Exploiting Keyword Co-occurrence and Citations for Query Generation

Badhrinath Sampathkumar
Southern Methodist University
Dallas, USA 75206
bsampath@smu.edu

Dr. Margaret Dunham
Southern Methodist University
Dallas, USA 75206
mhd@engr.smu.edu

Pete Fenner
Lightbus Technologies
Richardson, USA
4eaglemagic@tx.rr.com

ABSTRACT
This paper proposes a method of generating queries using sample patent documents and identifying keywords that have a high probability of getting relevant results. The technique is to extract keywords from the sample patent document using a keyword co-occurrence method and then using the cited patents to add additional keywords. This query is then expanded using the patent's classification classes. The generated query is then submitted to the search engine and the experimental results are provided.

Keywords
Information retrieval, query generation, domain specific keywords, patent search

1. INTRODUCTION
For any search engine, the quality of results is influenced by the quality of the input query. Most often while looking for new information, a user follows an iterative process of modifying his query depending on the current results. In cases where the user is familiar with the available information, a good query results in good recall and precision of the search engine. In cases where the user's knowledge of the domain is limited and the number of stored documents is large, generating efficient queries is quite difficult [5]. Also, when the subject of the search is fairly complex, a query is usually continuously modified by the user to obtain good results. This is particularly evident in a patent search engine where the patents are complex documents often describing multiple concepts in hard to understand language using highly specific domain jargon. It may also be the case that those doing the search are not domain experts. It would thus be helpful if a tool were available that could suggest useful keywords which the user could add to his query.

Whenever search is motivated by existing available information, more often than not, a document will be available, that can be used as a source for generating a query. This is particularly the case when searching the United States Patent and Trademark Office (USPTO) database [8]. Searching the data often involves being provided a target patent (application) to which related prior patents are desired. Although the USPTO system provides search tools, these are Boolean keyword based, and thus require the non-experts to identify important keywords. The ability, instead, to simply use the patent as the input would not only facilitate easier searching, but would also insure more accurate complete results. The objective of this work is to investigate the applicability of doing just that: providing a patent text document as input to the USPTO search engine. We thus propose the use of a preprocessor as a front end to the existing USPTO search engine. This facilitates the use of the current up to date USPTO database while facilitating easier access to the data. Our front end thus extends on the capabilities of the simple Boolean query interface to the patents thus providing more flexibility and functionality. The advanced search interface provided for the USPTO database is not intuitive, with the user having to type in the query in a format specific to the search engine. A large number of query terms make this process tedious and frustrating for the user. Having a preprocessor frontend thus also provides an easier access to the data. The primary users of our system would mainly be either a patent examiner looking for patents similar to the one under examination or a potential patent applicant who is looking for existing patents related to his invention. The patents being searched for are referred to as prior art. In both these cases the users have a large sample text available and are looking for other documents similar to it.

This paper is organized as follows. Section 2 discusses some related work in this area. Section 3 gives the query generation mechanism and describes the algorithm used in our approach. Section 4 gives the implementation details of the system. Section 5 presents some preliminary measurements made with this system.

2. RELATED WORK
Related work exists in two areas: automatic query generation and USPTO database access. We briefly examine each of these areas.

2.1 Patent Search Tools
There have been many projects and products which have examine the use of software to facilitate access to patent database access. These tools can be divided into two classes: those that access their own cached copies of the patent databases, and those that provide a frontend to access the existing patent databases.

There are two major products that provide access to their own archive of the patents. GetthePatent.com [11] facilitates access of patents by text, field, query, and number. The goal is to provide the ability to download multiple pages of a patent to a users PC using their own archive of patent databases including but not limited to the USPTO database. The accessed archive is updated weekly. Perhaps the most well know patent access tool is provided by Google Patent Search [15]. Using a traditional Google GUI interface, Google Patent Search facilitates a straightforward easy to use access of its own USPTO database archive.

