Automatic Query Generation for Patent Search

Xiaobing Xue
Center for Intelligent Information Retrieval
Computer Science Department
University of Massachusetts, Amherst, MA, 01003, USA
xuexb@cs.umass.edu

W. Bruce Croft
Center for Intelligent Information Retrieval
Computer Science Department
University of Massachusetts, Amherst, MA, 01003, USA
croft@cs.umass.edu

ABSTRACT

Patent search is the task of finding relevant existing patents, which is an important part of the patent's examiner's process of validating a patent application. In this paper, we studied how to transform a query patent (the application) into search queries. Three types of search features are explored for automatic query generation for patent search. Furthermore, different types of features are combined with a learning to rank method. Experiments based on a USPTO patent collection demonstrate that the single best search feature is the combination of words and noun-phrases from the summary field and the retrieval performance can be significantly improved by combining three types of search features.

Categories and Subject Descriptors
H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval

General Terms
Algorithms, Experimentation, Performance

Keywords
Prior-art Search, Patent Retrieval, Learning to Rank, Information Retrieval

1. INTRODUCTION

Since patents play an important role in Intellectual Property protection, patent search has attracted considerable attention recently. Prior-art search, one of the main types of patent retrieval, helps the patent examiners to find previously published relevant patents when they need to validate or invalidate a patent application. Compared with other search tasks, prior-art search has its own unique properties. First, different from a technical report, the writers of a patent will focus on how to extend the coverage of their patent or to emphasize novel aspects of the patent, but not how to help the reader understand the techniques easily. Therefore, it is not unusual to see vague expressions and non-standard terminology in a patent. Second, prior-art search is a recall-oriented retrieval task, where not missing a relevant document is more important than retrieving a relevant document at the top rank. Usually, in prior-art search, the patent examiners will carefully examine the first 100 or 200 documents retrieved by the search engine instead of browsing just the top few results. These properties mean that information retrieval models that work well with TREC collections may not be applicable in the patent environment.

Previous work on prior-art search focused on developing the retrieval model, but ignored the generation of a search query from the query patent. Most of this research used the claims extracted from the query patent as the search query without carefully considering whether it is the best choice. Larkey [1] has studied how to transform a patent into a query for patent classification, however this approach has not been explored carefully for prior-art search. Our previous work [2] started to work on this problem and found the summary part of a patent is a much better source for generating query words for prior-art search than the claim part. In this paper, we will explore more factors in automatic query generation and attempt to combine different search features with learning to rank methods. Furthermore, we will report on recall-oriented performance measures, since they are more interesting for the patent task.

In the next section, we briefly describe related work. Then, in section 3, we present the types of features that are used in the experiments. Section 4 discusses how these features are combined. Section 5 presents the experimental results and discusses them.

2. RELATED WORK

Larkey [1] studied the problem of patent classification instead of prior-art search. However, due to the different properties of these two tasks, observations from patent classification are not necessarily valid for prior-art search, and sometimes even opposite conclusions can be reached. For example, the experiments in this paper show that words from the title field are the least useful for prior-art search, while Larkey assigned more weight to the title for patent classification. Osborn et al. [3] reported some early results about patent retrieval on a subset of the USPTO collection. With phrases as the indexed features, they observed some improvement on retrieval performance. Takaki et al. [4] analyzed the claim part of the patent and discovered query subtopics. The final score of a retrieved document was a weighted combination of the scores of using each extracted query topic as the query. Mase et al. [5] proposed a two-stage strategy for patent retrieval. In the first stage, the claim part of the query patent was used to retrieve the top

CIKM'09, November 2–6, 2009, Hong Kong, China.

ALGORITHM: Transforming Patent to Query
INPUT: *Patent*, *Num*, *Field*, *Weight*, *NP*
OUTPUT: *Query*
PROCESS: Rank words in *Field* according to their *tfidf* scores and then select *Num* top ranked words as the query words. Assign *Weight* to each query word to get $Query_w$. Repeat the above steps for noun-phrases to get $Query_{np}$. If *NP* is true, set *Query* as the combination of $Query_w$ and $Query_{np}$; otherwise set *Query* as $Query_w$.

Figure 1: General algorithm for transforming the query patent to an effective search query.

1,000 patents. In the second stage, several techniques were used to rerank the top 1,000 patents. In a recent paper, Fujii [6] applied link analysis techniques to the citation structures of the patent collection. After combining the citation-based scores with the text-based scores for patents, better performance than only using the text information was achieved.

