Concise Papers

A User-Friendly Patent Search Paradigm

Yang Cao, Ju Fan, and Guoliang Li

Abstract—As an important operation for finding existing relevant patents and validating a new patent application, patent search has attracted considerable attention recently. However, many users have limited knowledge about the underlying patents, and they have to use a try-and-see approach to repeatedly issue different queries and check answers, which is a very tedious process. To address this problem, in this paper, we propose a new user-friendly patent search paradigm, which can help users find relevant patents more easily and improve user search experience. We propose three effective techniques, error correction, topic-based query suggestion, and query expansion, to improve the usability of patent search. We also study how to efficiently find relevant answers from a large collection of patents. We first partition patents into small partitions based to their topics and classes. Then, given a query, we find highly relevant partitions and answer the query in each of such highly relevant partitions. Finally, we combine the answers of each partition and generate top-k answers of the patent-search query.

Index Terms—Patent search, error correction, query suggestion, query expansion

1 Introduction

Patents play a very important role in intellectual property protection. As patent search can help the patent examiners to find previously published relevant patents and validate or invalidate new patent applications, it has become more and more popular, and recently attracts much attention from both industrial and academic communities. For example, there are many online systems to support patent search, such as Google patent search,[1] Derwent Innovations Index (DII),[2] and USPTO.[3]

As most patent-search users have limited knowledge about the underlying patents, they have to use a try-and-see approach to repeatedly issue queries and check answers, which is a very tedious process. To help users easily find relevant patents, the first step for the patent search is to capture users' search intention. In other words, suggesting search keywords for users is the most critical part of the search strategy. After selecting the precise search keywords, the next step is finding and ranking the relevant answers. Most of existing methods focus on devising a complicated ranking model to rank patents and finding the most relevant answers [5], [14]. However, they do not pay enough attention to effectively capturing users' search intention, which is at least as important as ranking patents. To address this problem, in this paper, we propose a new user-friendly patent search paradigm, which can help users find relevant patents more easily and improve user search experience. As users' query keywords may have typos, existing methods will return no answer as they cannot find patents matching query keywords. To alleviate this problem, we propose an error-correction technique to suggest similar terms for the query keywords and return answers of the similar terms. In addition, to help users formulate high-quality queries, as users type in keywords, we suggest keywords that are topically relevant to the query keywords. In this way, users can interactively issue queries and modify their keywords if there is no relevant answer, which can provide users with gratifications. As users may not understand the underlying patents precisely, they may type in ambiguous keywords or inaccurate keywords. On the other hand, the same concept/entity may have different representations. For example, "car" and "sedan" are relevant to "automobile." Thus, if users type in a keyword "car," we may need to expand the keyword to "automobile." To this end, we propose a query expansion-based technique to recommend users relevant keywords. We discuss two methods to efficiently suggest relevant keywords. To summarize, we use these three techniques to help users search patents more easily and improve the usability of patent search.

In addition, existing methods only focus on the effectiveness of patent search [7] and neglect the fact that the search efficiency is also very important. To address this problem, we propose a new method to improve search efficiency. We note that the patents are usually classified into different classes based on the topics. There are around 400 classes and about 135,000 subclasses [7]. For a patent search query, only several classes of patents could be relevant to the query. Therefore, we can classify the patents based on the classes and the topics of the patents using the topic model [3], and generate several patent partitions, such that patents in the same partition are very topically relevant and those in different partitions are not very relevant. Then, given a query, we find highly relevant partitions and use each partition to efficiently find relevant patents of the query. Finally, we combine the results from each partition and generate the top-k answers. Experimental results show that our method achieves high efficiency and result quality.

1. http://www.google.com/patents.
2. http://apps.webofknowledge.com.
3. http://www.uspto.gov/patents/process/search/index.jsp.