There have been several other products that function as frontends to existing patent databases. IP-Discover [12] is a software tool which can be placed on a user's PC. As with GetThePatent, many different patent databases are searched. IP-Discover is not really a Web based tool as it searches the relevant patent databases and

downloads results in a background mode. Unlike GetthePatent, IP-Discover accesses the relevant patent databases as opposed to an archive. SurfIP also accesses many different patent databases [13]. It can also be used to search trademark and design databases. It provides Boolean keyword access, patent number search access, and access via formatted field values to the specific patent databases themselves. The GUI interface is easier to use than that of the actual patent database systems themselves. PatentHunter is a software package that provides a frontend to download complete patents to a user's PC [14]. It directly accesses the USPTO database. Free Patents Onine provides an online Web portal to access the USPTO database [15]. It uses a sophisticated Porter Stemming method to improve recall. Identified patents are downloaded in either a compressed pdf format or XML. Some aggregate operations of patent fields (such as forward and backward link counts) are provided as are data analysis tools.

In the above sampling of the existing patent search tools, none of the products facilitate an algorithm such as the one we propose. Also, none of them allow a patent (or text document) to be provided as input.

2.2 Automatic Query Generation

There have been many ways proposed to generate queries automatically. Primarily query generation is done in two ways,

1. The user provides a word that he/she is interested in and the system uses this word and establishes a set of relevant keywords related to this word

2. A system takes non intrusive samples of the user's environment to establish a probable context and suggests keywords that user may be interested in.

In the method proposed by Ryu et. al. [3], the keyword expansion approach is used. The keyword that the user gives is acted upon by an inference engine that uses an ontology knowledge base and generates a list of related keywords. A semantic thesaurus is then used to add keywords to these terms. Term weights are then calculated for these expanded keywords and top n relevant keywords are then used as the query that is submitted to a search engine. With a patent search engine, we do not have the facility to establish a context and keywords may have multiple meanings with respect to the context in which they occur and the number of different contexts may be large. This makes it difficult for a user to define an ontology which could be used by an inference engine to generate additional keywords.

The method proposed by Kulyukin for automatic query generation [4] is suitable to situations where retrieval of documents is not the primary task and the size of the document collection is known. The system takes random samples of the text on which the user is working and generates queries. The results of the queries are presented to the user and feedback is solicited in a non-intrusive manner. There are two drawbacks in using this approach with a patent search engine. One is for the user to validate the results and the other is the prior knowledge of the document collection. Given the vast number of documents in the patent database, it would be very difficult for the user to quickly make any form of relevance judgments. In our technique we do not know the document collection and avoid user feedback.

The methods listed above do not work with a dynamic patent database. In this environment we do not have a corpus to train the system.

Our contribution is two fold. First, we allow the user to specify the input text, which is arbitrarily large like the claims section from a prior patent or from a patent application. Secondly, the query is generated from a single document without any need for training.

3. PATENT SEARCH

Our work was primarily motivated by the difficulty involved in searching for patent documents. Given the large body of text present in a patent database, it is a difficult proposition when one has to search for patents that are similar to a given text. While the user would have to read through the entire text to validate the usefulness of the returned document, there are no first level tools that assist in identifying a set of potential prior arts. For example, when it comes to searching for patents, different types of searches may be possible [8].

Novelty Search: This is done to establish the patentability of an invention. This search would include any and all public domain knowledge to determine whether the invention is novel enough to be patented. The search realm for this type of search would include more than just the patent database.

Infringement Search: This search is conducted to find whether the invention would infringe an already existing and unexpired patent. This search primarily involves the patent databases. The claims section of the patent is ideally suited for this search.

State-of-the-Art Search: This search is used to establish the current state-of-the-art is the field related to the invention. This search usually yields the list of patents that form the prior art for this invention as long as the infringement problems are avoided.

Right-to-Use Search: This search is used to establish whether there is a basis for not enforcing an infringed unexpired patent. This basis is usually determined by examining the application process for the infringed patent and an examination of the prior art of the infringed patent.

Our proposed technique would be useful as a first level search tool in all the above search cases. In all these cases, forming queries to maximize relevant results can prove to be tedious. Having a tool that could suggest useful keywords and rank them based on importance would be highly effective for the users.

3.1 USPTO Patent Document

A USPTO Patent document is a legal document published by the USPTO describing the invention for which the patent has been granted. The patent document is made of the following sections,

- [PN] Patent number, which identifies the patent
- [IN] The name of the inventor
- [AN] The name of patent owner or the assignee
- [PEX] The examiners of the patent application
- [CCL], [ICL] US and international search classification of the subject matter
- [REF] [OREF] Prior art that are referenced by this patent. These may be other patents and/or other documents in the public domain

- [ABST] Abstract section describing the environment and nature of invention in less than 250 terms
- [SPEC] Description that describes the background of the invention along with the field of invention and proceeds to describe the details of the patent.
- [ACLM] The Claims section which lists the qualities of this invention in terms of what it does.

