCG@n(l_q) = \sum_{i=1}^{n} c(i)(2^{y_i} - 1). \tag{3}$$

  In Formula (3), $c(i)$ is a rank-decaying function, and a widely-used one is defined as

$$c(i) = \begin{cases} \frac{1}{log(1+i)} & i < L, \\ 0 & i \geq L, \end{cases}$$

  where L is called the "*truncation level*", and it captures the fact that the quality of a ranked list is mainly determined by the order of the top results. The ground-truth ranked list of documents, denoted as g_q, can be obtained by sorting the documents according to their ground-truth relevance scores. The $NDCG$ score of the ranked list l_q at n, denoted by $NDCG@n(l_q)$, is then defined as the DCG score of l_q at n normalized by the DCG score of g_q at n, that is,

$$NDCG@n(l_q) = \frac{DCG@n(l_q)}{DCG@n(g_q)}.$$

We use an example to further illustrate the DCG and $NDCG$ scores. Suppose a ranking method generates a ranked list with 6 documents, $l_q = [D_1, D_2, D_3, D_4, D_5, D_6]$, and the relevance scores for all documents are: 3, 2, 3, 0, 1, 2. Then, the DCG@6 score of l_q is $DCG@6(l_q)$=8.09. If we sort the document in terms of their relevance scores, we get the ground-truth ranked list to be $g_q = [D_1, D_3, D_2, D_6, D_5, D_4]$ and $DCG@6(g_q) = 8.693$. Thus, the $NDCG$@6 score of l_q is $NDCG@6(l_q) = \frac{DCG@6(l_q)}{DCG@6(g_q)} = 0.93$.

- *Precision* and *Mean Average Precision (MAP)* [3].
  Given a query q and a ranked list of documents l_q, suppose every document in l_q is either "relevant" or "irrelevant" to q. The measurement *Precision at* n, denoted by $P@n$, measures the accuracy of the top-n results in l_q w.r.t. q, and is defined as follows:

$$P@n = \frac{|\{d|d \text{ is relevant and is within top } n\}|}{n}.$$

  The *Average Precision(AP)* of the ranked list l_q with respect to the query q is then defined as

$$AP(l_q) = \sum_{n=1}^{L} \frac{P@n \times pos(n)}{|\{d|d \text{ is relevant}\}|},$$

  where L is the "*truncation level*", and the function $pos(n)$ is given by:

$$pos(k) = \begin{cases} 1 & \text{the document at position } n \text{ is relevant, or} \\ 0 & \text{otherwise.} \end{cases}$$

  Given a set of queries and the ranked lists retrieved w.r.t. each query, the MAP score is the mean of the AP scores for each query.

Further Discussion It is well recognized that feature selection is important for classification, ranking and so on, since an effective set of features reduces model over-fitting, enhances the accuracy, and decreases model training time. The authors in [16] propose a ranking feature quality measurement, which shows that a feature's quality is based on the ranking result given by the model trained only by this feature.

In the ranking process, documents with the same relevance score, called ties, usually are ranked arbitrarily by using the parameters MAP or $NDCG$. Given a set of documents with definite relevance scores, arbitrarily handling ties possibly gives very different ranked lists when MAP and $NDCG$ vary tremendously. For example, suppose we have a set of documents and the ground truth is $g = [D_1, D_3, D_2, D_6, D_5, D_4]$, and the ground truth relevance scores for all documents are: $\langle 3, 3, 2, 2, 1, 0\rangle$. A model predicts relevance scores for each document is given by: $\langle 1, 1, 1, 1, 1, 1\rangle$. Then arbitrarily handling ties can give predicted ranked lists $l^{'}$ and $l^{''}$ as follows: $l^{'} = [D_1, D_3, D_2, D_6, D_5, D_4]$ with $MAP = 1$ and $NDCG = 1$, and $l^{''} = [D_4, D_5, D_6, D_2, D_3, D_1]$ with $MAP = 0$ and $NDCG = 0$.

Compared to [16] which uses MAP and $NDCG$ to evaluate the effectiveness of a feature, our proposed significance criterion, $S(\alpha)$, has the advantage of handling ties in the ranked list.

3 SFP-Based Ranking Framework

In this section, we present the framework of **S**ignificant **F**requent **P**attern-based **Rank**ing (*SFP-Rank*), which carries out ranking by the following steps: (1) data preprocessing, (2) coverage based significant pattern generation, and (3) model learning.

We first explain why frequent patterns as features are effective for solving the ranking problem. Then, a coverage based significant pattern generation algorithm that mines significant frequent features from the processed dataset is presented. Finally, selected significant patterns are used to extend the feature space of the original data, and the extended feature space is used to train the ranking model.

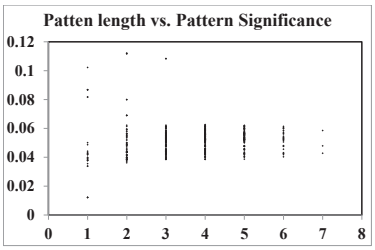
3.1 The Effectiveness of Using Frequent Patterns for Ranking

A frequent pattern has two essential properties in a dataset: having considerable number of combined features and possessing high frequency. We now study these two properties and explain how they are related to the ranking effectiveness of patterns.

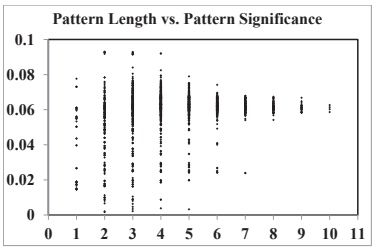
The usefulness of combined features The first property is the use of combined features. In practice, a large portion of frequent patterns are combined features. Compared with single features, combined features are better at capturing the underlying semantics of the documents and, thus, are more effective for producing accurate rankings.

With respect to the ranking problem, combined features are more likely to have a higher ranking significance than single features. To illustrate this, we plot the significance of single features (i.e. having pattern length equal to one) and combined features (i.e. having pattern length larger than one) in Microsoft LETOR dataset and Query Suggestion Dataset in Figure 2. We use Figure 2(a) as an example, from which we can see that combined features are more significant than single features, and other datasets also support this observation. By using significant combined patterns as features, we offer more effective features for ranking model training, and thus the ranking performance of the model can be improved.

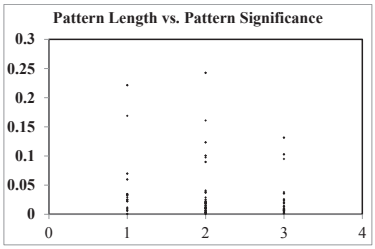
Ranking Significance and Pattern Frequency The second property of frequent patterns implies that frequent patterns cover a large number of instances in the dataset. Our further analysis will show that the significance of a low-support pattern is bounded by a small value. A low-support pattern has a small



(a) LETOR MQ2008 Fold2



(b) LETOR MQ2008 Fold5



(c) Query Suggestion Dataset

Fig. 2. Pattern Significance vs. Pattern Length

significance upper bound because of its limited coverage in the dataset and, thus, has limited effect on predicting the relevance score of a document.