The NTCIR patent retrieval track[1] has become one of the most important platforms for comparing the retrieval performance of different systems and testing new ideas. The query topics are patents and the queries used are one or more claims of the query patent. In NTCIR4, expert judgments were used as the relevance data. Due to the cost, only 34 query topics were developed. In NTCIR5 and NTCIR6, a patent's citations were used as relevance judgments and thousands of query topics were developed automatically.

"Learning to rank" techniques have attracted considerable attention in recent years. The basic idea is to learn the retrieval model with machine learning techniques. Several machine learning techniques have been used, such as Ranking SVM [7], RankBoost [8] and RankNet [9]. Recently, a boosting style method, AdaRank [10] was proposed which considered the query as the basic unit and has the property of directly optimizing the performance measure used for the search task.

3. FEATURES FOR PRIOR-ART SEARCH

3.1 Retrieval-Score Features

This type of feature focuses on how to transform the query patent into an effective search query. The retrieval score for the transformed search query will be used as the feature. In this paper, we use Indri [11] as the retrieval model, thus different retrieval-score features differ on how the query patent is transformed into an Indri query.

To design a transforming method, we need to consider several factors: *Num*, how many query words should be kept, *Field*, where to extract query words; *Weight*, which weighting methods are used for query words; *NP*, whether to use noun-phrases as a complement. A general transforming algorithm is provided in Fig. 1, where the discussed factors are used as parameters.

For *Num*, we consider words from 10-100. For *Field*, we consider six fields of a patent with explicit tags, the title field (ttl), the abstract field (abst), the brief summary field (bsum), the description of the figures(drwd), the detailed text description field (detd) and the claim field (clms). Besides them, we also extract the primary claim field (pclms), which is the most important claim in the claim field. Also, we consider the case that the query words or noun-phrases are extracted from the whole patent (all), which ignores the structure information. For *Weight*, we consider to use the equal weight (*bool*), use the term frequency (*tf*) and use the combination of term frequency and the inverted document frequency (*tfidf*). For *NP*, we consider to use noun-phrase (true) or not (false).

[1] http://research.nii.ac.jp/ntcir/publication1-en.html

Table 1: Equations used to calculate low-level features. 'nor' denotes the normalized value.

L1	$\sum_{w_i \in q_i \cap d_j} \delta_i \cdot c(w_i, d_j)$	*tf*
L2	$\sum_{w_i \in q_i \cap d_j} \delta_i \cdot \frac{c(w_i,d_j)}{\lvert d_j \rvert}$	nor(*tf*)
L3	$\sum_{w_i \in q_i \cap d_j} \delta_i \cdot \log(c(w_i, d_j) + 1)$	log(*tf*)
L4	$\sum_{w_i \in q_i \cap d_j} \delta_i \cdot \log(\frac{\lvert C \rvert}{df(w_i)})$	*idf*
L5	$\sum_{w_i \in q_i \cap d_j} \delta_i \cdot c(w_i, d_j) \log(\frac{\lvert C \rvert}{df(w_i)})$	*tfidf*
L6	$\sum_{w_i \in q_i \cap d_j} \delta_i \cdot \frac{c(w_i,d_j)}{\lvert d_j \rvert} \log(\frac{\lvert C \rvert}{df(w_i)})$	nor(*tf*)*idf*
L7	$\sum_{w_i \in q_i \cap d_j} \delta_i \cdot \log(c(w_i, d_j) + 1) \log(\frac{\lvert C \rvert}{df(w_i)})$	log(*tf*)*idf*

Table 2: Summary of the category features. Q denotes the query patent and D denotes the patent in the collection.

	Q	D		Q	D
C1	<OCL>	<OCL>	C6	<XCL>	<FSC>
C2	<OCL>	<XCL>	C7	<FSC>	<OCL>
C3	<OCL>	<FSC>	C8	<FSC>	<XCL>
C4	<XCL>	<OCL>	C9	<FSC>	<FSC>
C5	<XCL>	<XCL>	C10	<ICL>	<ICL>

3.2 Low-Level Features

Some useful information from the query patent can't be expressed using Indri queries, so we designed two other types of features. The first type corresponds to important statistics used in information retrieval, such as *tf*, *idf*, *tf.idf* or their variants. Similar features have been used by previous work in learning to rank [12]. Table 1 shows the seven types of low level features used. In Table 1, q_i is the search query transformed from the query patent, w_i is the query word, δ_i is the weight of w_i and d_j is the document(patent) in the collection. $c(w_i, d_j)$ denotes the number of times w_i appears in d_j, $|d_j|$ denotes the total number of words in d_j, $|C|$ denotes the total number of documents in the collection and $df(w_i)$ denotes the number of documents where w_i has appeared.