The complete list of patent document fields and their notations can be found in [11].

3.2 Search Engine

We use the patent search engine hosted at USPTO for running the queries generated by our applications. The search engine handles simple field based queries and has an advanced search section where detailed queries can be specified using the described query format. In this paper we use the generated query words to create the query in the advanced query format. The query format is given below:

FIELD/(*field item*) AND FIELD/(*field item*) AND ..

where,

FIELD may be one of; ACLM, ABST, SPEC, APD, PN

Field item is the value associated with that field

The fields ACLM, ABST, SPEC are used to search the body of the patents while the others are specific fields of the patent such the patent number, classification and dates.

ABST refers to the abstract of the patent. The abstract contains less than 250 words and is not a useful indication of the details of the invention. It is used as an overview section for users than as a potential search text. ACLM refers to the claims section of the patent and is the best description of the patent. We primarily search this section for obtaining good results. SPEC refers to the description section of the patent. While SPEC usually contains the largest amount of text, it is usually diluted compared to the claims section as it is more verbose compared to the claims section. In our studies we primarily used the ACLM and the SPEC based queries.

4. OUR APPROACH

The best source information about a document is contained within the document itself. This property is exploited by the tf-idf measure [17] which promotes words that occur frequently within a document but occur less frequently in a document collection. This metric is widely used in indexing schemes for search engines. If one can have the knowledge of the index of a search engine, then it would be easy to select the keywords for the query. But it is not feasible to as the indexes change continuously over time as new documents are added to the system. So, the next option is to get a set of keywords that closely resemble the index stored by the search engine for that document.

We use an algorithm originally proposed as an alternative to the tf-idf indexing scheme. In our initial studies we found that it works well as technique to generate keywords for queries as opposed to indexing.

4.1 Keyword Extraction

We now describe the algorithm proposed by Matsuo Et. al. [1]. Following this description we introduce our extended version that takes citations and classification in to account.

The main idea of Matsuo's algorithm is that, if a term occurs frequently with a set of other high frequency terms, then the term is likely to be significant. This co-occurrence bias is calculated using the χ^2 test. The test is used to calculate the deviation of expected frequencies from observed frequencies. Statistically, χ^2 is defined as,

$$\chi^2 = \sum_{g \in G} \frac{\left(freq(w,g) - n_w p_g\right)}{n_w p_g} \quad (1)$$

Where,

- *freq(w,g)* is the frequency of co-occurrence of terms w and g
- G is the set of frequently occurring terms in the document
- $n_w p_g$ is the expected frequency of co-occurrence with n_w being the number of terms in the sentences where w occurs and p_g is the probability that a term in the document may co-occur with g

Matsuo's algorithm uses a modified Chi Squared as shown below:.

$$\chi'^2(w) = \chi^2(w) - \max_{g \in G} \frac{\left(freq(w,g) - n_w p_g \right)}{n_w p_g} \quad (2)$$

In this equation the χ^2 value of the maximal term is subtracted. This is useful, since the χ^2 value would be low if it co-occurs with only one term and high if it co-occurs with multiple terms. To improve the reliability χ^2 values, the algorithm clusters frequent terms and calculates the χ^2 values over the clusters. This is primarily done to prevent large χ^2 values of terms that occur with co-occurring frequent terms. Clustering is done pair wise using Jensen-Shannon divergence [7] that calculates the similarity between two distributions. Two terms belong to the same cluster if their Jensen Shannon divergence is greater than the threshold value (0.9 x log 2). The clustering is also done using mutual information. This means that if terms w_1 and w_2 occur together frequently, then they are considered to be in the same cluster. The mutual information threshold was set as log 2. The threshold values were determined by running a few experiments with a variety of patent documents and observing the results. We started the thresholds at 1.0 and kept varying the threshold each time until the results had relatively lower number of irrelevant terms towards the bottom of the list.

4.2 Query Generation

We utilize the keyword co-occurrence algorithm to generate a basic set of keywords from a given patent document. Once the set of initial keywords are generated we add to this list in two ways. We use the patent classification system as a semantic database to add additional terms to the query. Next we utilize the citations list of the input patent and process those through the keyword co-occurrence algorithm. We then add to the initial list the unique list of keywords generated from using the citations and the classification database.