We now analyze the relationship between a pattern's possible significance upper bound and a pattern's support. Given a query q, we retrieve a document set D_q and rank the documents in it. To simplify our discussion, we assume that the relevance score y of these documents is an integer of 0 or 1, and that $P(y = 1) = p_q$. The frequency of documents containing pattern α is $P(\alpha = 1) = \theta_q$, and $P(y = 1|\alpha = 1) = s_q$.

The ground truth is a list in which the documents with $y = 1$ are ranked higher than the documents with $y = 0$, and the documents with the same y are ranked arbitrarily. When using a pattern α to train the LTR model, the model has two prediction options as follows:

(1) $y(predicted) = 1$ when a document contains α, $y(predicted) = 0$ when a document does not contain α.

(2) $y(predicted) = 0$ when a document contains α, $y(predicted) = 1$ when a document does not contain α.

The above two models are compared and the better one is used for the final result. Then, the Kendall Tau distance of the better model is regarded as the significance of this pattern. Since the documents with the same predicted relevance score are ranked arbitrarily, the final ranked list may vary and thus

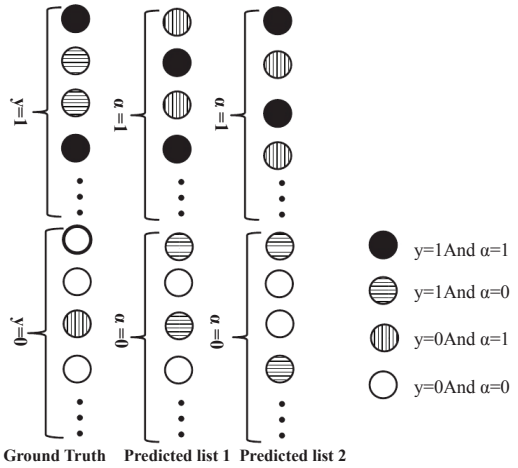


Fig. 3. An Example of Pattern Based Prediction

MAP or $NDCG$ may also vary (cf. see an example of prediction in Figure 3). However, our significance criterion can handle document ties.

Based on Formula (2), we have the expression as follows:

$$\tau(l_q^\alpha, g_q) = \frac{n_c(l_q^\alpha, g_q) - n_d(l_q^\alpha, g_q)}{\sqrt{n(q) - n_t(l_q^\alpha)}\sqrt{n(q) - n_t(g_q)}} = \frac{|\theta_q(s_q - p_q)|}{\sqrt{\theta_q(1-\theta_q)p_q(1-p_q)}}.$$

From the above formula, we can see that p_q is decided by the property of the dataset, and is constant for a given dataset. For a dataset with $p_q \leq 0.5$, we have:

$$\tau(l_q^\alpha, g_q)_{ub} = \begin{cases} \sqrt{\frac{\theta_q(1-p_q)}{p_q(1-\theta_q)}}, & \text{with } \frac{\partial \tau(l_q^\alpha, g_q)_{ub|s_q=1}}{\partial \theta_q} \geq 0 \text{ if } 0 \leq \theta_q < p_q \\ \sqrt{\frac{p_q(1-\theta_q)}{\theta_q(1-p_q)}}, & \text{with } \frac{\partial \tau(l_q^\alpha, g_q)_{ub|s_q=\frac{p_q}{\theta_q}}}{\partial \theta_q} \leq 0 \text{ if } p_q \leq \theta_q < 0.5 \\ \sqrt{\frac{p_q\theta_q}{(1-\theta_q)(1-p_q)}}, & \text{with } \frac{\partial \tau(l_q^\alpha, g_q)_{ub|s_q=0}}{\partial \theta_q} \geq 0 \text{ if } 0.5 \leq \theta_q < 1 - p_q \\ \sqrt{\frac{(1-p_q)(1-\theta_q)}{\theta_q p_q}}, & \text{with } \frac{\partial \tau(l_q^\alpha, g_q)_{ub|s_q=1-\frac{1-p_q}{\theta_q}}}{\partial \theta_q} \leq 0 \text{ if } 1 - p_q \leq \theta_q \leq 1 \end{cases} \tag{4}$$

For simplicity, we only give an illustration when $0 \leq \theta_q < p_q$. In this case, τ reaches its upper bound when $s_q = 1$; the upper bound is given by:

$$\tau(l_q^\alpha, g_q)_{ub|s_q=1} = \frac{\theta_q(1-p_q)}{\sqrt{\theta_q(1-\theta_q)p_q(1-p_q)}}.$$

The partial derivative of $\tau(l_q^\alpha, g_q)_{ub|s_q=1}$ w.r.t θ_q is then given by:

$$\frac{\partial \tau(l_q^\alpha, g_q)_{ub|s_q=1}}{\partial \theta_q} = \frac{1}{2}\sqrt{\frac{1-p_q}{p_q}}\sqrt{\frac{1-\theta_q}{\theta_q}}\frac{1}{\theta_q^2} \geq 0. \tag{5}$$

From Equation (5), we can see that the upper bound of τ linearly increases with pattern frequency in this case, and reaches its maximal when θ_q approaches p_q. When a dataset has the distribution of $p_q > 0.5$, we have:

$$\tau(l_q^\alpha, g_q)_{ub} = \begin{cases} \sqrt{\frac{p_q\theta_q}{(1-\theta_q)(1-p_q)}}, & \text{with } \frac{\partial \tau(l_q^\alpha, g_q)_{ub|s_q=0}}{\partial \theta_q} \geq 0 \text{ if } 0 \leq \theta_q < 1-p_q \\ \sqrt{\frac{(1-p_q)(1-\theta_q)}{\theta_q p_q}}, & \text{with } \frac{\partial \tau(l_q^\alpha, g_q)_{ub|s_q=1-\frac{1-p_q}{\theta_q}}}{\partial \theta_q} \leq 0 \text{ if } 1-p_q \leq \theta_q < 0.5 \\ \sqrt{\frac{\theta_q(1-p_q)}{p_q(1-\theta_q)}}, & \text{with } \frac{\partial \tau(l_q^\alpha, g_q)_{ub|s_q=1}}{\partial \theta_q} \geq 0 \text{ if } 0.5 \leq \theta_q < p_q \\ \sqrt{\frac{p_q(1-\theta_q)}{\theta_q(1-p_q)}}, & \text{with } \frac{\partial \tau(l_q^\alpha, g_q)_{ub|s_q=\frac{p_q}{\theta_q}}}{\partial \theta} \leq 0 \text{ if } p_q \leq \theta_q \leq 1 \end{cases} \tag{6}$$