- *Y. Cao is with the State Intellectual Property Office of the PRC, Beijing 100088, China. E-mail: caoyang_2@sipo.gov.cn.*
- *J. Fan and G. Li are with the Department of Computer Science and Technology, Tsinghua University, Room 10-204, East Main Building, Beijing 100084, China. E-mail: fan-j07@mails.tsinghua.edu.cn, liguoliang@tsinghua.edu.cn.*

Manuscript received 4 Nov. 2011; revised 2 Feb. 2011; accepted 5 Feb. 2012; published online 9 Mar. 2012.
Recommended for acceptance by B. Cui.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2011-11-0681.
Digital Object Identifier no. 10.1109/TKDE.2012.63.

2 Overview of Our User-Friendly Patent Search Paradigm

In this paper, we propose a user-friendly patent search method which can help users easily find relevant patents and improve user search experience. Fig. 1 illustrates the architecture of our patent-search paradigm.

The User-friendly Interface component is used to capture users' search intention and refine query keywords so as to find relevant answers. It consists of three subcomponents, error correlation, topic-based query suggestion, and query expansion. In addition, it groups the answers based on their topics to help users navigate answers. It also provides users with the patent snippets of the answers to help users quickly check whether the returned answers are relevant. Thus, users can interactively issue queries, browse the results and get the final answers, which can help them find relevant answers more easily.

To improve the efficiency, we partition patents into different data partitions based on their topics. We use a cluster to manage the patent data. Patent partitions are stored in different nodes in the cluster. The Indexing component builds inverted indexes on top of each partition. Then, for each query, the Patent Partition Selection component selects top-ℓ highly relevant data partitions and routes the query to such relevant partitions to find local answers. The Query Processing component computes answers in the local partitions. Finally, the Query Aggregation component

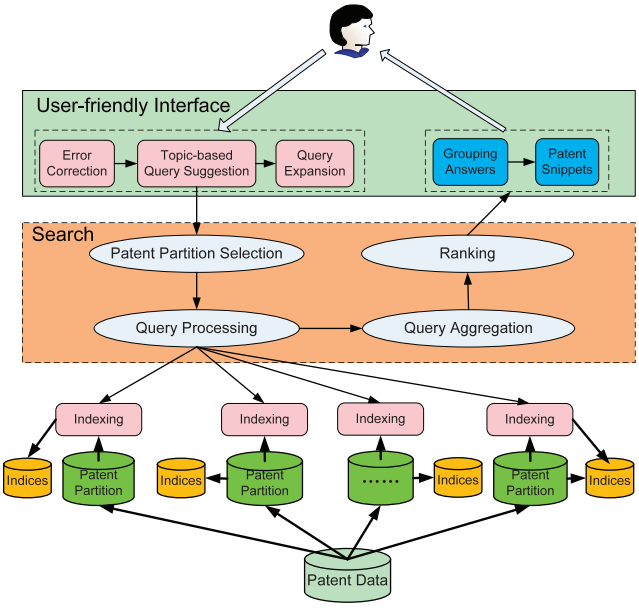


Fig. 1. User-friendly patent search architecture.

combines the local results and the Ranking component ranks the answers to return the final top-k answers.

3 User-Friendly Patent Search

There are several unique challenges in patent search, mainly due to the difficulty of understanding users's query intent and efficiently matching the query keywords to patents. In this section, we present several effective techniques to address these challenges.

3.1 Patent Partition

We partition into different data partitions due to the following reasons. First, patents inherently have different classes. There are about 400 classes and around 135,000 subclasses. Second, the number of patents is usually very large. For example, in USPTO, there are approximately 8 million patents and 3 million patent applications.[4] Moreover, the number of patents is increasing rapidly. For example, the annual growth rate of the total number of patents in China is 26.1 percent. Third, for a patent search query, only some classes/subclasses of patents could be relevant to the patent query. Based on these reasons, we partition the patents based on their classes and topics using the topic model [3] as follows: We first extract the topic of each patent. Then, we partition the patents with the same topic into the same data partition, and each topic corresponds to a data partition. Note that the patents in the same partition are highly relevant and those in different partitions are irrelevant.