The algorithm steps are given in the next section. We retain the steps from the original keyword co-occurrence algorithm and add our modifications to it. We can either run the algorithm against the claims section or the specification sections of the input patent.

4.3 Algorithm Listing

Step 1: Preprocessing

- Discard stop words listed in [9].
- Stem the remaining words using the Porter Stemming Algorithm [12]

Step 2: Selection of frequent terms

- Select the top frequent terms up to 30% of the number of running terms (N_{total})

Step 3: Clustering frequent terms

- Cluster a pair of terms whose Jensen-Shannon divergence is above the threshold (0.90xlog 2).
- Cluster a pair of terms whose mutual information is above the threshold (log(2.0)).
- Terms can be clustered by either of the two clustering method. Let the resultant set of clusters be called C.

Step 4: Calculation of expected probability

- Count the number of terms co-occurring with c Є C, denoted as n_c, to yield the expected probability $p_c = n_c/Ntotal$.

Step 5: Calculation of χ^2 value

- For each term w, count co-occurrence frequency with c Є C, denoted as *freq(w, c)*. Count the total number of terms in the sentences including w, denoted as n_w. Calculate χ'^2 value given by (2).

Step 6: Output keywords

- Normalize the χ'^2 values using the highest value returned and show the list of words that are greater than a threshold value. We have found that a threshold value between 0.2 and 0.5 depending on the text size works well to return relevant terms. Mark this as `List1`.

Step 7: Extract terms from classification category

- Get the classification numbers of the input patent and extract the words in the classification category after discarding stop words. Mark this as `List2`.

Step 8: Run the keyword co-occurrence algorithm against the citations

- Run the keyword generation algorithm against the appropriate section of the citation patents and get a list of the unique terms. Mark this as `List3`.

Step 9: Merge the generated keywords

- Merge `List1`, `List2` and `List3`

Step 10: Form a USPTO query from the keywords generated

- Form a USPTO query and submit the resultant query to the Patent Search Engine.

Step 11: Calculate the actual relevance of the results

- Calculate the relevance of the resultant patents with the input patent to determine the actual relevance.

We illustrate the use of this algorithm with the following example. For purposes of being concise, we make use of small amounts of text here. Consider the abstract of a USPTO issued patent shown below,

USPTO No. 5095480,

Title: Message routing system for shared communication media networks

Input Text: A plurality of disparate communication network systems communicate with each other through the use of different physical media protocols. Each of the systems has at least one input and one output. A message routing system couples a transmitter at any one system input to a receiver at any other system output using a message format that is structure independent of the location of the receiver in the system. Each receiver/transmitter device coupled to any one system input has a unique, fixed and unchangeable identification code regardless of the communication network system to which it is connected. To couple a message from any one receiver/transmitter device to a second receiver/transmitter device at an unknown location within the communication network system, a message format is transmitted from the sending location containing the fixed, unique identification code of the receiving station. A routing system having a plurality of intermediate routing devices receives the message format and couples it to the receiving station at the unknown location using only the fixed, unique identification codes of the transmitting and receiving stations and the addresses of the intermediate routing devices for determining routing.

Identified Terms From Patent: receiver, systems, routing, device, message

Suggested Terms (From Classification): address, architecture, area, channel, channels, communicated, data, delay, distributed, expense, field, geographical, header, information, interexchange, nodes, packet, pattern, plurality, pulse, routing, selection, signalling, specified, switch, switching, toll

Suggested Terms (From Citations): lines, packets, connected, output, telephone, radio, sets, specific, zones, devices, searching, arrangement, paging, switches, local, data, channels, cellular, personal, computers, cellular, system, nodes, network

USPTO Query: ABST/receiver AND systems AND routing AND device AND message AND lines AND communication AND paging AND nodes AND data AND personal AND telephone AND switches AND information AND packet AND network AND computers

No. of Results: 1419

Results using only the input patent keywords: 21363

5. IMPLEMENTATION

This section describes our implementation of the algorithm. The algorithm is implemented as a python CGI application running on the apache web server. The application would connect to the USPTO database and fetch a patent document, which was then processed to remove the HTML tags and stored in a local database

to speedup subsequent fetches. The outline of the program flow is given in Figure 1.

