Automatic Boolean Query Suggestion for Professional Search

Youngho Kim
yhkim@cs.umass.edu

Jangwon Seo
jangwon@cs.umass.edu

W. Bruce Croft
croft@cs.umass.edu

Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts Amherst, MA 01003, USA

ABSTRACT
In professional search environments, such as patent search or legal search, search tasks have unique characteristics: 1) users interactively issue several queries for a topic, and 2) users are willing to examine many retrieval results, i.e., there is typically an emphasis on recall. Recent surveys have also verified that professional searchers continue to have a strong preference for Boolean queries because they provide a record of what documents were searched. To support this type of professional search, we propose a novel Boolean query suggestion technique. Specifically, we generate Boolean queries by exploiting decision trees learned from pseudo-labeled documents and rank the suggested queries using query quality predictors. We evaluate our algorithm in simulated patent and medical search environments. Compared with a recent effective query generation system, we demonstrate that our technique is effective and general.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – Query Formulation, Search Process.

General Terms
Algorithms, Experimentation.

Keywords
Boolean query suggestion, prior-art search, patentability search.

1. INTRODUCTION
Query suggestion is an effective and practical way to help users formulate queries [15, 16]. While there have been many studies on how to provide alternative queries in general web search [15, 16], little work has been done about suggestion for domain-specific search, e.g., patent retrieval, legal search, and medical information search. Many of the users in such domains are search professionals, e.g., patent examiners and information specialists in companies and law firms, who perform specialized search tasks such as prior-art search and legal discovery. Query suggestion techniques should be designed for the unique search characteristics of these domains. For example, professional search is typically more recall-oriented than consumer search. In the patent validity task, for example, patent examiners formulate search queries from a new patent to validate its patentability, and generally spend about 12 hours to complete a single task by examining approximately about 100 patent documents retrieved by 15 different queries on average [1]. Another typical characteristic of professional search is the need to document the searches that are carried out.

For a number of reasons, both historic and technical, Boolean queries are particularly common in professional search. For example, in patent search, recent surveys [1, 2] revealed that the use of Boolean operators is one of the most important features to formulate effective queries from the perspective of patent professionals. Also, according to [2], most patent professionals who participated in the survey did not regard query term weighting and query expansion as important whereas 96.3% of participants agreed that Boolean operators are necessary. This is not because Boolean queries are the most effective. In fact, a number of studies over the years (e.g., [5, 6, 7, 9, 11]) have shown that "keyword" queries are often significantly more effective. Boolean queries, however, are easy for information professionals to manipulate and are essentially self-documenting in that they define precisely the set of documents that are retrieved.

Despite the importance of Boolean queries in professional search, there has not been much research on helping information professionals formulate those queries. Tseng and Wu [3] indicated that the provision of suggested search vocabulary would be helpful in patent search. Other studies on prior-art search that automatically generate queries from patent text (e.g., [6, 7]) did not investigate Boolean query suggestion. Current government or commercial patent search systems [1] used by information professionals all support Boolean queries but not query suggestion.

In this paper, we propose a method to suggest Boolean queries for professional search. We define a Boolean query as the sequence of terms associated by conjunction (AND) where each term can be prefixed by negation (NOT). Although the OR operator is often used by professionals to indicate synonym groups, the retrieval evidence shows that AND and NOT have much more impact on effectiveness in domains such as patent search with very detailed documents (e.g., [4]). Adding synonym structure is left for future work. Although the suggested Boolean queries can be generated and used with any search engine, we use a simple statistical Boolean retrieval model for our experiments (explained in Section 5). We do not adopt any additional query processing and term weighting because those features are not generally preferred by professionals and not supported by commercial search systems.

SIGIR'11, July 24–28, 2011, Beijing, China.

[1]PATENT SCOPE (http://www.wipo.int/patentscope/), PatFT (http://patft.uspto.gov/), DELPHION (http://www.delphion.com/)

In order to suggest Boolean queries, we first focus on generating Boolean queries that describe the content of an initial set of retrieved documents. We then rank the generated queries to place effective (in terms of finding more relevant documents) queries at higher ranks. In other words, our system performs two sub-tasks: 1) Boolean Query Generation, and 2) Boolean Query Ranking. In the first task, we extract queries composed of Boolean operators and various terms, representing a pseudo-relevant document set, i.e., the top-*k* documents retrieved by a baseline system. To do this, we learn a decision tree from the pseudo-relevant documents so that the decision tree can determine whether a document is pseudo-relevant or not. Afterwards, each positive decision rule (i.e., a path from the root to a positive leaf node indicating pseudo-relevance) formulates a Boolean query. Our Boolean query generation process based on decision trees has two advantages: i) a (binary) decision tree can be equivalent to a Boolean function in terms of its expressiveness [20], and ii) decision trees naturally determine the number of query terms.

In the next step, among the many generated Boolean queries, we select the effective ones by ranking them. This is consistent with typical query suggestion techniques used in web search, where the "best" suggestions are presented to the user. We learn to rank the generated queries using various quality predictors proposed for adhoc retrieval [21, 22, 23, 24, 25] and several new features that consider unique properties of Boolean queries.

We show the effectiveness of our Boolean query suggestion system by verifying that the system is capable of generating effective Boolean queries at high ranks. Although our focus is on patent search, we show the generality of our approach through additional experiments with a medical literature database.

The rest of this paper is organized as follows. In Section 2, we outline previous work in query suggestion and generation, and describe the limitations of previous work. Section 3 defines the task of Boolean query suggestion for professional search, and we present the methods we used in Section 4. Section 5 contains the experimental results and discussion. Finally, we summarize the contributions of our research and future work in Section 6.

2. RELATED WORK AND LIMITATIONS

In this section, we explore previous work for query suggestion for web search and query generation.

Automatic query expansion [13] has been intensively researched to bridge the gap between users' queries and relevant documents. In particular, pseudo-relevance feedback [12] is known as one of the most effective techniques. Although many successful query expansion techniques have been proposed (e.g., [13, 14]), most of them are not easily applicable to our tasks because they are not able to generate Boolean queries required for professional search environments. However, the query expansion method proposed by Mitra et al. [13] is strongly related to our work because they addressed the effectiveness of Boolean filters to improve precision of automatic query expansion. Specifically, they manually formulate fuzzy Boolean operators (conjunction and disjunction) and select expanded terms from a set of pseudo-relevant documents refined by the Boolean filters. However, their work is limited in that the Boolean filters are manually constructed while we focus on automatic formulation. Moreover, they did not consider Boolean queries.

Jones et al. [17] proposed a query substitution system that suggests strongly related queries identified from user query sessions. In query reformulation, Wang and Zhai [18] discovered associated terms from query logs to substitute the original query terms or add new terms into an original query. Also, query recommendation techniques proposed in [16] provide alternatives by clustering related queries in query logs, and White et al. [15] studied types of query suggestions for web search preferred by users via a user study. While the query logs and session information that most previous techniques depend on are readily available for web search, such resources are mostly not available in domain-specific search environments that we address.