3.2 Effective Indexing

For each partition, we build a well-known inverted index structure. For each query keyword, we use the index structure to find patents containing the keyword. Then, we intersect the patents corresponding to different keywords to generate the most relevant patents. In each partition, we can use any effective ranking function to rank the patents in the partition. As patents in each partition are very relevant, we can do more deep ranking by considering the correlation between different patents.

To facilitate query suggestion, we construct a trie structure on top of keywords in the patent partition. Each keyword in the patent partition corresponds to a unique path from the root of the trie to a leaf node. Each node on the path has a label of a character in the keyword. For each leaf node, we store an inverted list of IDs of records that contain the corresponding keyword. Readers interested in more detail about the trie structure are referred to [10]. We will discuss how to use the trie structure to do effective query suggestion as discussed in Section 3.3.2.

3.3 User-Friendly Interface

To capture users' query intention, we introduce several effective techniques to make patent search user friendly and help users easily find relevant patents.

3.3.1 Automatic Error Correction

As query keywords that users have typed in may have typos, traditional methods will return no answer as they cannot find answers that contain the query keywords. Obviously, this method is not user friendly. Instead, it is better to correct the typos, recommend users similar keywords, and return the answers of the similar keywords. To quantify the similarity between keywords, existing methods usually adopt edit distance. The edit distance between two keywords is the minimum number of edit operations (i.e., insertion, deletion, and substitution) of single characters needed to transform the first one to the second. For example, the edit distance of "patent" and "paitant" is 2. Two keywords are said to be similar if their edit distance is within a given threshold. There are some recent studies on efficient error correction, which use a filter-and-refine framework to find similar keywords of a query keyword. The method first uses the filter step to find a subset of keywords which may be potentially similar to the query keyword. Then, it uses a verification step to remove those false positives and get the final similar keywords [8]. Although we can use these methods to efficiently suggest keywords for complete keywords, they cannot support prefix keyword the user is completing. To address this problem, we can use the trie structure to do efficient keyword correction and completion [6], [10], [9]. Using the trie structure, even users type in a partial keyword, we can also efficiently suggest relevant accurate keywords. The basic idea is that if a prefix is not similar enough to a trie node, then we do not need to consider the keywords under the trie node. We can use this observation to efficiently suggest similar keywords. More details can be referred to [10].

3.3.2 Topic-Based Query Suggestion

We devise a novel model for effectively suggesting keywords as users type in queries letter by letter. The basic idea of our method is to use the topic model to estimate the *probability* of the next query keyword [4]. Intuitively, if a keyword in patents is more topically coherent with the previously typed query keywords, it would obtain a higher score. Specifically, we can focus on estimating two important probabilities: the probability of a keyword conditioned on topics, and the probability of sampling a keyword from a patent. Both of the two probabilities are used to estimate the score of each keyword. An LDA model [3] can be utilized to learn the keyword distribution over each topic from the underlying patents. LDA can be classified as a *soft-clustering* technique which allows a keyword to appear in multiple topics and takes into account the degree of a keyword belonging to each topic. The keyword distribution over a set of patents is learned by using a language model. The language model approach can capture the property of the patents and predict the *likelihood* of sampling a specific keyword. Thus, we can combine the two probabilities and use the topic-based method to suggest relevant keywords.

Formally, the probability of each term with a prefix conditioned on the context is defined as follows: Let $s = q_1, q_2, \ldots, q_{|s|}$ be a sequence of previous query terms, that is, the context. Let p be the prefix of the term that the user is typing in and $C = \{c_1, c_2, \ldots, c_{|C|}\}$

4. http://www.uspto.gov/patents/process/search/index.jsp.

be a set of complete terms with the prefix p. We denote $D = \{d_1, d_2, \ldots, d_{|D|}\}$ to be a partition of patents retrieved by the context s and $T = \{t_1, t_2, \ldots, t_{|T|}\}$ to be a set of topics. The task of our model is to estimate the probability $P(c|s)$ for each $c \in C$ and suggest the top terms with highest probabilities.