Note that the above significance equation is defined for a single query q. In practice, we usually have several queries (i.e. a query set Q) and their related documents in the process. The significance of a pattern α, i.e., $S(\alpha)$, is thus defined as the average of the significance w.r.t. all the queries in Q. To calculate the average significance for Q, we divide Q into two query sets: Q_1 in which the corresponding documents of each query have $p_q \leq 0.5$, and Q_2 in which the corresponding documents of each query have $p_q > 0.5$. In the whole document set D, the frequency of documents containing pattern α is denoted by θ. Then we have the expression of the upper bound of $S(\alpha)$ as follows:

$$S(\alpha)_{ub} = \begin{cases} \frac{1}{\|Q\|}\left(\sum_{j=1}^{\|Q_1\|}\sqrt{\frac{\theta_{q_j}(1-p_q)}{p_{q_j}(1-\theta_{q_j})}} + \sum_{i=1}^{\|Q_2\|}\sqrt{\frac{p_{q_i}\theta_{q_i}}{(1-\theta_{q_i})(1-p_{q_i})}}\right), \\ \quad \text{with } \frac{\partial \tau(l_q^\alpha, g_q)_{ub}}{\partial \theta} \geq 0 \\ \quad \text{if } 0 \leq \theta < \frac{1}{\|Q\|}\left(\sum_{j=1}^{\|Q_1\|} p_{q_j} + \sum_{i=1}^{\|Q_2\|}(1-p_{q_i})\right). \\ \frac{1}{\|Q\|}\left(\sum_{j=1}^{\|Q_1\|}\sqrt{\frac{(1-p_{q_j})(1-\theta_{q_j})}{\theta_{q_j}p_{q_j}}} + \sum_{i=1}^{\|Q_2\|}\sqrt{\frac{p_{q_i}(1-\theta_{q_i})}{\theta_{q_i}(1-p_{q_i})}}\right), \\ \quad \text{with } \frac{\partial \tau(l_q^\alpha, g_q)_{ub}}{\partial \theta} \leq 0 \\ \quad \text{if } \frac{1}{\|Q\|}\left(\sum_{j=1}^{\|Q_1\|}(1-p_{q_j}) + \sum_{i=1}^{\|Q_2\|}(p_{q_i})\right) \leq \theta \leq 1. \end{cases} \tag{7}$$

The above analysis reveals a non-trivial relation between pattern frequency and significance. Equation (7) implies that the upper bound of significance monotonically increases with θ when θ is small (i.e. θ is in the range of $[0, \frac{1}{\|Q\|}(\sum_{j=1}^{\|Q_1\|} p_{q_j} + \sum_{i=1}^{\|Q_2\|}(1-p_{q_i}))]$) since the partial derivative $\frac{\partial \tau(l_q^\alpha, g_q)_{ub}}{\partial \theta} \geq 0$ by using Equation (6). On the other hand, Equation (7) implies that the upper

bound of significance monotonically decreases with θ when θ is high (i.e. θ is in the range of $[\frac{1}{\|Q\|}(\sum_{j=1}^{\|Q_1\|}(1-p_{q_j})+\sum_{i=1}^{\|Q_2\|}(p_{q_i})),\ 1]$) since the partial derivative $\frac{\partial\tau(l_q^\alpha,g_q)_{ub}}{\partial\theta} \leq 0$ by using Equation (6).

Intuitively, very low support patterns are clearly not desirable but the significance may increase if the support increases, since the bound increases with θ. However, very high support patterns are also not desirable, since the bound decreases with θ. Admittedly, Equation (7) only indicate the "possible" significance of patterns with respect to their support. It is thus an interesting theoretic problem to clarify the relation between the significance and the support of a patterns. This needs to take into consideration of the parameters characterizing a given dataset, which is beyond the scope of this study.

3.2 Coverage Based Significant Pattern Generation

Although frequent patterns are useful for improving the accuracy of the ranking, it does not mean all frequent patterns are usable. A good example is those stop words that appear frequently in most documents, but these words are almost useless in ranking documents. Since frequent patterns are generated by only considering their frequency, the number of mined frequent patterns can be very large and may contain a large portion of useless patterns in the mining result. This is undesirable for model training, since these patterns not only increase the model training time, but also degrade the ranking performance. Thus, we propose our pattern generation algorithm based on dataset coverage, which shares similar spirit with the feature selection methods used in [10, 25]. The basic idea is that, in the pattern generation process, we mine the most significant pattern from the current dataset, and for those instances not covered by this pattern, we recursively find other patterns from those instances.

We now present our **C**overage **B**ased **S**ignificant **P**attern **Gen**eration (or CBSP_Gen) algorithm, which generates the most significant patterns from the data.

In Algorithm 1, Lines 1 to 3 check if the current dataset size is smaller than the minimum instance size m, if yes, then we stop finding more patterns. Line 4 uses the FP-Close [17] algorithm to mine frequent patterns from the current dataset, then Lines 5 to 13 scan all the mined patterns, compute the significance for each pattern and find the pattern with the highest significance value. When computing the significance for a pattern α, for each query q in the query set Q, we first use a pattern α to train the ranking model and get the ranking list l^α given by the trained model. Then, the Kendall tau distance between l^α and the ground truth ranking list is computed. The significance of the pattern α is the average of the Kendall tau distance over Q. Although, for every pattern α, we need to train the ranking model for —Q— times, the pattern α only contains one feature and thus the model training step is still quite efficient.

Line 14 adds the selected significant pattern to set F_s. Lines 15 and 16 delete the instances that have been covered by current selected significant pattern. Line 17 recursively calls *CBSP_Gen* to continue finding significant pattern on

Algorithm 1 CBSP_Gen **C**overage **B**ased **S**ignificant **P**attern **Gen**eration

Input: A set of instances D from which features are to be mined, A support threshold p normalized between 0 and 1, Minimum instances size m.
Output: A selected set of patterns F_s.

if $|D| \leq m$ or (instances in D have the same relevance score) **then**
  return
end if
$P = FPGrowth(D, p)$
$BestPat = null$
$MaxSig$ =0
for each pattern α in P **do**
  $CurSig = computeSignificance(\alpha, D)$
  if $CurSig \geq MaxSig$ **then**
    $MaxSig = CurSig$
    $BestPat = \alpha$
  end if
end for
$F_s = F_s \cup BestPat$
S=subset of instances covered by $BestPat$
D= D-S
$CBSP\_Gen(D, p, m, F_s)$

the remaining instances that have not been covered by the selected pattern set F_s.

Bound on Number of Returned Features The algorithm works in a recursive manner and terminates when the instances in the training database are almost all covered (less than m instances not covered). We derive an upper bound of the number of recursions that *CBSP_Gen* has to go through.