In patent search, although automatic query generation is crucial, most previous work has focused on improving a retrieval model. The participants in the patent retrieval task of NTCIR-6 [8] used terms from claim sections without term selection. Mase and Iwayama [9] added terms from abstract sections into claim-based queries and weighted the query terms by their TF-IDF scores in the query patent. Xue and Croft [6, 7] described a query generation technique for patent search. In [6], they generated a patent search query by selecting an effective section and extracting the top-ranked words from the section using TF-IDF weights. Their finding is that the "brief summary" section of patents can produce the most effective queries, while weighting query words by term frequencies is more effective than TF-IDF weighting. They also expanded search queries to incorporate noun phrases, but the improvement was not significant. In [7], they improved retrieval performance of queries generated from [6] by using a learning-to-rank model and various features. The queries generated by this approach, however, contain many terms that are weighted and have other constraints, making them unsuitable for query suggestion. Similar to the NTCIR workshop, TREC recently proposed the Chemical track. A sub-task of the track addresses search for chemical patents [10]. Though Gobeill et al. [11] considered query expansion and showed the best performance among the participants, their expansion technique is somewhat limited in that it depends on a concept identifier and external resources (PubChem[2]) which would not be applied to general patents.

The Boolean query suggestion approach is based on our previous work with professional searchers and studies such as Bache and Azzopardi [4]. They pointed out that patent searchers are in favor of exact-match models, i.e., traditional Boolean retrieval, because Boolean models can improve the retrievability of target documents. Accordingly, they proposed hybrid retrieval models in which target patents are ranked by TF-IDF or Okapi BM25 models and then filtered by conjunctive and disjunctive Boolean operators (AND and OR). Though their models are empirically effective, those are limited in using only two-word queries, i.e., two words are associated by conjunction or disjunction.

In the medical domain, Hashmi et al. [19] developed a system to generate context-specific queries from clinical guidelines, but they also used an external knowledge base to create query terms.

3. PROBLEM FORMULATION

In this section, we formulate the Boolean query suggestion problem for professional search and define associated terms.

[2] The database of compounds structure and description for chemical molecules (http://pubchem.ncbi.nlm.nih.gov/)

Definition 1 (Professional Search): *Professional search* is interactive information retrieval performed by professionals in a specific domain such as the patent or medical domain. The main difference of this task from adhoc retrieval (e.g., web search) is that professional search tends to be recall-oriented. The search behavior of professionals is quite different from that of general users. For example, since professionals tend to prefer finding more relevant documents to finding a small number of relevant documents at the top ranks, they examine more retrieval results than web search users. Accordingly, search processes are often long and repeatable, e.g., a patent examiner issues several queries until the obtained results are satisfactory.

Definition 2 (Topic): A *topic* is a subject for which queries against search engines are formed and the retrieval results are examined by professionals. For example, in prior-art search, a new patent to be validated can be a topic. In the medical domain, a description of patient symptoms can be a topic.

Definition 3 (Boolean Query): A *Boolean query* is a sequence of query terms all of which are connected by conjunction and each of which can be prefixed by negation, e.g., $aluminum \wedge \neg tartar \wedge casting\ mold$. In this work, as query term candidates, we consider bigrams as well as unigrams.

Definition 4 (Pseudo-Relevant Documents): *Pseudo-relevant documents* are the top k documents retrieved by a *baseline* system. The *baseline* system can handle a weighted or expanded query using a state-of-the-art retrieval model. We exploit the pseudo-relevant documents to formulate Boolean queries that are suggested to users.

Definition 5 (Boolean Query Generation): *Boolean query generation* is formulating Boolean queries from a set of query term candidates. Using terms appearing in a set of pseudo-relevant documents for topic, we formulate Boolean queries that consist of effective terms and Boolean operators (AND and NOT), where query term candidates can be unigrams or bigrams extracted from the pseudo-relevant documents.

Definition 6 (Boolean Query Ranking): *Boolean query ranking* is determining a preference among generated Boolean queries for a topic with respect to a recall metric, e.g., recall at 100 (R@100). This is necessary for suggesting a reasonable number of effective Boolean queries (e.g., 5~10) to users because many queries can be generated in the Boolean query generation phase. We produce a ranked list of generated Boolean queries where an effective Boolean query should be placed within the high ranks (e.g., top 10).

4. BOOLEAN QUERY SUGGESTION

In this section, we first propose a decision tree-based method for Boolean Query Generation, and then describe a Boolean Query Ranking model using various query quality predictors. Figure 1 demonstrates the overall process of our system. In Boolean Query Generation, we train decision trees using the baseline retrieval result (containing the top-*k* pseudo-relevant documents and beyond-*k* non-relevant documents) and formulate corresponding Boolean queries (BQs) (details in Section 4.1). In Boolean Query Ranking, the ranking model trained from sorted lists of BQs with respect to R@100 can rank the generated BQs by query quality predictors (details in Section 4.2). The top ranked queries are presented as suggestions.

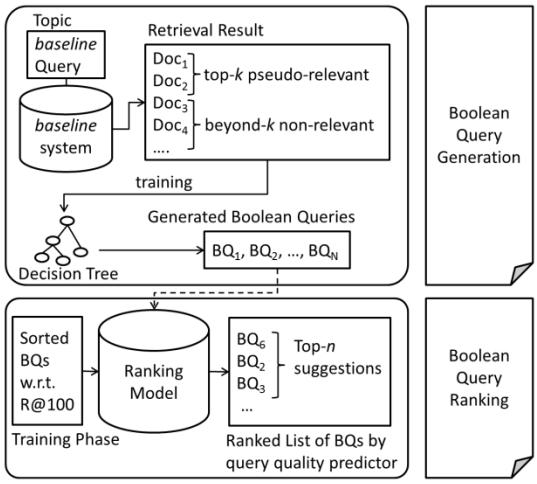

Figure 1: Boolean Query Suggestion System Workflow.

4.1 Decision Tree-based Boolean Query Generation

Binary decision trees are equivalent representations of Boolean functions [20]. If we could train a decision tree where a node corresponds to a term appearing in training documents in order to determine whether a document is relevant to a topic, the learned decision tree could imply a Boolean query representing a set of relevant documents. In addition, the length and query terms of a Boolean query are naturally determined by the depth and the nodes of the tree with reasonable accuracy. A problem, however, is that we do not have training data to learn a tree which can be generalized for every query because each query is associated with a different set of terms. So, instead of relevant documents, we use pseudo-relevant documents defined in Section 3 as training data. In other words, we learn a decision tree by using the top-*k* documents as positive examples. As negative examples, presumably non-relevant documents (ranked beyond-*k* in the baseline retrieval result) are used. Accordingly, Boolean queries generated from the positive nodes of the learned decision tree are expected to be as effective as the baseline query because the decision tree is learned from the pseudo-relevant documents.