We give a generative model to estimate the probability $P(c|s)$ [4]. Each topic t_i in the topic set T and each patent d_j in the patent partition D are incorporated into the probability estimation:

$$P(c|s) = \sum_{t_i \in T} \sum_{d_j \in D} P(c|s; t_i, d_j) \cdot P(t_i, d_j|s) \text{ [4]}. \quad (1)$$

Intuitively, given a context s, our model would examine the likelihood that a user issues the term c if t_i is the topic of c and d_j is an expected answer. It is conceptually similar to the query likelihood $P(q|d_i)$ in the language model approaches: we first estimate a model from each topic and each patent, and then compute the likelihood of generating c according to this model as well as the context s. To estimate $P(c|s; t_i, d_j)$, ideally we should enumerate every possible term which is not only under the topic t_i, but also used by users who want to retrieve patent d_j given a context s. However, in reality, it is very difficult to achieve this goal. So, we assume that c is conditional independent of s given t_i and d_j and use $\lambda P(c|t_i) + (1-\lambda)P(c|d_j)$ (where λ is a tuning parameter) as an estimate for $P(c|s; t_i, d_j)$. On the other hand, the probability $P(t_i, d_j|s)$ can be analogous to the *prior* of choosing t_i and d_j given the context s. The more the topic t_i and the patent d_j are relevant to the context s, the more they have the *ability* to generate any term. We assume that t_i and d_j are independent of each other, thereby resulting in $P(t_i, d_j|s) = P(t_i|s) \cdot P(d_j|s)$. Hence, $P(c|s)$ is further estimated by

$$\begin{aligned} P(c|s) &= \sum_{t_i \in T} \sum_{d_j \in D} \big(\lambda P(c|t_i) + (1-\lambda)P(c|d_j)\big) P(t_i|s) P(d_j|s) \\ &= \lambda \sum_{t_i \in T} P(c|t_i) P(t_i|s) + (1-\lambda) \sum_{d_j \in D} P(c|d_j) P(d_j|s) \text{ [4]}. \end{aligned} \quad (2)$$

Clearly, (2) shows that the probability $P(c|s)$ depends on two factors. The first factor is the relationship between c and s in terms of topics. The second one is the likelihood of sampling c from each patent d_j retrieved by s. The two factors are combined linearly with a parameter λ. Note that in the case that the context is an empty string (i.e., $s = \varepsilon$), $P(c|s)$ is simply estimated by the frequency of patents containing c.

Next, we discuss the computation of $P(c|s)$ in (2). $P(c|t_i)$ is trained from the entire patent partition via LDA, and $P(t_i|s)$ can be rewritten to (3) with Bayes formula

$$P(t_i|s) = \frac{P(s|t_i) \cdot P(t_i)}{P(s)} \text{ [4]}. \quad (3)$$

Because the assumption that terms are independent of each other conditioned on topics is used in LDA and $s = q_1, q_2, \ldots, q_{|s|}$, we get (4) (where $s' = q_1, q_2, \ldots, q_{|s|-1}$)

$$P(t_i|s) = \frac{P(t_i) \cdot \prod_{l=1}^{|s|} P(q_l|t_i)}{P(s') \cdot \sum_{t_j \in T} P(q_{|s|}|t_j; s') \cdot P(t_j|s')} \text{ [4]}. \quad (4)$$

Obviously, both $P(s')$ and $P(q_{|s|}|t_j; s') \cdot P(t_j|s')$ have been computed from the estimate of $P(q_{|s|}|q_1, q_2, \ldots, q_{|s|-1})$ in the previous round.