3.3 Category Features

The other type of feature that is difficult to represent using Indri queries are the category features, which use the class information in the patent. The patent fields <OCL>, <XCL>, <FSC/FSS> describe the primary class codes, the secondary class codes, and the related class codes of the US classification system, respectively. The <ICL> field indicates the class codes of the international classification system. A category feature can be defined as the similarity of the category fields between the query patent and the patents in the collections. Based on the combinations of different types of category fields, 10 category features are obtained, which are shown in Table 2. The similarity of two category fields is decided by whether they share the same class code.

4. FEATURE COMBINATION

Since we linearly combine different features, it is essential to decide the combination weights. The citation field of a patent <UREF> can be used as a substitute for manual relevance judgments for training. Thus, a training set with thousands of queries can be easily constructed. Given this

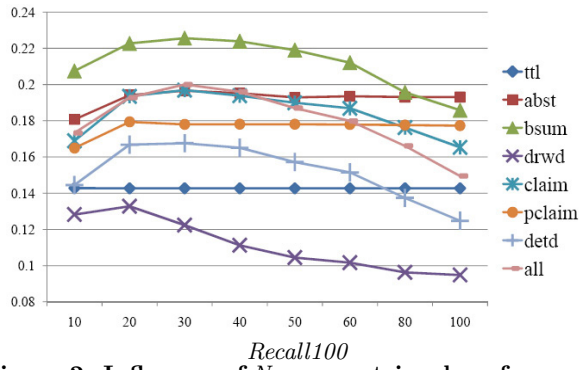

Figure 2: Influence of *Num* on retrieval performance.

large scale training set, we use learning to rank techniques to help "learn" the combination weights. The expectation is that the weights that work well on the training set will also generate reasonable performance on unseen queries.

AdaRank is used to combine different features in this paper. In the training phase, at each round, AdaRank selects one feature from the feature set based on its retrieval performance on the training set and automatically calculates its combination weight. Generally, the feature selected at each round will focus on the "hard" queries, where previously selected features do not perform well. In the predicting phase, the selected features are combined to predict the rank of documents for unseen queries.

5. EXPERIMENTS

5.1 Corpus

The same corpus as our previous work [2] is used. The USPTO corpus consists of 1,604,386 patents published from 1980 to 1997. We used the patents published from 1980 to 1996 as the test collection, and the patents published in 1997 as candidates for the query set. We restricted the query patents to have at least 20 citations and all the types of fields we are interested in. The final size of the query set is 3,736, which we randomly split it into a training set with 3,361 patents and a test set with 373 patents. Since it is extremely difficult, or even impossible, to get relevance judgments from experts for those thousands of queries, we use a patent's citation field <UREF> as a substitute, which is the same strategy adopted by NTCIR5-6. Indri is used to index the full text of patents in the collection. The Krovetz stemmer is also used. The mean average precision (MAP) and the recall at the first 100 documents (Recall100) are reported. Two-tailed t-tests are conducted to decide statistical significance.

5.2 Effect of Single Search Features

5.2.1 *Effect of Retrieval-Score Features*

For the retrieval-score features, as mentioned in Section 3.1, we explore retrieval performance with different values of the parameters *Num*, *Field*, *Weight* and *NP*.

First, we test eight different values for *Num*[2]. The *Weight* is set to *bool* and *NP* is set to false. The results are shown in Fig. 2[3].

[2]The eight different values are 10, 20, 30, 40, 50, 60, 80, 100.
[3]Due to the space limit, we only show the results on Recall100. The results on MAP is very similar

Table 3: Influence of *Weight* on retrieval performance.

	MAP			Recall100		
Field	*bool*	*tfidf*	*tf*	*bool*	*tfidf*	*tf*
ttl	0.042	0.039	0.043	0.143	0.129	0.144
drwd	0.044	0.048	0.047	0.133	0.144	0.143
detd	0.055	0.057	0.066*	0.167	0.171	0.189*
pclms	0.059	0.062	0.055	0.179	0.183	0.167
clms	0.066	0.066	0.064	0.194	0.195	0.187
abst	0.066	0.070	0.074*	0.194	0.195	0.207*
all	0.067	0.068	0.078*	0.193	0.198	0.215*
bsum	0.078	0.082	0.094*†	0.223	0.231	0.252*†

Table 4: Influence of *NP* on retrieval performance.