Once we learn a decision tree from a topic, we identify a single path from a root to a positive leaf node in the decision tree and convert the rule (path) into a Boolean query. Accordingly, a decision tree produces as many Boolean queries as the number of positive leaf nodes. Figure 2 depicts how to generate Boolean queries from an example decision tree whose attributes (query term candidates) are *alloy*, *wheel*, and *steel*, and **True**/**False** values of each leaf node denotes a positive/negative decision for input documents. For example, a document including *alloy* and *wheel* is classified as **True** (or relevant) because a number of pseudo-relevant examples used for training include the two terms. That is, the path from *alloy* to the first **True** leaf can formulate query Q_1, which is expected to retrieve documents containing *alloy* and *wheel*. Since we concentrate on conjunction and negation, we generate two queries, Q_1 and Q_2, rather than a single unified query such as $(alloy \wedge wheel) \vee (\neg alloy \wedge steel)$ Note that AND and NOT have more impact on the effectiveness and Q_1 or Q_2 is empirically better than the unified query w.r.t R@100 that we use to evaluate a Boolean query.

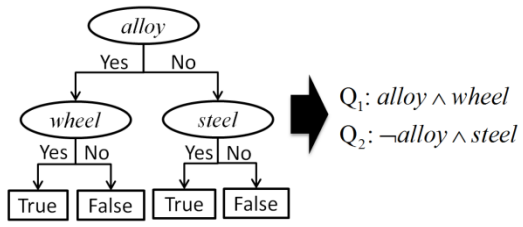

Figure 2: Decision Tree-based Boolean Query Generation.

We describe the Boolean Query Generation algorithm used for the example in the following. Figure 2 shows the process of generating Boolean queries from several sets of query term candidates (i.e., attributes), a set of pseudo-relevant documents (the top-*k* baseline retrieval results) and a set of non-relevant documents (the beyond-*k* baseline retrieval results). To produce a sufficient number of Boolean queries (among which we can find many effective Boolean queries), for each topic, we train several decision trees with different attributes, while all the trees are trained by the same training set. In this, the training set includes the *k* positive (pseudo-relevant) documents and an equal number of negative instances (non-relevant) for a topic. To obtain *N* sets of attributes, various approaches (e.g., thesaurus and concept identification) can be used. In this paper, we change the number of query term candidates in each set to get multiple attribute sets.

For the patent search experiments, in this paper, we select query term candidates from a unigram language model estimated from a topic patent (i.e., new patent) or its pseudo-relevant patent set. We first select unigrams which are likely to be generated from the following language models, assuming that terms are effective for retrieving pseudo-relevant patents if the terms frequently occur in the topic patent or pseudo-relevant patents.

$$P(w|D) \approx \frac{tf(w:D)}{|D|}$$

$$P(w|PRel) \approx \frac{\sum_{d_i \in PRel} tf(w:d_i)}{\sum_{d_i \in PRel} |d_i|}$$

where D is topic patent, $PRel$ is the set of pseudo-relevant patents for topic, $tf(w:d_i)$ indicates a term frequency of a word w in a document d_i, and $|d_i|$ denotes the length of d_i.

Stop-words[3] are removed. Using term probabilities, we can rank terms in a topic document or pseudo-relevant documents. We select the top-*m* terms as attributes for decision trees. In particular, to obtain multiple sets of attributes, we consider 20 different *m*'s (in {5,10,…,100}) , each of which makes an attribute set.

In addition, from an observation that patents contain many compound words that may help to improve the retrievability, we can add an equal number of bigrams into each set of selected unigrams. To rank bigrams, we estimate smoothed bigram language models for the topic patent and the pseudo-relevant patent set as follows:

$$P(w_{i-1}w_i|D) \approx (1-\lambda)\frac{tf(w_{i-1}w_i:D)}{|D|-1} + \lambda P(w_i|D)$$

[3] Stop-words contain articles, prepositions, acronyms (e.g., *fig.*), (relative) pronouns, and general nouns (e.g., *method*, *figure*, and *apparatus*), frequently appeared in patent documents (available at http://www.cs.umass.edu/~yhkim/).

ALGORITHM Boolean Query Generation
INPUT:
- *N* different sets of attributes s.t. { $can_1, can_2, \ldots, can_N$} where can_i is a set of query term candidates
- The baseline retrieval results for a topic, *B*

OUTPUT: A set of Boolean queries, *S*
PROCESS:
- Initialize $S = \{\}$
- For $i = 1,\ldots,N$
  - Construct training data from *B*: the top-*k* documents (positive examples) and *k* documents randomly selected from the beyond-*k* (negative examples)
  - Train a decision tree using the training data and can_i as attributes.
  - Find paths from the root to every positive leaf node in the decision tree and formulate corresponding Boolean queries.
  - Append the Boolean queries to *S*.
- End For
- Return *S*

Figure 3: Boolean Query Generation Algorithm.

$$P(w_{i-1}, w_i|PRel) \approx (1-\lambda)\frac{\sum_{d_i \in PRel} tf(w_{i-1}w_i:d_i)}{\sum_{d_i \in PRel}|d_i| - 1} + \lambda P(w_i|PRel)$$

where λ is a bias to unigrams. We set λ to 0.7 and eliminate bigrams which contain any stop-word.

For the medical search experiments, we use the same setting except that we use only pseudo-relevant documents because topic documents are not usually given. In addition, since long Boolean queries may not return any retrieval results, we eliminate Boolean queries which contain more than 15 terms.

4.2 Boolean Query Ranking

To select a reasonable number of effective queries from a pool of generated Boolean queries, we propose a Boolean query ranking model and introduce features for the model.

4.2.1 Learning-to-Rank Boolean Queries

In order to rank generated Boolean queries, we learn a ranking function which predicts the preference between Boolean queries. That is, given a topic and generated Boolean queries, our ranking model produces a ranked list of the Boolean queries in descending order of recall at 100 (R@100). To do this, we use ranks by R@100's of the Boolean queries generated for each topic as target values to be predicted. The formal definition of this model is given as follows.

Suppose that $Y = \{r_1, r_2, \ldots, r_l\}$ is a set of ranks, where l denotes the number of ranks, and we can order the ranks $r_1 \succ r_2 \succ \cdots \succ r_l$ where $\succ$ indicates the preference between two ranks. For training, a set of topics $\boldsymbol{T} = \{T_1, T_2, \ldots, T_m\}$ is given and each topic T_i is associated with $\boldsymbol{BQ_i} = \{BQ_{i1}, BQ_{i2}, \ldots, BQ_{in(T_i)}\}$ a set of Boolean queries, where $n(T_i)$ means the number of generated Boolean queries for T_i and a list of labels $\boldsymbol{y_i} = \{y_{i1}, y_{i2}, \ldots, y_{in(T_i)}\}$ each of which $y_{ij} \in Y$ indicates the rank of each Boolean query, BQ_{ij}, in T_i. A feature vector $x_{ij} = \Psi(T_i, BQ_{ij}) \in X$ is generated from each topic and Boolean query pair. We can represent a set of training examples as $S = \{(T_i, \boldsymbol{BQ_i}, \boldsymbol{y_i})\}_{i=1}^{m}$.