The second factor of $P(c|s)$ is a sum of likelihoods of generating c from each patent, i.e.,

$$\sum_{d_j \in D} P(c|d_j) \cdot P(d_j|s).$$

We use the Jelinek-Mercer smoothing technique in language model approach to estimate $P(c|d_j)$ in

$$P(c|d_j) = (1-\eta)\frac{\text{count}(c, d_j)}{\mid d_j \mid} + \eta \frac{\text{count}(c, D)}{\mid D \mid} \text{ [4]}, \quad (5)$$

where $\frac{\text{count(c,d}_j\text{)}}{|d_j|}$ is term frequency of c in the patent d_j and $\frac{\text{count(c,D)}}{|D|}$ the frequency of c in a *background* model, i.e., the entire partition of patents, D. In addition, η is a tuning parameter. The intuition of (5) is that we interpolate the maximum likelihood estimation, $\frac{\text{count(c,d}_j\text{)}}{|d_j|}$, with the background model by using a tuning parameter η. This smoothing technique has been intensively examined and shown effective [4]. $P(d_j|s)$ is the prior probability that the patent d_j generates any term given the context s. We use a normalized ranking score of d_j under s to estimate it (the TF-IDF model).

3.3.3 Query Expansion

In many cases, users cannot understand the underlying data precisely. In this way, they may type in ambiguous keywords or inaccurate keywords. In addition, the same concept may have different representations. To this end, we can use WordNet to expand a keyword. If the query word is indexed by WordNet, we can easily get the relevant keywords of the query keyword using an inverted list structure. However, WordNet is artificially generated for common words. If the query keywords are not in WordNet, we cannot recommend relevant keywords. To address this problem, we have two solutions.

The first one is to utilize search engines, since most search engines will suggest relevant keywords as users type in queries. We can issue the patent query to search engines and get the relevant keywords from the search engines, such as Google. The second way is to mine the relevant keywords from the query logs. To this end, we use the click-through data to mine the correlated queries as follows: For two queries, if users click the same returned result (patent), they are potentially relevant. We utilize this property to mine relevant queries. For two queries, we use the number of times user clicked on the same patent to denote their relevance. If a keyword pair with their co-occurrence is larger than a given threshold, the two keywords are relevant and we use them to do query expansion.

Obviously, we can combine the two methods to improve the quality of suggested keywords. Given a keyword, we use both search engine and our mined results to get its relevant keywords. Then, we select the keywords with the highest scores to expand the keyword.

3.4 Ranking

Existing methods focus on effective ranking models to improve the result quality, and there are many ranking models [5], [14] to evaluate the relevancy between a query and a patent. Note that any existing ranking function can be incorporated into our search paradigm. Here, we only give several important factors that a good ranking model should consider [11]:

1. The importance of a patent p, denoted by W_p. The more important a patent, the higher probability relevant to a query. We can model patents as a graph where nodes are patents and edges are citations between patents. Thus, we can use the graph to compute the weight of a patent.
2. The keyword relevancy of a patent p to a query Q, denoted by $R(p, Q)$. We can use the well-known IR method (e.g., tf∗idf) to compute the relevancy.
3. The topic relevancy of patent p to query Q, denoted by $T(p, Q)$. We use the above topic-based method to compute the value.
4. The prior-art relevancy of a patent P_p, which can be computed similar to [5].

In this way, we combine the above factors to rank a patent p given with respect to a query Q, denoted by $S(p|Q)$, as follows:

$$S(p|Q) = \alpha * W_p + \beta * R(p,Q) + \gamma * T(p,Q) + (1-\alpha-\beta-\gamma) * P_p. \quad (6)$$

We use the above function to compute the relevancy between patent p and query Q and return the top-k most relevant patents.

3.5 Patent Partition Selection

Given a query, a straightforward method will issue the queries to each patent partition, and find relevant answers from each patent partition. Finally, it combines the answers from different patent partitions. However, many data partitions may be irreverent to the query, and thus we do not need to issue the query to such patent partitions. To improve the efficiency, we will not issue the query to every patent partition. Instead, we select the top-ℓ relevant patent partitions and use them to answer the query.