	MAP		Recall100	
Field	w	w+p	w	w+p
ttl	0.043	0.042	0.144	0.137
drwd	0.047	0.048	0.143	0.145
detd	0.066	0.066	0.189	0.187
pclaim	0.055	0.056	0.167	0.168
claim	0.064	0.066*	0.187	0.191
abst	0.074	0.074	0.207	0.208
all	0.078	0.080*	0.215	0.219*
bsum	0.094	0.096*	0.252	0.256

Fig. 2 shows that, for most fields, the best performance is obtained with 20 or 30 words. With more words, the change in the performance is not significant, but the time spent on search is significantly increased. For the title field, 10 words are enough, since the titles of most patents contain less than 10 words. Thus, in the following experiments, *Num* is set to 10 for the title field and 20 for other fields, considering the balance between accuracy and efficiency.

Second, we explore the parameter *Field* and the parameter *Weight*. *NP* is also set to false. The results are shown in Table 3[4]. * denotes *tf* is significantly different with both *bool* and *tfidf*. † denotes 'bsum+tf' is significantly different with all other field and weight combinations.

The Recall100 part of Table 3 verifies the results we obtained in our previous work [2]: for *Weight*, *tf* is the best weighting method; for *Field*, the summary field is much better than the claim field. Also, the improvement of bsum+*tf* is more obvious on Recall100 than on MAP.

Third, we want to explore *NP*. *Weight* is set as *tf*. The results for each field type are displayed in Table 4. 'w' denotes only using words; 'w+p' denotes combining words and noun phrases with weights 0.8 and 0.2. * denotes significantly different with the performance of 'w'.

Table 4 shows that, in most cases incorporating noun phrases improves performance slightly.

5.2.2 *Performance of Low-Level Features*

Since the low-level features do not give good retrieval performance on their own, we use them to rerank the top 1000 documents returned by bsum+*tf*. Fig. 3 shows the performance of L1-L7(see Table 1) .

Fig. 3 shows that the low-level features do not improve the performance of 'bsum', which is used to decide the initial rank list of documents. Among low-level features, L3 ($\log(tf)$), L4 (idf) and L7 ($\log(tf)idf$) are better than the others.

[4]The MAP part of this table has been reported in our previous work [2]. For completeness, it is also reported here.

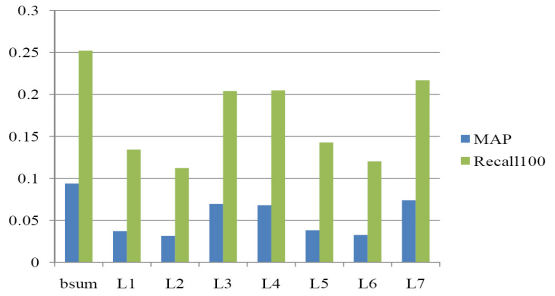

Figure 3: Retrieval performance of low-level features.

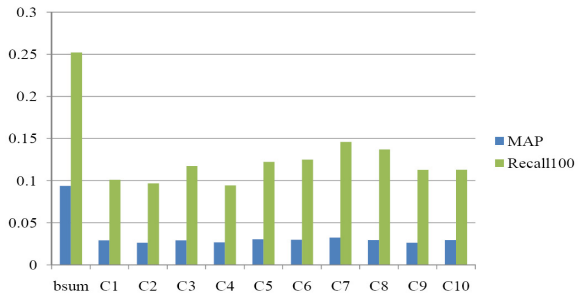

Figure 4: Retrieval performance of category features.

5.2.3 *Category Features*

Similar to the low-level features, we also use the retrieval-score feature 'bsum' to obtain the initial rank and then use the category features to rerank the retrieved documents. Fig. 4 compares C1-C10 (see Table 2) with 'bsum'.

Fig. 4 shows that the category features perform much worse than 'bsum'. Some potential reasons are: the class code is too broad and the class code hierarchy is changed every year.