A ranking function $f: \mathrm{X} \mapsto \Re$ maps a feature vector associated with a Boolean query to a score for the query. Specifically, this model generates a permutation of integers spanned in $[1, \mathrm{n}(T_i)]$ for a topic, the corresponding Boolean query list, and the ranking function f. The permutation $\pi(T_i, \boldsymbol{BQ_i})$ is defined as a bijection from $\{1,2,\ldots,\mathrm{n}(T_i)\}$ to itself where BQ_{ij} is identified by an integer of $[1, \mathrm{n}(T_i)]$ and $\pi(j)$ denotes the position of BQ_{ij}. The model is learned to minimize a loss function which is defined by the disagreements between permutation $\pi(T_i, \boldsymbol{BQ_i})$ and rank list $\boldsymbol{y_i}$ for every training topic.

For learning, we use Ranking SVM [26]. In contrast to Boolean Query Generation where only pseudo-relevance is considered, we use real relevance judgments to compute R@100 of training examples for Boolean Query Ranking. This is because Boolean Query Ranking uses generalizable features while Boolean Query Generation uses terms which strongly depends on topics.

4.2.2 Features

In order to compose a feature vector for our query ranking model, we leverage features from previous studies for predicting query performance [21, 22, 23]. Previous studies (e.g., [24, 25]) proved that query quality predictors are effective for ranking sub-queries. Since generated Boolean queries also consist of subsets of terms in topic documents (e.g., query patents), we can expect those quality predictors also help to recognize effective Boolean queries. However, we additionally use more features specialized for our task because we observed that Boolean queries often show different characteristics from adhoc queries. Accordingly, we categorize our features into two groups, General Query Quality Predictor and Boolean Query Quality Predictor. Table 1 summarizes the features in each group.

General Query Quality Predictors contain features proposed by previous studies for quality prediction of adhoc queries. As shown in Table 1, these features include QCS, QS, SOQ, SCQ, IDF, and ICTF. Since Boolean queries show different aspects from adhoc queries for which those features have been proposed, we need to adjust the way these features are computed. For example, since adhoc queries do not contain negation (e.g., $\neg tartar$) in contrast to a Boolean query, we consider terms associated only with conjunctions. SOQ measures cosine similarity between a Boolean query and the baseline query while QS is computed only within pseudo-relevant documents, not within the whole collection because we aim to generate Boolean queries to retrieve pseudo-relevant documents. For IDF, ICTF, and SCQ, as [24] did, we calculate the sum, the standard deviation, the ratio of the maximum to the minimum, the maximum, the arithmetic mean, the geometric mean, the harmonic mean, and the coefficient of variation of each value of a query term. These modified rules are applied to both unigrams and bigrams.

Boolean Query Quality Predictors are features with the purpose of estimating Boolean query quality. All these features except BQTF are related to the retrieval results of a Boolean query because comparing a Boolean query result with the baseline result is a simple and effective way to predict Boolean query quality. BQCB is the ratio of the number of documents retrieved by both a Boolean query and the baseline system to the number of documents retrieved by the baseline system. This feature denotes how many of the documents retrieved by the baseline system can be found by a Boolean query. BQS is a measure of the number of pseudo-relevant documents retrieved by a Boolean query relative to the whole size of pseudo-relevant documents, i.e., *k*. This feature helps to assure the effectiveness of a Boolean query. LBQR measures the number of retrieved documents for a Boolean query. Since we found that an effective Boolean query sometimes returns a shorter result list containing highly relevant documents than the baseline result, we consider this feature as a signal to find such Boolean queries. BQTF counts the frequency of a conjunctive query term in pseudo-relevant documents, assuming that a frequent term in pseudo-relevant documents might be effective for retrieving the documents. Note that we do not consider negation terms because they rarely appear in pseudo-relevant documents. Besides, for BQTF, the same statistics as used for IDF are calculated.

Table 1: Two categories of Ranking Features.

General Query Quality Predictors	
QCS	Query Clarity Score [21]
QS	Query Scope [23] in pseudo-relevant patents
SOQ	Similarity to Original Query [24]
SCQ	Similarity Collection Query [22]
IDF	Inverse Document Frequency
ICTF	Inverse Collection Term Frequency [24]
Boolean Query Quality Predictor	
BQCB	Boolean Query result Coverage of Baseline retrieval results
BQS	Boolean Query Scope in pseudo-relevance
LBQR	Length of Boolean Query Result
BQTF	Boolean Query Term Frequency in pseudo-relevant documents

Overall, a feature vector contains 37 different feature values (from 10 different types).

5. EXPERIMENTS: Professional Search Simulation

We evaluate our Boolean query suggestion system by simulating professional search. We first describe how to set up experiments to simulate search processes and then provide experimental results and discussion with Boolean query examples.

5.1 Experimental Setup

To perform decision tree learning, the C4.5 algorithm[4] was used, with pruning turned on to obtain more accurate trees. For Boolean Query Ranking, SVM$^{\text{rank}}$ [5] is used as a learning-to-rank algorithm, and 10-fold cross-validation is performed. Queries and documents are stemmed by the Krovetz stemmer. More details are provided as follows.

5.1.1 Search Tasks

We setup experiments for two different search tasks considering two domains of interest: the patent and medical domains. Our task for the patent domain is patentability search which aims to find prior patents which may conflict with a new patent. On the other

[4] http://en.wikipedia.org/wiki/C4.5_algorithm

[5] http://www.cs.cornell.edu/People/tj/svm_light/svm_rank.html

hand, the search task for the medical domain is reference retrieval upon the request of disease information that the patient is undertaking. Both tasks are known to be recall-oriented.

5.1.2 Assumptions for Simulation

We suggest Boolean queries to help professionals formulate queries. In order to demonstrate the practical effectiveness of our system, we test our system under the following assumptions. First, we assume that professionals directly use suggested Boolean queries without reformulation because we want to show the lower bound of performance that our system can achieve. In real environments, professionals may use our suggestions or formulate new Boolean queries based on the suggestions. Second, we assume that the searchers will examine a maximum of the top 100 of every Boolean query result since 100 patents on average are examined in real examination processes as reported in [1].

5.1.3 Collections

Experiments are conducted on two different collections, patents and medical documents. As a patent corpus, we use USPTO (United States Patent and Trademark Office) patents provided by NTCIR-6 [8]. The collection contains 981,948 patents published from 1993 to 2000. To develop topics (new patents), we randomly selected 100 patents published in 2000, ensuring that their citations list more than 20 patents and at least 90% of them are included in the test collection. As done in the TREC chemical track [10] and NTCIR-6 [8], we considered patents cited in each topic patent as "relevant". We call this collection USPAT. For a medical corpus, we used the OHSUMED collection [27] which consists of 348,566 medical references (documents) and 106 queries (topics). This test collection contains relevance judgments manually annotated using three relevance levels (*definitely relevant*, *possibly relevant*, and *not relevant*). We consider *definitely* and *possibly relevant* as "relevant."