We need to evaluate relevancy of a query to a patent partition. There are several factors we need to consider to rank a patent partition. The first one is the topic relevancy. That is, whether the patent partition is topically relevant to the query keywords. The second one is keyword relevancy. That is, whether the patent partition contains query keywords. We can use `tf*idf` model to evaluate the relevancy. The third one is prior-art relevancy. That is, whether the patent partition is novel enough to the query. We can use the above equations and combine the three factors to select top-ℓ relevant partitions. There are some recent studies to select highly relevant databases [13], [15]. We can build inverted indexes for the keywords to the patent partitions that contain the keywords. Using the inverted indexes and our ranking model, we can easily extend their methods to select highly relevant patent partitions.

3.6 Query Processing

Given a query, to find its top-k answers, we first select top-ℓ relevant patent partitions, and issue the query to such relevant patent partitions. We use above ranking functions to compute the scores of different patent partitions. For each partition, we efficiently find top-k answers using our indexing structures and ranking model. Then, we combine the answers from each selected partition and generate the final top-k answers based on our ranking model. Our method can prune many irrelevant patent partitions and can improve the efficiency significantly. On the other hand, we propose three effective techniques to improve result quality.

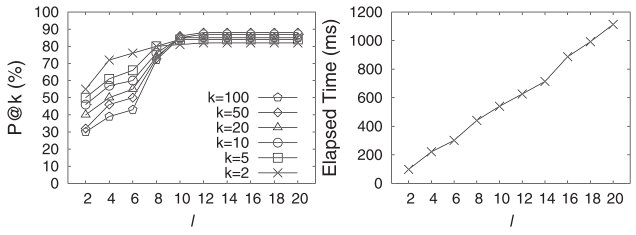
4 Experimental Study

We have implemented our proposed techniques. We used the well-known USPTO data set.[5] We crawled 0.5 million patents and the raw data set size was about 60 GB. As there is no standard query set, we used the similar method in [14] to generate a query set as follows: We used the patents in 2008 as a query set and used others as a data set. For each patent in the query set, we selected the top-t keywords with the highest frequencies (removing the stop words) from its title and abstract as a query, where t is varied from 1 to 5. We compared with the state-of-the-art method SVM_{PR} [5].

All the algorithms were implemented in C++ and compiled using GCC 4.2.3 with -O3 flag. All the experiments were run on eight Ubuntu machines with an Intel Core E5420 2.5 GHz CPU and 4 GB memory.

4.1 Effect on ℓ

We evaluate the effect on ℓ (the number of selected partitions). We partitioned the patents into 24 partitions and evaluated the effectiveness and efficiency by varying ℓ. To evaluate the result quality, we used the well-known metrics precision and p@k, where the precision is the ratio of the number of retrieved relevant results to the number of retrieved results, and p@k is the precision of the top-k results. Fig. 2a shows the results.



(a) Quality (b) Efficiency

Fig. 2. Effect on ℓ.

We can see that with the increase of ℓ, the precision first increased and then slowed down to a stable level. The main reason is as follows: First, the more patents used to answers a query, the more relevant answers, and thus the higher precision. Second, as each query usually belongs to limited number of topics, only several partitions are relevant to the query. Thus, the precision is stable when ℓ is large enough. For example, where $\ell > 10$, they achieved nearly the same precision.

Then, we evaluated the performance. Fig. 2b shows the results. We see that with the increase of ℓ, the elapsed time increased. This is because the more data used to answer a query, the higher overhead to find answers.

In the remainder of the experiments, we set $\ell = 10$.

4.2 Quality Comparison

In this section, we compare the result quality. Table 1 shows the experimental results.

We can see that our three techniques can improve the result quality. For example, for p@50, error correction improves the precision from 0.61 to 0.77, query suggestion increases the precision to 0.83, and query expansion can improve the precision to 0.74. More importantly, our method by combing the three methods can improve the precision to 0.88, and the improvement ratio is about 44.2 percent. The main reason is as follows: First, the automatic error correction can provide users accurate keywords based on users inputs. Second, the query expansion can suggest relevant keywords. Third, topic-based query suggestion can provide users topic-relevant keywords.