Assume $min\_sup = \theta_0$, and *CBSP_Gen* produces a set of frequent patterns $BestPat_i$ in the ith iteration, $sup(BestPat_i) \geq \theta_0$. In the ith iteration, we eliminate the training instances S_i from the current set of training instances since they are covered by the feature $BestPat_i$. Therefore, we have the following equation which specifies the reduction of the training instance database:

$$|D_i| = |D_{i-1}| - |S_i|,$$

where D_i is the set of training instances remaining after the ith iteration, S_i is the id list of transactions which contain $BestPat_i$, and D_0 is the complete set of training instances.

Since $sup(BestPat_i) \geq \theta_0$, we have $|S_i| \geq \theta_0 |D_{i-1}|$ in equivalence. Then we have

$$|D_i| = |D_{i-1}| - |S_i| \leq (1 - \theta_0)|D_{i-1}|.$$

According to the above formula, we have:

$$|D_i| \leq (1-\theta_0)^i |D_0|.$$

Assume after n iterations, the training dataset reduces to $|D_n| = m$. Since $(1-\theta_0)^i |D_0| \geq |D_n| = m$, we can derive:

$$n \leq \frac{log\frac{|D_0|}{m}}{log\frac{1}{1-\theta_0}} = log_{\frac{1}{1-\theta_0}} \frac{|D_0|}{m}.$$

According to the above expression, if $\theta_0 = 0.5$ and $m = 1$, $n \leq log_2|D_0|$. If $\theta_0 = 0.2$ and $m = 1$, $n \leq log_{1.25}|D_0|$. If the training dataset has 1 million instances, then $n \leq 20$ if $\theta_0 = 0.5$; $n \leq 62$ if $\theta_0 = 0.2$. We can see that even for a relatively large dataset, our algorithm only has several iterations and can terminate soon.

3.3 Model Training and Predicting in SFP-Rank

We now present the model training and predicting details in our *SFP-Rank* approach. Algorithm 2 is developed for *SFP-Rank* Training and Algorithm 3 is developed for *SFP-Rank* Predicting. In the training phase, after we preprocess the dataset (Line 1), our proposed pattern generation method (Algorithm 1) is used to generate a set of optimal significant patterns F_s (Line 2). The generated patterns are used to extend the original feature space of the dataset (Line 3), and the extended dataset is used to train a ranking model M, using RankSVM, RankNet, and etc (Line 4). In the prediction phase as illustrated in Algorithm 3, we use the optimal pattern set F_s to extend the feature space of the testing instances (Line 1), and then ranking model M is used to predict the relevance score of testing instances (Line 2).

Algorithm 2 SFP-Rank Training

Input: Training dataset D
Output: Ranking model M, Optimal pattern set F_s
1: D' =Preprocessing(D). //data discretization etc.
2: F_s =CBSP_Gen(D', p, m, F_s). //pattern generation (Algorithm 1).
3: D'' =FeatureSpaceExtension(F_s, D'). //feature space extension using F_s and D'.
4: M =ModelTraning(D'') //model training based on extended dataset.
5: **return** F_s and M

4 Experiments

In this section, we evaluate the effectiveness of the *SFP-Rank* approach. The major components of our framework that we study are Data Preprocessing, Pattern Generation, and Ranking, as already shown in Figure 1. First, we present

Algorithm 3 SFP-Rank Predicting

Input: Optimal pattern set F_s, Ranking model M, Testing instance t
Output: Predicted relevance score y for t
1: t' =FeatureSpaceExtension(F_s, t) //feature space extension for t using F_s.
2: y =Prediction(M, t') //relevance score prediction for t' using model M
3: **return** y

the datasets, the data preprocessing method, and the mining strategy we used in the experiments in Section 4.1. Then, we evaluate the effectiveness of the ranking produced by *SFP-Rank* in Section 4.2. Finally, we study the scalability of *SFP-Rank* in Section 4.3.

4.1 Experimental Setup

Datasets In our experiments, the Microsoft's LETOR benchmark [30] is adopted for evaluation. LETOR is a benchmark dataset that is commonly used for studying LTR. The benchmark is composed of several data subsets, evaluation tools, and baseline evaluation results (such as RankSVM, RankBoost, etc) for ranking performance evaluation. Each data subset contains a set of queries, a set of features for query document pairs, and a set of corresponding relevance scores for the evaluation. We choose the LETOR4.0 MQ2008 dataset, the statistics of which is listed in Table 1. The MQ2008 dataset contains five folds. For each fold, the training set is first used to learn a ranking model. The validation set is used for model parameters tuning, and the ranking model is then used on the testing set. The estimated relevance scores on the testing set are employed to derive the standard $NDCG@n$, $P@n$, and MAP measures for the ranking evaluation.

Table 1. Statistics of the MQ2008 Dataset

No. of Folds	5
No. of Features	46
No. of Queries	784
No. of Query-Documents	15211
No. of Documents	14384

Ranking Model In our experiments, *RankSVM* [22] is employed to derive the ranking model. RankSVM utilizes instance pairs and their preference labels in the training. Specifically, we take RankSVM^{Struct}, which is the most up-to-date implementation of RankSVM with optimized speed and performance.

Data Preprocessing Most pattern mining algorithms, such as Apriori [2] and FP-Close [17], can only handle discrete attributes. However, since the attributes

of most of the ranking datasets (e.g., Microsoft's LETOR datasets, Yahoo's LTR competition[1] datasets) are continuous, data discretization should be performed before frequent pattern mining. Naive discretization methods such as binary discretization or n-equal-width bin discretization suffer from two major problems: first, information loss, which decreases the significance of frequent patterns, and second, noisy patterns, which are useless patterns that make mining and pattern selection more expensive. Since, if the discretization is not fine enough, it assigns many different values to the same bins, thus generating noisy patterns with information loss.

In our experiment, we compare several discretization methods such as equal-width-bin discretization, MDL [14]. We find that, compared to the original datasets, MDL yields the best results due to minimal information loss. In the remainder of the paper, we refer our data discretization method as MDL, unless otherwise specified.

Frequent Pattern Mining Frequent pattern mining is a well-studied theme with various available algorithms and software tools for our work. In our experiment, instead of using tools of mining frequent patterns, we consider algorithms for mining *Closed Frequent Patterns* (CFPs) in our framework, since a CFP is a concise representation of all its redundant non-closed sub-patterns. We adopt the algorithm FP-Close [17] to mine closed frequent patterns in our experiments due to its high efficiency and well-proved reliability [10]. To maximize the number of significant patterns, we divide each dataset into several partitions according to the relevance scores. We first mine the frequent patterns in each partition. The mined patterns are merged together, and pattern selection is then applied to the merged patterns to find the optimal pattern set.