5.1.4 Baseline system and statistical Boolean retrieval model

For collecting pseudo-relevant documents as well as comparing performance, we employ baseline systems. As a baseline system for the USPAT, we consider the prior-art query generation method proposed in [6]. We generate a prior-art query which includes 100 unigrams ranked by TF-IDF from the "brief summary" section of each topic patent. To formulate the queries, we used both unigrams and bigrams, with 100 unigrams showed the best performance in R@100. Each query term is weighted by its TF in the topic patent. In addition, as [6] used, we employ the Indri search engine [28] to run each baseline query. For example, a baseline query is formed as "*#weight* (15 *glyph* 20 *character* …)". We consider the top 100 retrieved patents as pseudo-relevant. For OHSUMED, we consider all terms from the "Patient Information" and "Information Request" sections of each topic as a baseline query. The top 100 documents returned by the Indri search engine are regarded as pseudo-relevant.

To run Boolean queries, we use a statistical Boolean retrieval model. For each topic, we first find all documents satisfying the given Boolean function (i.e., Boolean query) and rank the documents by the generative probability of the query:

$$P(BQ|D) \approx \prod_{q \in BQ} P(q|D) \approx \prod_{q \in BQ} \frac{tf_{q,D} + \mu \cdot P(q|C)}{|D| + \mu}$$

where D is a target document satisfying a Boolean query BQ, q is the query term not associated with negation in BQ, $tf_{q,D}$ is the term frequency of q in D, $P(q|C)$ is the probability of q in the collection C, and μ is the Dirichlet smoothing parameter [29] set to 2000. Note that we do not use any query processing including query term weighting in this Boolean retrieval model. Since many current patent search systems (e.g., Patent Scope) are also based on these simple term statistics, query evaluation using this statistical Boolean retrieval model would be more practical and similar to real search environments than using other enhanced retrieval techniques (e.g., learning-to-rank or a dependency model) that are hard to integrate into current patent search systems.

5.1.5 Evaluation Measure

We use several metrics for evaluation. Since professional search is different from adhoc retrieval, and we are evaluating Boolean Query Suggestion, we employ new metrics in addition to conventional IR evaluation metrics.

Failure Rate measures the percentile ratio of "failure" Boolean queries to all generated ones for each topic. Boolean queries which failed to retrieve any target documents are considered as a "failure".

Success Rate measures the percentile ratio of "effective" Boolean queries to the all generated ones, where "effective" means a Boolean query performing identical to or better than the baseline query with regard to R@100. This metric denotes how many Boolean queries achieve the baseline performance.

Recall is a traditional IR evaluation metric; we use R@100 because every retrieval result is truncated at rank 100 according to the assumption for simulation. In order to evaluate our query ranking model, we use the best recall scores of the top-*n* Boolean queries for each topic because if the ranking model can place effective queries within top-*n* suggestions, they will be available to searchers.

F1 and F2 indicate weighted F-scores, both harmonic means of precision and recall, where F2 denotes double the weight on recall than F1. In particular, we compute F1@100 and F2@100 of the best performing query (with respect to R@100) from the top-*n* ranked queries. These metrics reflect the practical effectiveness of Boolean queries. Sometimes, the best Boolean query returns a short result list where many documents are relevant, which results in higher precision than the baseline results and expedites examination processes by professionals. F1 and F2 capture both recall and precision simultaneously, and help to measure search efficiency.

5.2 Results

5.2.1 Generation Performance

The first experiment is conducted to verify the effectiveness of Boolean Query Generation. We use the results of our baseline system as specified above and documents retrieved at ranks higher than 100 as non-relevant. In the USPAT, we generate 4 types of attribute sets; unigrams and unigrams+bigrams from a topic patent, and unigrams and unigrams+bigrams from the pseudo-relevant documents. For OHSUMED, only 2 types, unigrams and unigrams+bigrams from the pseudo-relevant set, are used because there is no topic document. Table 2 shows the performance of Boolean query generation. We report the average of each evaluation metric over all topics.

Table 2: Boolean Query Generation Performance. 'unigram' and 'bigram' indicate the Boolean queries generated from decision trees whose attributes are unigrams and bigrams, respectively. 'topic-doc' and 'prel-doc' denote that attributes are extracted from topic documents and a set of pseudo-relevant documents, respectively.

Metric	USPAT				OHSUMED	
	unigram (topic-doc)	uni+bigram (topic-doc)	unigram (prel-doc)	uni+bigram (prel-doc)	unigram (prel-doc)	uni+bigram (prel-doc)
Avg. # of Gen.	188.97	205.02	175.86	218.16	206.53	238.43
Avg. # of Fail.	17.25	15.06	11.97	13.28	16.37	18.22
Avg. Fail. Rate	9.47%	7.64%	6.98%	6.19%	7.93%	7.64%
Avg. # of Scc.	14.13	9.62	14.32	10.86	14.58	16.81
Avg. Scc. Rate	7.26%	4.56%	7.84%	4.63%	6.18%	5.85%

Table 3: Boolean Query Ranking Performance. A † indicates a significant difference from the baseline and a * denotes a significant difference of unigram results from unigram+bigram ('uni+bi') results in each row (the paired *t*-test is performed with $p < 0.05$). Significantly improved results on the baseline in each column are marked in bold. 'cut-off' indicates that all Boolean queries ranked within the cut-off ranks are examined. 'New Rel' and 'Missed Old Rel' denote the percentile ratio of new relevant documents (not retrieved by the baseline but discovered by the suggested Boolean query) and missed old relevant documents (which were retrieved by the baseline but missed by the suggested Boolean query) to all relevant documents, respectively.