4.3 Efficiency Comparison

In this section, we compare the efficiency. We partitioned the data into 24 partitions. We used eight computes to manage the data and each node managed three partitions. For each partition, we built the corresponding indexes. We first compared the two methods by varying the number of keywords. Fig. 3a shows the results. We can see that for different numbers of keywords, our method always outperformed the existing method SVM_{PR}, with the speedup ratio about 8. This reflects that our method achieved high efficiency since we employ an effective partition-based method.

We then evaluated the two methods by varying the number of returned answers. Fig. 3b shows the results. We can see that for different values, our method always outperformed SVM_{PR}. This is because, we partition the data into eight partitions and each partition was indexed and searched by different cores. More importantly, our partition-based method can prune many irrelevant patents and thus can improve the performance significantly.

4.4 Scalability

We also evaluate the scalability of our method. Fig. 4 shows the experimental results by varying the number of patents. Fig. 4a shows the scalability on quality and Fig. 4b illustrates the results on efficiency. We can see that with the increase of the number of

5. http://www.uspto.gov/.

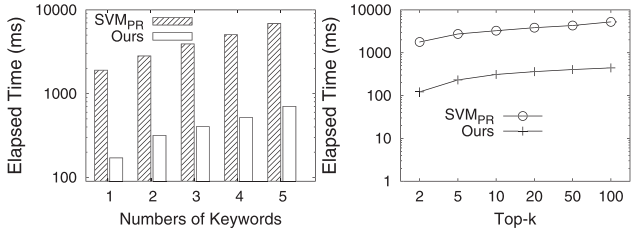
TABLE 1
Result Quality Comparison

p@n	SVM_{PR} [5]	Error Correction	Query Suggestion	Query Expansion	All
2	0.52	0.68	0.78	0.70	0.84
5	0.55	0.72	0.79	0.71	0.85
10	0.58	0.74	0.77	0.72	0.86
20	0.60	0.75	0.80	0.76	0.87
50	0.61	0.77	0.83	0.74	0.88
100	0.60	0.76	0.82	0.75	0.87

patents, the precision of query suggestion and query expansion increased slightly. This is because we can utilize more data to select the topic of each patent and find more relevant keyword pairs. The more data used, the higher precision of the topic model. For query correction, the precision nearly kept the same as we only used the trie structure to correct the keywords. On the other hand, our method scaled very well.

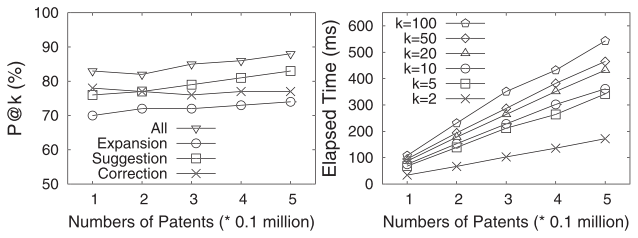
5 Related Works

Larkey [7] studied the problem of patent classification; however, the paper neglected the prior-art search (novelty search). Guo and Gomes [5] proposed SVM patent ranking model to improve the search quality. Xue and Croft [14] studied how to automatically transform a query patent into a search query and use the search query to find answers. They focused on how to extract query words from patents, how to weight them and whether to use noun-phrases. Our problem is different from theirs as we focus on improving efficiency and quality to answer a keyword query. Azzopardi et al. [1] surveyed eighty patent analysts in order to obtain a better picture of their search habits, preferences, and the types of functionality, and gave some findings from this survey. Magdy et al. [12] discussed two approaches for the patent prior-art search. The first approach is a simple method with low-resources requirement, and the second one is a sophisticated method, using an advanced level of content analysis. Bashir and Rauber [2] evaluated the coverage of prior-art queries extracted from query patents using retrievability measurement. Different from existing studies, we propose a user-friendly patent search paradigm.