5.3 Combining Different Types of Features

We use AdaRank to combine different types of features. The experiments are conducted on seven different feature sets: Ret, Low, Cat, Ret+Low, Ret+Cat, Low+Cat and Ret+Low+Cat (All). The performance of SingleBest and AdaRank are compared over each feature set. Results can be found in Table 5. * denotes significantly different with the SingleBest. α, β and γ denote significantly different with the performance of AdaRank on Ret, Ret+Low and Ret+Cat, respectively.

Table 5 shows that the AdaRank combination performs significantly better than SingleBest on most feature sets. This is especially true when all candidate features are used, AdaRank can improve SingleBest by 12.5% on MAP (0.108/0.096) and by 13.3% on Recall100 (0.290/0.256). We further compare the performance of AdaRank on Ret, Ret+Low, Ret+Cat and All. AdaRank performs significantly better on Ret+Low and Ret+Cat than on Ret. This observation shows that although Low and Cat features are not as strong as Ret features, they can provide some complementary information to Ret features. After we put all features together, All performs significantly better than Ret+Low and All can also bring some improvement on Ret+Cat but not significant. The above observations show that the benefits of combining all features together mainly come from the complementarity of Ret and Cat features.

Table 5: Performance of different combination techniques.

Feature Set	MAP		Recall100	
	SinBest	AdaR	SinBest	AdaR
Ret$^{\alpha}$	0.096	0.101*	0.256	0.268*
Low	0.074	0.078	0.217	0.227
Cat	0.033	0.061*	0.146	0.203*
Ret+Low$^{\beta}$	0.096	0.104*$^{\alpha}$	0.256	0.277*$^{\alpha}$
Ret+Cat$^{\gamma}$	0.096	0.108*$^{\alpha}$	0.256	0.287*$^{\alpha}$
Low+Cat	0.074	0.088	0.217	0.255*
All	0.096	0.108*$^{\alpha\beta}$	0.256	0.290*$^{\alpha\beta}$

6. CONCLUSION

Prior-art search is an important task of patent retrieval. In a departure from previous work, we focus on how to automatically transform a query patent into a search query. After exploring different factors of a successful transformation, we provide answers to how many query words should be used, where to extract query words, how to weight them and whether to use noun-phrases. Furthermore, we show that combining different features can significantly improve retrieval performance. In the future, working on more reliable relevance judgments is an important issue.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant #IIS-0711348 and the Information Retrieval Facility (IRF). Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

7. REFERENCES

[1] Larkey, L.: A patent search and classification system. In: DL99, Berkeley, CA (1999) 179–187
[2] Xue, X., Croft, W.B.: Transforming patents into prior-art queries. In: SIGIR09, Boston, MA (2009) 808–809
[3] M. Osborn, T.S., Marinescu, M.: Patent database: a preliminary report. In: CIKM97, Las Vegas, NV (1997) 216–221
[4] T. Takaki, A.F., Ishikawa, T.: Associative document retrieval by query subtopic analysis and its application to invalidity patent search. In: CIKM04, Washington, DC (2004) 399–405
[5] H. Mase, T.M., Ogawa, Y.: Proposal of two-stage patent retrieval method considering the claim structure. ACM Transactions on Asian Language Information Processing **4**(2) (2005) 186–202
[6] Fujii, A.: Enhancing patent retrieval by citation analysis. In: SIGIR07, Amsterdam, the Netherlands (2007) 599–606
[7] R. Herbrich, T.G., Obermayer, K.: Large Margin rank boundaries for ordinal regression. MIT Press: Cambridge, MA (2000)
[8] Y. Freund, R. D. Iyer, R.E.S., Singer, Y.: An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research **4** (2003) 933–969
[9] C. Burges, T. Shaked, E.R.A.L.M.D.N.H., Hullender, G.: Learning to rank using gradient decent. In: ICML05, Bonn, Germany (2005) 89–96
[10] Xu, J., Li, H.: Adarank: a boosting algorithm for information retrieval. In: SIGIR07, Amsterdam, the Netherlands (2007) 391–398
[11] Metzler, D., Croft, W.: Combining the language model and inference network aprroaches to retrieval. Information Processing and Management Special Issue on Bayesian Networks and Information Retrieval **40**(5) (2004) 735–750
[12] T.-Y. Liu, J. Xu, T.Q.W.X., Li, H.: Letor: Benchmark dataset for research on learning to rank for information retrieval. In: LR4IR 2007, in conjunction with SIGIR 2007. (2007)