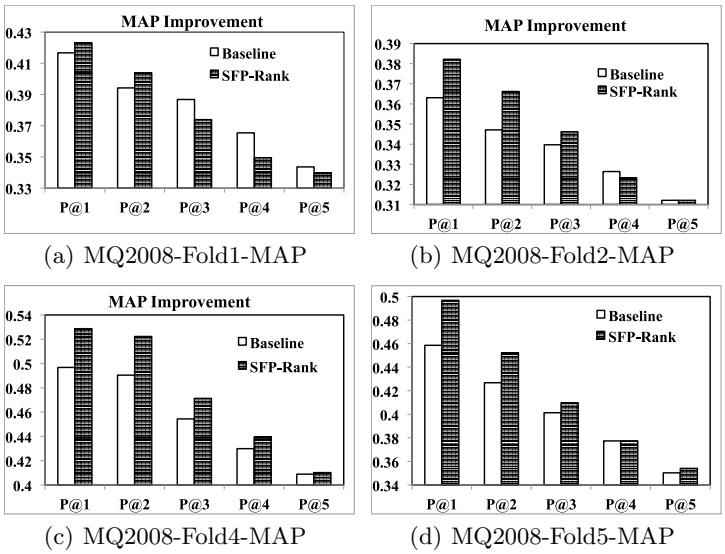
4.2 Ranking Performance

Table 2. Summary of Ranking Improvement on MQ2008 dataset

	MAP			*NDCG@10*		
Fold	Baseline	SFP-Rank	Improvement	Baseline	SFP-Rank	Improvement
F1	0.4502	0.4552	1.11%	0.4577	0.4686	2.38%
F2	0.4213	0.4314	2.40%	0.4296	0.4417	2.82%
F3	0.4529	0.4529	0%	0.4686	0.4686	0%
F4	0.5284	0.5401	2.21%	0.5442	0.5535	1.71%
F5	0.4950	0.5131	3.66%	0.5159	0.5294	2.62%
Avg.	0.46956	0.47854	1.91%	0.4832	0.49236	1.89%

We now discuss the result concerning the effectiveness of ranking from *SFP-Rank*.

[1] http://learningtorankchallenge.yahoo.com/datasets.php



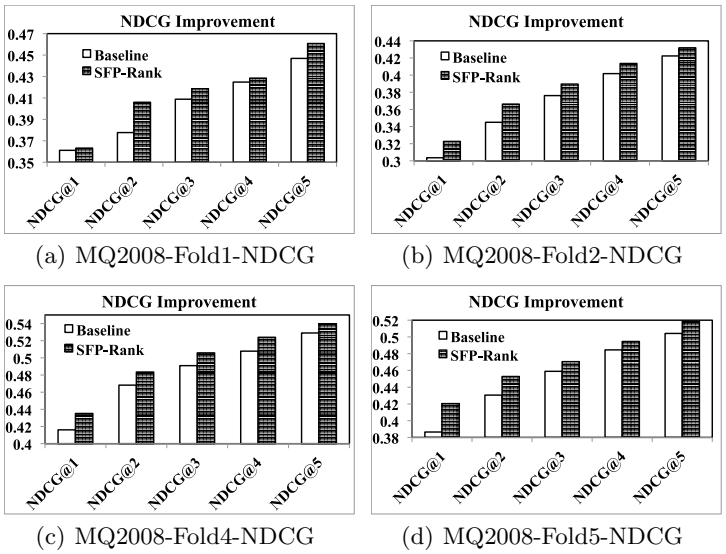
(a) MQ2008-Fold1-MAP (b) MQ2008-Fold2-MAP

(c) MQ2008-Fold4-MAP (d) MQ2008-Fold5-MAP

Fig. 4. *MAP* Improvement of Ranking

We set the minimum support threshold p to be 0.5, and fix the minimum instance size m to be 0. The ranking results in terms of MAP and $NDCG@10$ for the MQ2008 dataset are presented in Table 2. The details of all the folds except F5 against MAP and $NDCG$ measurements are given in Figures 4 and 5. From the results of two figures, we can easily check that the newly added significant frequent patterns can greatly improve the ranking performance. F3 (not shown) has no gain in improvement due to the fact that the original datasets is already good enough and the mined frequent patterns offer no further help as added features. The *SFP-Rank* algorithm achieves much better results compared to the baseline method (i.e. RankSVM with no pattern added) and the baseline RankSVM^{Struct} method (cf. Table 2) (Maximum: 3.66% in terms of MAP; Maximum 2.82% in terms of $NDCG$). This aligns with our claim in Section 3.1 that ranking performance can be improved by including an optimal frequent pattern set. Figures 4 and 5 give the detailed improvements on $NDCG@n$ and $Precision@n$ for n varying from 1 to 5. We can see especially that the quality of the top 5 ranked documents is higher compared with that of the baseline algorithm.

When the minimum support threshold p becomes smaller, an greater amount of frequent patterns are mined during feature selection. As more frequent patterns are available, features with larger significance values can be selected. Accordingly, the ranking results will be improved. However, the number of frequent patterns increases sharply as p gets smaller, and thus the time cost for frequent pattern mining becomes much longer. We set a minimum instance threshold m to allow at most m instances being uncovered by the selected features. In this way, we can achieve a better trade-off between the efficiency of the feature se-



(a) MQ2008-Fold1-NDCG (b) MQ2008-Fold2-NDCG

(c) MQ2008-Fold4-NDCG (d) MQ2008-Fold5-NDCG

Fig. 5. *NDCG* Improvement of Ranking

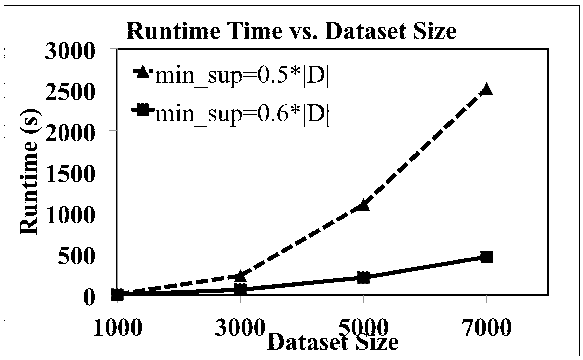
lection process and the quality of features, since a large number of patterns are possibly needed to be checked in order to cover the last one or two instances. However, in our experiments, all the instances can be quickly covered during feature selection, and so we simply set m to be 0. Actually, when m is set to be a small value like 1, 2, or 3, the ranking accuracy is stable.

4.3 Scalability Tests

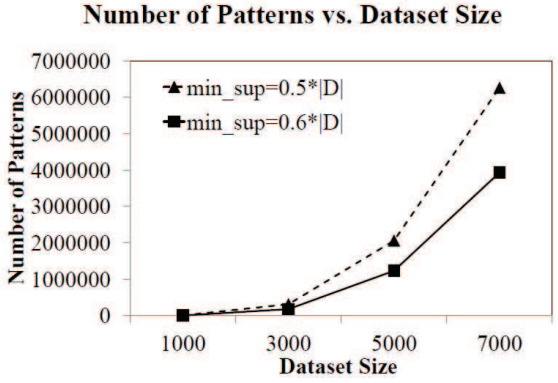
We now discuss the result concerning the efficiency of *SFP-Rank*. The process of scalability tests are conducted using the Microsoft LOTOR4.0 dataset. We vary the minimum support for frequent pattern mining to be $0.5 * |D|$ and $0.6 * |D|$ (recall that $|D|$ is the size of the dataset D). The runtime of *SFP-Rank* and the number of mined patterns with respect to the dataset size are respectively presented in Figure 6(a) and Figure 6(b). From the figures, we can see that, as we increase the size of the dataset, the number of patterns mined by the frequent pattern mining algorithm increases, thus the runtime of the pattern selection algorithm also increases. Finally, we remark that, even though pattern selection takes several minutes to establish, this can be performed completely offline in practice.