(a) Varying keyword number (b) Varying k

Fig. 3. Efficiency comparison.



(a) Quality (b) Efficiency

Fig. 4. Scalability.

6 Conclusion

In this paper, we proposed a new patent-search paradigm. We developed three effective techniques, error correction, topic-based query suggestion, and query expansion, to make patent search more user friendly and improve user search experience. Error correlation can provide users accurate keywords and correct the typing errors. Topic-based query suggestion can suggest topically coherent keywords as users type in query keywords. Query expansion can suggest synonyms and those relevant keywords of query keywords which are in the same concept with query keywords. We proposed a partition-based method to improve the search performance. Experimental results show that our method achieves high efficiency and quality.

References

[1] L. Azzopardi, W. Vanderbauwhede, and H. Joho, "Search System Requirements of Patent Analysts," *Proc. 33rd Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR)*, pp. 775-776, 2010.
[2] S. Bashir and A. Rauber, "Improving Retrievability of Patents in Prior-Art Search," *Proc. European Conf. Information Retrieval (ECIR)*, pp. 457-470, 2010.
[3] D.M Blei, A.Y Ng, and M.I Jordan, "Latent Dirichlet Allocation," *J. Machine Learning Research*, vol. 3, pp. 993-1022, 2003.
[4] J. Fan, H. Wu, G. Li, and L. Zhou, "Suggesting Topic-Based Query Terms as You Type," *Proc. Int'l Asia Pacific Web Conf. (APWEB)*, pp. 61-67, 2010.
[5] Y. Guo and C.P. Gomes, "Ranking Structured Documents: A Large Margin Based Approach for Patent Prior Art Search," *Proc. Int'l Joint Conf. Artificial Intelligence (IJCAI)*, pp. 1058-1064, 2009.
[6] S. Ji, G. Li, C. Li, and J. Feng, "Efficient Interactive Fuzzy Keyword Search," *Proc. Int'l Conf. World Wide Web (WWW)*, pp. 371-380, 2009.
[7] L.S. Larkey, "A Patent Search and Classification System," *Proc. Fourth ACM Conf. Digital Libraries*, pp. 179-187, 1999.
[8] C. Li, J. Lu, and Y. Lu, "Efficient Merging and Filtering Algorithms for Approximate String Searches," *Proc. Int'l Conf. Data Eng. (ICDE)*, pp. 257-266, 2008.
[9] G. Li, J. Feng, and C. Li, "Supporting Search-As-You-Type Using SQL in Databases," *IEEE Trans. Knowledge and Data Eng.*, vol. 25, no. 2, pp. 461-475, Feb. 2013.
[10] G. Li, S. Ji, C. Li, and J. Feng, "Efficient Fuzzy Full-Text Type-Ahead Search," *VLDB J.*, vol. 20, no. 4, pp. 617-640, 2011.
[11] G. Li, B.C Ooi, J. Feng, J. Wang, and L. Zhou, "Ease: An Effective 3-in-1 Keyword Search Method for Unstructured, Semi-Structured and Structured Data," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, pp. 903-914, 2008.
[12] W. Magdy, P. Lopez, and G.J.F Jones, "Simple vs. Sophisticated Approaches for Patent Prior-Art Search," *Proc. European Conf. Advances in Information Retrieval*, pp. 725-728, 2011.
[13] Q.H Vu, B.C Ooi, D. Papadias, and A.K.H Tung, "A Graph Method for Keyword-Based Selection of the Top-K Databases," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, pp. 915-926, 2008.
[14] X. Xue and W.B Croft, "Automatic Query Generation for Patent Search," *Proc. ACM Conf. Information and Knowledge Management (CIKM)*, pp. 2037-2040, 2009.
[15] B. Yu, G. Li, K.R Sollins, and A.K.H Tung, "Effective Keyword-Based Selection of Relational Databases," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, pp. 139-150, 2007.