	Metric	Recall@100		F1@100		F2@100		Missed Old Rel		New Rel	
USPTO	Baseline	0.2557		0.1184		0.1711		0.0%		0.0%	
	cut-off	unigram	uni+bi	unigram	uni+bi	unigram	uni+bi	unigram	uni+bi	unigram	uni+bi
	1	0.2227†	0.2174†	0.1062†	0.0969†	0.1515†	0.1421†	8.15%	9.60%	4.85%	5.21%
	2	0.2538	0.2445	0.1204	0.1096	0.1721	0.1604	6.49%	8.16%	6.30%	6.88%
	3	0.2670	0.2529	0.1264	0.1166	0.1808	0.1682	6.34%	7.65%	7.48%	7.36%
	4	0.2761	0.2535	0.1303	0.1169	0.1866*	0.1686	5.89%	7.65%	7.94%	7.42%
	5	0.2820	0.2592	**0.1330**$^{\dagger *}$	0.1191	**0.1905**$^{\dagger *}$	0.1721	5.58%	7.26%	8.21%	7.69%
	6	0.2852	0.2597	**0.1345**$^{\dagger *}$	0.1194	**0.1927**$^{\dagger *}$	0.1724	5.60%	7.20%	8.56%	7.69%
	7	**0.2883**$^{\dagger *}$	0.2622	**0.1359**$^{\dagger *}$	0.1209	**0.1947**$^{\dagger *}$	0.1745	5.53%	7.06%	8.79%	7.84%
	8	**0.2911**$^{\dagger *}$	0.2695	**0.1370**$^{\dagger *}$	0.1257	**0.1965**$^{\dagger *}$	0.1808	5.58%	6.98%	9.13%	8.46%
	9	**0.2952**$^{\dagger *}$	0.2710	**0.1388**†	0.1265	**0.1992**$^{\dagger *}$	0.1818	5.42%	6.88%	9.32%	8.66%
	10	**0.2991**$^{\dagger *}$	0.2722	**0.1402**†	0.1277	**0.2014**$^{\dagger *}$	0.1833	5.36%	6.83%	9.76%	8.91%
OHSUMED	Baseline	0.4377		0.2636		0.3462		0.0%		0.0%	
	cut-off	unigram	uni+bi	unigram	uni+bi	unigram	uni+bi	unigram	uni+bi	unigram	uni+bi
	1	0.3068†	0.3052†	0.2155†	0.2222†	0.2623†	0.2710†	15.02%	14.07%	6.20%	6.28%
	2	0.3618†	0.3611†	0.2490	0.2580	0.3063	0.3148	13.19%	13.09%	7.95%	7.88%
	3	0.3865†	0.3754†	0.2669	0.2774	0.3277	0.3382	12.48%	12.45%	7.89%	7.99%
	4	0.3970	0.3923†	0.2763	0.2874	0.3380	0.3493	11.17%	11.17%	8.18%	8.18%
	5	0.4009	0.4032	0.2836	0.2944	0.3440	0.3523	10.20%	10.56%	8.58%	8.58%
	6	0.4137	0.4082	0.2959	**0.3042**†	0.3569	0.3663	10.16%	10.11%	8.71%	8.58%
	7	0.4141	0.4106	**0.2961**†	**0.3045**†	0.3572	0.3676	10.11%	10.09%	8.76%	8.72%
	8	0.4143	0.4170	**0.2963**†	**0.3046**†	0.3573	0.3769	10.11%	10.04%	8.84%	8.84%
	9	0.4393	0.4232	**0.3076**†	**0.3169**†	0.3751	**0.3831**†	10.04%	9.63%	8.92%	9.84%
	10	0.4411	0.4232	**0.3089**†	**0.3185**†	0.3766	**0.3841**†	9.24%	9.41%	9.87%	9.84%

Table 4: Examples of the top-5 suggestions for Topic Patent titled as "compressor driving apparatus" 100+ indicates that the # of Retrieved documents is greater than 100. Each result is truncated at top-100. R is Recall@100 and F1 is F1@100.

n	unigram	R	F1	#Ret	unigram+bigram	R	F1	#Ret
1	inverter ⋀ compressor	0.35	0.1167	100+	¬ inverter driving ⋀ inverter ⋀ compressor ⋀ circuit	0.20	0.0748	87
2	inverter ⋀ compressor ⋀ circuit	0.55	0.1833	100+	inverter ⋀ air conditioner ⋀ circuit	0.50	0.1667	100+
3	inverter ⋀ motor	0.05	0.0167	100+	¬ power unit ⋀ air conditioner ⋀ output ⋀ inverter ⋀ circuit	0.55	0.2500	68
4	¬ inrush ⋀ ¬ metallic ⋀ inverter ⋀ compressor ⋀ relay	0.10	0.0435	72	¬ relay driver ⋀¬ compressor driving ⋀ inverter ⋀ circuit	0.05	0.0294	48
5	¬inrush ⋀ ¬metallic ⋀ ¬board ⋀ circuit ⋀ compressor ⋀ supply ⋀ inverter	0.25	0.1282	58	switching elements ⋀ air conditioner	0.30	0.1000	100+

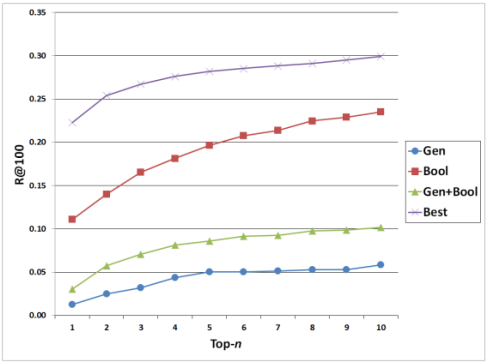

Figure 4: Recall of the best query within Top-*n* Boolean queries (unigram case in USPAT). 'Gen' and 'Bool' mean general query quality predictors and Boolean query quality predictors, respectively. 'Gen+Bool' uses all features, and 'Best' uses only selective features (QCS, SCQ, BQCB, and LBQR) achieving the best performance.

As shown in Table 2, our decision tree-based generation algorithm generates sufficiently many distinct Boolean queries. About 200 queries are generated for each topic, of which 6 ~ 9% fail to retrieve any target documents. In USPAT, pseudo-relevant documents are more reliable resources to generate Boolean queries than topic patents because of the smaller failure rate on average. Also, adding bigrams can lead decision trees to generate more queries, and the relative failure rate could drop. However, bigrams seems to be harmful in terms of the success rate. In addition, considering the number of "effective" Boolean queries (the number of successes), about 7% of queries show better or equal performance to the baseline system. Although this percentage may look low, we intend to obtain many effective queries via this generation process. Indeed, as you see from the number of successes, more than 10 effective queries are generated for each topic. If we can place these effective queries at top ranks using our Boolean query ranking method, professionals who examine these suggestions will find the effective queries. We address the performance of the query ranking technique in the following section.

5.2.2 Retrieval Performance

In the next series of experiments, we evaluate the effectiveness of Boolean Query Ranking by investigating if it succeeds in placing effective Boolean queries at high ranks, i.e., top 1 to 10. In training, generated queries for each topic are ordered by their R@100 scores. Also, in order to generate ground-truth ranked lists for training, we used Boolean queries from the topic-patent and pseudo-relevant patents in the USPTO, i.e., a ranked list contains unigram Boolean queries from the topic patent and pseudo-relevant patens and the other list includes unigram+bigram Boolean queries from the topic patent and pseudo-relevant patens. For the OHSUMED, we contain two ranked lists: unigram and unigram+bigram Boolean queries from pseudo-relevant documents. Creating a long ranked list by unifying unigram and unigram+bigram results is possible, but our scheme (handling unigram and unigram+bigram Boolean queries separately) empirically showed better performance. Overall, the training sets for the USPAT include 30,561 unigram and 38,034 unigram+bigram examples, and for OHSUMED contain 19,967 unigram and 23,122 unigram+bigram instances.