5 Application of SFP-Rank in Query Suggestion Ranking

In this section, we apply *SFP-Rank* to rank query suggestions generated by search engines. We show that *SFP-Rank* significantly outperforms a baseline method, RankSVM, in terms of both *precision*, MAP, and $NDCG$.



(a) Dataset Size vs. Runtime (sec)



(b) Dataset Size vs. Number of Patterns

Fig. 6. Scalability Analysis of SFP-Rank

5.1 Query Suggestions in Search Engines

User input queries in a search engine are usually short and ambiguous (cf. the average query length on a popular search engine was only 2.35 terms [20]). To clarify the users' search intent, many search engines like Google, Yahoo! and Live Search provide query suggestions to help users formulate more effective queries to refine their search as illustrated by the example in Figure 7. The suggestions are usually semantically related to the user's input query terms, and are usually mined from the search logs [7]. In order to easily identify the most effective suggestion candidates, we propose to apply *SFP-Rank* to rank the suggested queries in the search logs according to their retrieval effectiveness. Intuitively, the higher the ranking of the suggested query, the more effective the suggested query in the search refinement.

Fig. 7. Query Suggestions for the Input Query "apple"

5.2 Datasets

We utilize a real-world AOL search query log [1] to prepare the experimental data. A typical log entry includes an anonymous UserID, the query, the time at which the query was submitted, the rank of the clicked items and the domain portion of the clicked URLs. Some samples of AOL log entries are presented in Table 3. We utilize Jiang et al.'s method [21] to derive the search sessions for the 5000 queries in the raw AOL log, since this method was shown to be effective by their experiments. We use a 30-minute temporal cutoff to segment queries into *search sessions*. Queries within the same search session are assumed to be semantically related to one another, and thus they can be considered as good candidates for query suggestions.

Table 3. Examples of Search Query Log Entries

UserID	Query	Time	Rank	URL
606428	south africa map	2006-05-28 19:29:56	1	http://www.go2africa.com
3551754	printed price tags	2006-05-11 17:51:27		
2201072	nike air monarch	2006-04-19 14:55:40	4	http://www.shopping.com

Based on the query log and derived search sessions, we use a commonly used query taxonomy to derive 5,000 pairwise judgments [21] as the ground truth quality of the queries. Each query is represented by nine features presented in Table 4.

We compare the performance of the trained model with frequent pattern based enrichment using *SFP-Rank* against a baseline method, RankSVM, without feature enrichment.

5.3 Performance

Based on the nine features presented in Section 5.2, we conduct the ranking using *SFP-Rank*. First, we discretize the attributes according to MDL and obtain 4, 4, and 13 features for the 3rd, 5th and 6th attributes, and 1 feature for the other attributes, respectively. After performing the pattern mining, 9 patterns with the

Table 4. Nine Features of Query q

Feature Names	Meaning
Query length (by character)	Number of characters in q
Query length (by word)	Number of words in q
Sequence	Sequence number of q within the session
Total query number	Number of queries in the session
Previous time gap	Time gap between the previous query in the same session (in msec)
Following time gap	Time gap between the next query in the same session (in msec)
Noun number	Number of nouns in q
Adjective number	Number of adjectives in q
Verb number	Number of verbs in q

highest significance are chosen as shown in Table 6. The chosen patterns together with the nine features are used integrally to represent the search queries.

Table 5. Performance Comparison

Models	Precision	MAP	NDCG
RankSVM	0.4573	0.7286	0.6382
SFP-Rank	0.6401	0.8200	0.7600
Improv.	39.97%	12.54%	19.08%

The comparison results are presented in Table 5. We observe that the *SFP-Rank* model significantly outperforms the baseline RankSVM in terms of all the three metrics of *Precision*, MAP and $NDCG$.

Table 6. Selected Patterns

ID	Support	Pattern Details
1	1076	previous time gap: (17.5-inf) and following time gap: (55.5-2454.5]
2	605	previous time gap: (0.5-22.5]
3	577	sequence: (-inf-12.5] and following time gap: (0.5-22.5]
4	347	previous time gap: (17.5-inf) and following time gap: (0.5-22.5]
5	338	sequence: (-inf-12.5], previous time gap: (17.5-inf) and following time gap: (0.5-22.5]
6	289	sequence: (17.5-inf)
7	143	sequence: (-inf-12.5] and following time gap: (43.5-55.5]
8	145	previous time gap: (-inf-3.5] and following time gap: (0.5-22.5]
9	141	sequence: (-inf-12.5], previous time gap: (-inf-3.5] and following time gap: (0.5-22.5]

We use *Pattern 9* from Table 6 as an example to illustrate why the selected patterns are effective for ranking. Figure 8 shows an example of search sessions, in which a new session starts from Q_m to Q_k and a user may first input an ambiguous query. Based on the search results of the ambiguous query Q_m, the

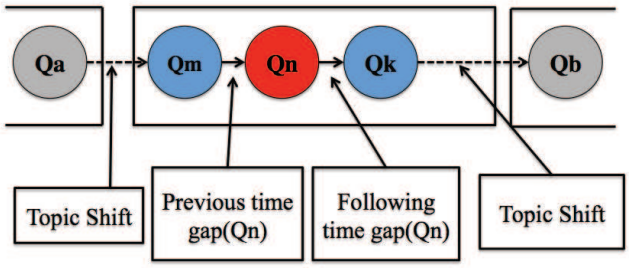


Fig. 8. An Example of Low Quality Query (Q_n) in a Session

user continues to refine his/her query terms in order to obtain better search results. *Pattern 9* is useful in eliminating low quality queries. For example, Q_n appears in the early stages of a query session (i.e. short *sequence*(Q_n)), and the user spends limited time on the search results of Q_m, which is the query immediately before Q_n (i.e. short *previous time gap*(Q_n)). Q_n may not be focused enough to represent a particular search intent. Furthermore, the user also spends a small amount of time on the search result of Q_n (i.e. short *following time gap*(Q_n)). All together we have the following three characteristics: (1) short *sequence*, (2) short *previous time gap* and (3) short *following time gap*. This evidence shows that Q_n tends to be a low-quality query, which should be ranked low in the query suggestions, and that *Pattern 9* is a very useful combined feature in query suggestion ranking to filter out low quality queries.