Prior to evaluating the performance of Boolean Query Ranking, we optimize the Ranking SVM by selecting optimal features. In order to obtain an optimal set, we tested several combinations of the proposed features (Section 4.2.2) using 10-fold cross-validation. To identify a better combination, we use the best recall scores of the top-1 to 10 ranked Boolean queries for each combination. We examine what features can help to produce better rankings. Figure 4 depicts the average recall of the best Boolean queries over all topics of unigram training set in the USPAT. **Gen** shows consistently the worst performance, and when combined with **Bool**, the performance of **Bool** is also degraded. The best combination (QCS, SCQ, BQCB, and LBQR) found via trials of several possible combinations that can outperform **Bool** only. This is consistent with what [25] found, i.e., SCQ and QCS are the most effective features for sub-query ranking. We additionally recognized that combining them with BQCB and LBQR is more effective because BQCB ensures that the Boolean query performs at least as well as the baseline and LBQR benefits effective Boolean queries returning a short result list. Experiments using the OHSUMED collection or unigram+bigram query terms showed similar tendencies.

With the optimal feature set, we now verify the effectiveness of our Boolean Query Ranking model by comparing with the baseline system. We calculate R@100, F1@100, and F2@100 of the best-performing query among the top-1 to 10 ranked queries.

Table 3 shows the retrieval results within the top-1 to 10 ranked Boolean queries by 10-fold cross validation. From the table, we can identify how many top-*n* Boolean queries need to be examined to find an "effective" one (i.e., performing as well as the baseline). In other words, results of the top-n queries which are not significantly different from the baseline result show that at least one effective Boolean query can be within the top-*n*.

In Table 3, we see that effective Boolean queries can be found within the top 2 or 4 suggestions in each corpus. In USPAT, an effective Boolean query is observed within the top 2 ranks in both unigram and unigram+bigram cases. Furthermore, in the unigram case, significantly improved results in terms of R@100 can be obtained by examining 7 or more Boolean queries. This is surprising to us because we expected Boolean queries to perform similar to the baseline. However, the result is a good indication that our system provides effective suggestions. In terms of F1 and F2, the top-5 unigram queries contain queries that outperform the baseline. These suggested queries retrieve about the same number of relevant documents as the baseline result, but with higher precision. That is, these Boolean queries may be more efficient in that they can allow professionals to examine fewer documents. On the other hand, effective queries are not successfully generated in case of unigram+bigram. For example, the number of generated effective queries in the unigram+bigram case is smaller than in the unigram case as seen in Table 2. Furthermore, many unigram results show statistically significant improvements over the unigram+bigram results, when comparing query performance at the same top-*n*.

In OHSUMED, more queries need to be examined to find effective Boolean queries compared to USPAT. For example, four query suggestions should be examined in the unigram case in the OHSUMED, while only two queries are needed in the USPAT. Furthermore, even more queries should be examined in the unigram+bigram case. In addition, we could not obtain significantly better Boolean queries with regard to R@100 in this domain. For F-scores, however, we also identify more efficient

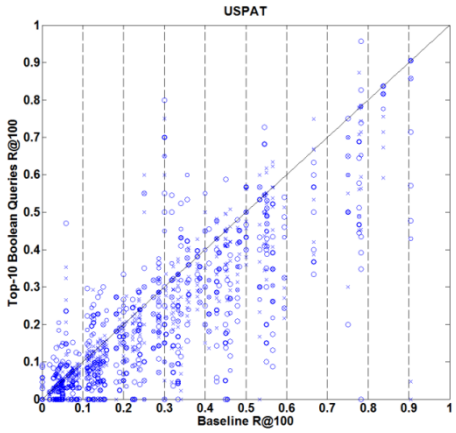

Figure 5: Scatter plot of the top-10 Boolean queries in USPAT. The x-axis represents R@100 of the baseline queries. A circle and a cross mean a unigram query and a unigram+bigram query, respectively.

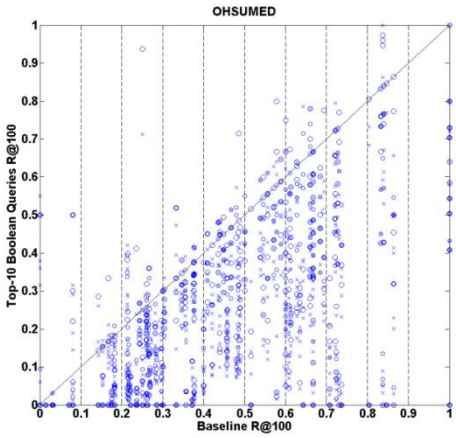

Figure 6: Scatter plot of the top-10 Boolean queries in OHSUMED. The x-axis represents R@100 of the baseline queries. A circle and a cross mean a unigram query and a unigram+bigram query, respectively.

Boolean queries by examining the top 6 or 7 queries. A critical difference between OHSUMED and USPTO is that there is little distinction between unigram and unigram+bigram results in the OHSUMED while unigram queries are consistently better than unigram+bigram queries in the USPTO. Overall, our ranking model is effective in placing "effective" Boolean query suggestions within the top 2 to 5 ranks.

In order to see how many new relevant documents (which were not retrieved by the baseline) are discovered and how many old relevant documents (which were retrieved by the baseline) are missed, by the best Boolean query within the top-*n*, we measure the ratios of the numbers of the new and missed old relevant documents to the number of all relevant documents (see "Missed Old Rel" and "New Rel" columns in Table 3). Generally, the more effective Boolean queries identified, the more new relevant documents are discovered while the less old relevant documents are missed. Specifically, for the best query in the top-10 suggestions, the number of newly discovered relevant documents is greater than that of missed old relevant documents. In addition, the difference between unigram and unigram+bigram queries is not significant.

Figures 5 and 6 depict scatter plots for the top-10 unigram and unigram+bigram Boolean queries in USPAT and OHSUMED, respectively. In Figure 5, the points on or above the diagonal line indicate effective Boolean queries (performing identical to or better than the baseline). The section from 0.0 to 0.1 in the x-axis contains many queries performing poorly. However, effective Boolean queries are also identified and some of them are superior to the low baseline. Also, there are many effective Boolean queries over the baseline performing moderately (0.1 ~ 0.7). In Figure 6, there are fewer poorly-performing queries (0.0 ~ 0.1), and most points are distributed between 0.1 and 0.9 in the x-axis. Along the diagonal line, effective queries in Figure 6 are more spread out than those of Figure 5. Although many Boolean queries underperform the baselines in both plots, our system is promising because effective queries can be identified by searchers in real environments.

Figure 7 depicts the accumulated length of the top-*n* query results which means the number of unique documents in all the results by the top-*n* queries. This is related to the efficiency of our system in

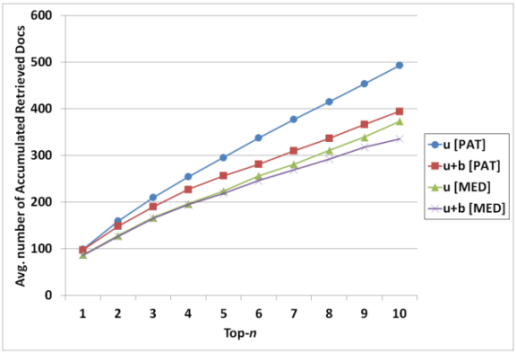

Figure 7: Average number of accumulated unique documents over the top-*n* Boolean queries. 'u' and 'b' indicate the results of unigram training examples and of bigram training examples, respectively. 'PAT' and 'MED' denote the USPAT and the OHSUMED collections, respectively.

the investigation, i.e., how long it takes to complete the task. Patent searchers generally examine maximally 600 patents to accomplish a single task (i.e., topic) [1]. In Figure 7, less than 500 unique documents accrue in the top-10 suggestions, which means that our suggestions can return a practical number of documents. Besides, since unigram+bigram queries return fewer documents than unigram queries, the coverage by unigram+bigram queries appears narrower.

5.2.3 Qualitative Analysis

We now provide a qualitative analysis of our system via real examples. Table 4 shows the top 5 Boolean queries suggested by our system, for a sample topic in USPAT. In this topic, the baseline system shows moderate performance (0.30 for R@100, 0.10 for F1@100), and some suggestions outperform the baseline. Many Boolean queries retrieve less than 100 documents, and some long suggestions (e.g., ¬power unit ∧ air conditioner ∧ output ∧ inverter ∧ circuit) can precisely retrieve relevant documents in the short result lists. Several suggestions return significantly more relevant documents. The suggested Boolean queries can provide reasonable query contexts. For example, "compressor" is often combined with "inverter", "supply", "circuit" in Table 4 because compressor driving apparatus can

include power supply, inverter drivers and storage circuits. Also, looking at the negated terms, professional searchers can recognize where negation is applied in the provided context. For example, "power unit" is negated when it comes with "air conditioner", "output", "inverter", and "circuit" in Table 4. Since we found that past cited patents are dealing with inverters or circuits for air conditioners, power supplies can be considered less important.

6. CONCLUSION

We proposed a framework to automatically suggest Boolean queries to assist professional searchers. We defined professional search as recall-oriented interactive tasks performed by information professionals. In order to provide reasonable suggestions in this context, we first generate a sufficient number of Boolean queries by exploiting decision tree learning and pseudo-relevant documents. To provide a reasonable number of suggestions, we rank the generated queries by a query ranking model using query quality predictors. In experiments, we simulated professional search processes in the patent and medical domains with our suggestion system. We found that our system can not only generate many effective Boolean queries but also select highly effective queries for suggestion. For future work, we plan to conduct experiments on the legal domain (e.g., finding relevant cases). Also, we will consider adding synonym structure into the current suggestion framework.

7. ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF CLUE IIS-0844226. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

8. REFERENCES

[1] H. Joho, L. Azzopardi, and W. Vanderbauwhede. A Survey of Patent Users: an analysis of tasks, behavior, search functionality and system requirement In *3rd Symposium on Information Interaction in Context*, 2010.

[2] L. Azzopardi, W. Vanderbauwhede, and H. Joho. Search System Requirements of Patent Analysts. In *SIGIR '10*, 2010.

[3] Y. Tseng and Y. Wu. A study of search tactics for patentability search: a case study on patent engineers. In *1st workshop on Patent Information Retrieval*, 2008.

[4] R. Bache and L. Azzopardi. Improving access to large patent corpora. *Transactions on Large-Scale Data and Knowledge-Centered Systems II* (LNCS), 2010.

[5] H. Turtle. Natural language vs. Boolean query evaluation: a comparison of retrieval performance. In *SIGIR '94, 1994*.

[6] X. Xue and W. B. Croft. Transforming patents into prior-art queries. In *SIGIR '09*, 2009.

[7] X. Xue and W. B. Croft. Automatic query generation for patent search. In *CIKM '09*, 2009.

[8] A. Fujii, M. Iwayama, and N. Kando. Overview of the patent retrieval task at the NTCIR-6 workshop. In *NTCIR-6*, 2007.

[9] H. Mase and M. Iwayama. NTCIR-6 Patent Retrieval Experiments at Hitach. In *NTCIR-6*, 2007.

[10] M. Lupu, F. Piroi, X. Huang, J. Zhu, and J. Tait. Overview of the TREC 2009 Chemical IR Track. In *TREC-18*, 2009.

[11] J. Gobeill, D. Teodor, E. Patsche, and P. Ruch. Report on the TREC 2009 Experiments: Chemical IR Track. In *TREC-18*, 2009.

[12] J. Rocchio. Relevance feedback in information retrieval. In *The Smart Retrieval System – Experiments in Automatic Document Processing*. Prentice-Hall, 1971.

[13] M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *SIGIR '98*, 1998.

[14] J. Xu. Improving the effectiveness of information retrieval with local context analysis. In *ACM Transactions on Information Systems*, 18(1), 2000.

[15] R. White, M. Bilenko, and S. Cucerzan. Studying the use of popular destinations to enhance web search interaction. In *SIGIR '07*, 2007.

[16] R. A. Baeza-Yates, C. A. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. In *EDBT workshops*, 2004.

[17] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW '06*, 2006.

[18] X. Wang and C. Zhai. Mining term association patterns from search logs for effective query reformulation. In *CIKM '08, 2008*.

[19] Z. Hashmi, T. Zrimec, and A. Hopkins. Automatic query generation from computerized clinical guidelines. *International Journal of Information Studies*, 1(4), 2009

[20] S. Russell and P. Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall, 2010.

[21] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting Query Performance. In *SIGIR '02*, 2002.

[22] Y. Zhao, F. Scholer, and Y. Tsegay. Effective Pre-retrieval Query Performance Prediction Using Similarity and Variability Evidence. In *ECIR '08*, 2008.

[23] B. He and I. Ounis. Inferring query performance using pre-retrieval predictors. In *18th Symposium on String Processing and Information Retrieval*, 2004.

[24] G. Kumaran and V. Carvalho. Reducing Long Queries Using Query Quality Predictors. In *SIGIR '09*, 2009.

[25] V. Dang, M. Bendersky and W. B. Croft. Learning to Rank Query Reformulations. In *SIGIR '10*, 2010.

[26] T. Joachims, Optimizing Search Engines Using Clickthrough Data In *KDD'02*, 2002.

[27] W. R. Hersh, C. Buckley, T. J. Leone, and D. H. Hickam. OHSUMED: an interactive retrieval evaluation and new large test collection for research. In *SIGIR '94*, 1994.

[28] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. Indri: a language-model based search engine for complex queries (extended version). *CIIR Technical Report*, 2005.

[29] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to adhoc information retrieval. In *SIGIR '01*, 2001.