6 Related Work

There are two areas of related work. One is frequent pattern mining based classification. Another is the work about learning to rank methods. We discuss them in the following two subsections.

6.1 Frequent Pattern Mining Based Classification

Traditional frequent pattern mining has been a focused theme in data mining research, which gives rise to a large number of scalable methods. A comprehensive survey can be found in [18]. Besides traditional techniques of deterministic frequent pattern mining, mining frequent itemsets over uncertain databases has also attracted much attention recently. For example, Tong et al. [36–38] compare eight representative approaches of uncertain frequent itemset mining and develop a comparable software platform.

The frequent pattern-based classification is inherently related to associative classification. In associative classification, a classifier is built upon high quality rules, such as the ones with high confidence and high support. The association

between frequent patterns and class labels is then used for prediction. The work related to this area includes CBA[27], CMAR[26], CPAR[46] to name a few. These methods differ in their rule selection criteria (e.g. confidence, support, etc), number of rules they select (e.g. dataset coverage, top N, etc), and prediction result combination methodology. Cheng [10] provides a theoretical analysis of the relation between a pattern's support and its information gain. Recent work in this area focuses on how to mine the discriminative pattern efficiently. For example, Cheng [10] proposes a pattern selection method called MMRFS to select frequent patterns from the candidate pattern set. HARMONY [44] adopts an instance-centric rule generation approach and achieves high accuracy and efficiency in the mining process. DDP-Mine [11] develops a more effective pruning technique and directly mines out informative patterns for classification. Batal et al. [4] establish a minimal predictive patterns framework to directly mine a compact set of highly predictive patterns to improve the features of classification problem.

Our approach is inspired by the success of existing frequent pattern based classification approaches. However, our study differs from the frequent pattern based classification problem in two aspects. First, we consider the characteristic of LTR problem and propose a new criterion named *significance* for evaluating the feature. Second, we study the relationship between a pattern's significance and its frequency and demonstrate that frequent patterns are able to improve the performance of LTR. Notably, our approach is flexible and current frequent pattern mining algorithms can be directly used in SFP-Rank to generate frequent patterns.

6.2 Learning to Rank Methods

Ranking is a fundamental problem in many database and IR application areas, such as recommendation systems, document retrieval and advertising and so on. Previous works such as boolean models, vector models and probabilistic models [3] usually have the problem of high cost parameter tuning, since we usually have to consider a large number of relevant features for documents and queries in practice.

Machine learning techniques provide many feasible solutions to learn a ranking model. The techniques offer many different means to automatically learn parameters and then make use of a large percentage of the features in the model learning process. This approach of establishing a ranking model is referred to as the *Learning to rank* (LTR) approach. According to the work in [8, 9], current LTR methods can be classified into three categories as follows: (1) the pointwise approach, (2) the pairwise approach, and (3) the listwise approach. In the pointwise approach, each training example is treated as an independent instance and a model is trained to map each document's features to its relevance score, which could be based on regression [12] or classification [28, 24]. The pairwise approach trains the ranking function to minimize a loss function which is based on pair-wise preferences. The ranking problem is then transformed into a binary classification problem. Typical examples of such models includes RankSVM [22],

RankNet [5], FRank [39], MHR [31], RankBoost[15], and CRR[32] and the like. In the listwise approach, the models consider the whole document list instead of document pairs by either directly optimizing the IR measures, or indirectly optimizing the IR measures by employing a loss function correlated to IR measures. Directly optimizing the IR measures is difficult, since they depend on the rank and are not differentiable. Example methods include [6], SVM^{map} [47], AdaRank [45], Boltzrank [43], and NDCG-Boost [40, 23]. Indirectly optimizing the IR measures includes RankCosine [29], and ListNet [9]. For example, RankCosine uses the cosine similarity between the ranked list and the ground truth as a query level loss function. Recently, the LTR approach has been successfully applied to other application areas such as biological motion trajectories [35], social update streams [19], answering ranking [42], and etc. Besides the aforementioned approaches, association rules have also been applied to solve the LTR problem by Veloso [41]. When predicting the orders, several high confidence rules are used and the final relevance score is computed by the weighted combination of the relevance score of all these selected rules.

Our approach differs from these approaches in the following three aspects. First, we use frequent patterns to extend the feature space. Second, we do not only consider confidence or support of patterns or association rules. We also consider the characteristics of the ranking problem and importantly, develop means to select high significance, low redundancy patterns as features for effective ranking. Third, our approach is compatible with most current LTR algorithms and demonstrates significant ranking improvement.

While most of the research on LTR focuses on designing effective ranking models as a fundamental part, few studies considered improving the quality of the document features, which have a high impact on ranking quality. The studies in [33] show that, by incorporating a set of properly selected frequent patterns as features, the performance of the ranking model such as RankSVM can be substantially improved. Accordingly, a frequent pattern based ranking approach called FP-Rank was proposed. Compared to [33], which is a preliminary version of this paper, we propose a new feature evaluation criterion called *feature significance* to measure the effectiveness of features for training ranking model. The relationship between the significance of a pattern and its frequency is analyzed. Based on feature significance, a ranking approach SFP-Rank is also designed. By applying SFP-Rank in a scenario of ranking search engine query suggestions, we show that our approach is effective in identifying useful suggestion candidates for search refinement.

7 Conclusions

In this paper, we propose a ranking framework *SFP-Rank* that aims to achieve a more effective learning to rank approach by exploiting frequent patterns. Our study confirms that frequent patterns offer high quality features and they are able to improve the performance of a ranking model. Compared with commonly used feature selection approaches, our ranking feature selection method is able

to find an effective pattern subset that is specific for a ranking problem. The improvement is clearly evidenced by the ranking accuracy measured by MAP and $NDCG$ in a spectrum of experiments designed for studying *SFP-Rank*. We also study an application of *SFP-Rank* in query suggestion ranking. This demmonstrates the applicability of our ranking framework in real applications, since we improved the $NDCG$ and MAP significantly in the test datasets.

There is still some further work based on our approach. In particular, it would be interesting to see the performance of the *SFP-Rank* when it combines with other feature selection methods designed for LTR. The significant features selected by our method are shown to improve the ranking accuracy of RankSVM. Following this line, a more comprehensive study that involves other ranking methods such as RankNet can be carried out to confirm the generality of the approach. Also, another promising area to carry out would be developing more efficient algorithms for directly mining significant frequent pattern for LTR. Finally, some extended work of our approach can be done to show that the significant pattern based feature enrichment methods can also be applied to other domain applications, such as LTR in the context of transaction graphs and collaborative filtering.

References

1. Aol dataset. http://zola.di.unipi.it/smalltext/datasets.html
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB '94. pp. 487–499 (1994)
3. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison Wesley (1999)
4. Batal, I., Hauskrecht, M.: Constructing classification features using minimal predictive patterns. In: Proceedings of the 19th ACM international conference on Information and knowledge management. pp. 869–878. CIKM '10, ACM, New York, NY, USA (2010)
5. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: ICML'05. pp. 89–96 (2005)
6. Burges, C., R.Ragno, E.V.Le: Learning to rank with nonsmooth cost functions. In: NIPS '06. pp. 193–200 (2006)
7. Cao, H., Jiang, D., Pei, J., He, Q., Liao, Z., Chen, E., Li, H.: Context-aware query suggestion by mining click-through and session data. In: KDD '08. pp. 875–883 (2008)
8. Cao, Y., Xu, J., Liu, T.Y., Li, H., Huang, Y., Hon, H.W.: Adapting ranking svm to document retrieval. In: SIGIR '06. pp. 186–193 (2006)
9. Cao, Z., Qin, T., Liu, T.Y., Tsai, M.F., Li, H.: Learning to rank: from pairwise approach to listwise approach. In: ICML '07. pp. 129–136 (2007)
10. Cheng, H., Yan, X., Han, J., Hsu, C.W.: Discriminative frequent pattern analysis for effective classification. In: ICDE'07. pp. 169–178 (2007)
11. Cheng, H., Yan, X., Han, J., Yu, P.S.: Direct discriminative pattern mining for effective classification. In: ICDE'08. pp. 169–178 (2008)
12. Cossock, D., Zhang, T.: Subset ranking using regression. In: Learning Theory, LNCS'06, vol. 4005, pp. 605–619 (2006)

13. Fagin, R., Kumar, R., Sivakumar, D.: Comparing top k lists. In: SODA '03. pp. 28–36 (2003)
14. Fayyad, Irani: Multi-interval discretization of continuous-valued attributes for classification learning. In: UAI '93. pp. 1022–1027 (1993)
15. Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. J. Mach. Learn. Res. 4, 933–969 (December 2003)
16. Geng, X., Liu, T.Y., Qin, T., Li, H.: Feature selection for ranking. In: SIGIR'07. pp. 407–414 (2007)
17. Grahne, G., Zhu, J.: Efficiently using prefix-trees in mining frequent itemsets. In: FIMI'03 (2003)
18. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. Data Min. Knowl. Discov. 15(1), 55–86 (2007)
19. Hong, L., Bekkerman, R., Adler, J., Davison, B.D.: Learning to rank social update streams. In: SIGIR '12. pp. 651–660 (2012)
20. Jansen, B.J., Spink, A., Bateman, J., Saracevic, T.: Real life information retrieval: a study of user queries on the web. SIGIR Forum 32(1), 5–17 (Apr 1998)
21. Jiang, D., Leung, K.W.T., Ng, W.: Context-aware search personalization with concept preference. In: CIKM '11. pp. 563–572 (2011)
22. Joachims, T.: Training linear svms in linear time. In: KDD'06. pp. 217–226 (2006)
23. Karimzadehgan, M., Li, W., Zhang, R., Mao, J.: A stochastic learning-to-rank algorithm and its application to contextual advertising. In: WWW '11. pp. 377–386 (2011)
24. Li, P., Burges, C.J.C., Wu, Q.: Mcrank: Learning to rank using multiple classification and gradient boosting. In: NIPS'07. pp. 845–852 (2007)
25. Li, W., Han, J., Pei, J.: Cmar: accurate and efficient classification based on multiple class-association rules. In: ICDM'01. pp. 369 –376 (2001)
26. Li, W., Han, J., Pei, J.: Cmar: Accurate and efficient classification based on multiple class-association rules. In: ICDM'01. vol. 0, p. 369 (2001)
27. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: KDD'98. pp. 80–86 (1998)
28. Nallapati, R.: Discriminative models for information retrieval. In: SIGIR '04. pp. 64–71 (2004)
29. Qin, T., yan Liu, T., feng Tsai, M., dong Zhang, X., Li, H.: Learning to search web pages with query-level loss functions. Tech. rep. (2006)
30. Qin, T., Liu, T.Y., Xu, J., Li, H.: Letor: A benchmark collection for research on learning to rank for information retrieval. Information Retrieval 13, 346–374 (2010)
31. Qin, T., Zhang, X.D., Wang, D.S., Liu, T.Y., Lai, W., Li, H.: Ranking with multiple hyperplanes. In: SIGIR '07. pp. 279–286 (2007)
32. Sculley, D.: Combined regression and ranking. In: KDD '10. pp. 979–988. ACM, New York, NY, USA (2010)
33. Song, Y., Leung, K., Fang, Q., NG, W.: Fp-rank: An effective ranking approach based on frequent pattern analysis. In: DASFAA'13 (2013)
34. Tan, J., Bu, Y., Yang, B.: An efficient close frequent pattern mining algorithm. In: ICICTA '09. vol. 1, pp. 528 –531 (Oct 2009)
35. Thomas Fasciano, R.S., Shin, M.C.: Learning to rank biological motion trajectories. Image and Vision Computing (2012)
36. Tong, Y., Chen, L., Cheng, Y., Yu, P.S.: Mining frequent itemsets over uncertain databases. PVLDB'12 5(11), 1650–1661 (2012)
37. Tong, Y., Chen, L., Ding, B.: Discovering threshold-based frequent closed itemsets over probabilistic data. In: ICDE'12. pp. 270–281 (2012)

38. Tong, Y., Chen, L., Yu, P.S.: Ufimt: an uncertain frequent itemset mining toolbox. In: KDD'12. pp. 1508–1511 (2012)
39. Tsai, M.F., Liu, T.Y., Qin, T., Chen, H.H., Ma, W.Y.: Frank: a ranking method with fidelity loss. In: SIGIR '07. pp. 383–390 (2007)
40. Valizadegan, H., Jin, R., Zhang, R., Mao, J.: Learning to rank by optimizing ndcg measure. In: NIPS '09 (2009)
41. Veloso, A.A., Almeida, H.M., Gonçalves, M.A., Meira Jr., W.: Learning to rank at query-time using association rules. In: SIGIR '08. pp. 267–274 (2008)
42. Verberne, S., van Halteren, H., Theijssen, D., Raaijmakers, S., Boves, L.: Learning to rank for why-question answering. Information Retrieval 14, 107–132 (2011)
43. Volkovs, M.N., Zemel, R.S.: Boltzrank: learning to maximize expected ranking gain. In: ICML '09. pp. 1089–1096 (2009)
44. Wang, J., Karypis, G.: On mining instance-centric classification rules. IEEE Trans. on Knowl. and Data Eng. 18, 1497–1511 (2006)
45. Xu, J., Li, H.: Adarank: a boosting algorithm for information retrieval. In: SIGIR '07. pp. 391–398 (2007)
46. Yin, X., Han, J.: Cpar: Classification based on predictive association rules. In: SDM'03 (2003)
47. Yue, Y., Finley, T., Radlinski, F., Joachims, T.: A support vector method for optimizing average precision. In: SIGIR'07. pp. 271–278 (2007)