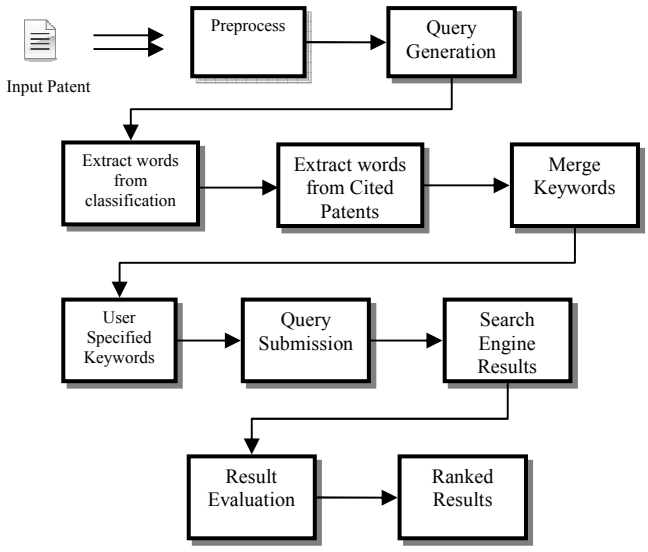

Figure 1: Implementation Flow Sequence

Input Text: For our search purposes we primarily use the claims section since it forms the best description of the patent. The abstract section has very little text to conduct any meaningful search and the description contains a lot of loose text that could dilute the generated query resulting slightly reduced precision.

Preprocess: The input text is preprocessed by removing the common English stop words. Then the set of patent domain stop words identified in [9] are removed. The remaining words are then stemmed using the porter algorithm. The python based NLTK implementation was used for most of the text processing.

Query generation: The algorithm described in the previous section is used to generate the query keywords. These keywords are then converted to the patent search engine format based on the user's selection of the field to be searched.

Extract Words from Classification: The classification numbers from the input patent is used to generate a set of additional words. Usually these words tend to be very closely related to the terms generated from the patent text.

Extract Words from Cited Patents: The citations from the patent are used to generate additional keywords. The purpose of doing this is to span the keywords across as many classification categories as possible. For example, some patents may reference an invention in a totally different field. Doing the citation analysis would greatly increase the coverage of the patent query that we generate.

Merge Keywords: We then merge the keywords together. The merging is currently done in a naïve manner. We take the top ranked words from each cited patent to formulate the query. We then add any unique words that we get from the classification categories.

User Specified Keywords: The user is allowed add keywords that he feels would be useful in improving the search results. The list of keywords that are generated is also made selectable by the user. The user could select the keywords randomly from the list or go with the top n terms.

Query Submission: The formulated query is now submitted to the USPTO search engine using python's urllib semantics [9].

Search Engine Results: The results that are returned by the USPTO search engine contain the patent number and its title. The patents identified by the numbers returned are also fetched.

Result Evaluation: The returned patens text is parsed and is compared with the input text of the generated query. We use a vector model and compare the two documents using the cosine method.

Ranked Results: The compared set of documents is then ranked according to their similarity with respect to the input text.

6. EVALUATION & RESULTS

To evaluate the performance of this system, we ran a series of experiments for various input patents. We used two metrics to evaluate the performance of our implementation.

- *Precision* which determines the number of relevant documents in the retrieved set *and*
- *Expected search length* which evaluates the system based on the position of the relevant document in the retrieved set.

We do not use recall as a measure because we don't know the relevant set. Also, it is not possible to fix the relevant set based on the citations and references. Rather than using recall as a measure, we use sliding ratio. This would give an indication of whether the system is useful. Also, we define a way of determining relevance from result set using the similarity measures that were returned.

In the experiments we generated the queries using the claims text, primarily because the claims section has the best information about the patent. This is also evident from table 1 which gives the average normalized χ^2 for terms extracted from claims and description sections.

Patent Number	Claims	Description
5095480	0.751247	0.5366318
7251748	0.5600668	0.487591
6071203	0.516233	0.4883837
5554588	0.517553	0.4763623

Table 1: Average χ^2 Values

The number of terms included in the query was limited by the query length for the USPTO search engine which is a 256 character limit on the expanded query. In cases when the query terms return too few results from the USPTO search engine, we used the search engine hosted at [19]. When we include a large number of terms related to the input patent to form a query, it results in returning only the input patent. This is as expected since a larger set of words would more accurately define a documents index. Since patent claims tend to be well defined for a particular invention, the search engine returns the patent the query was generated from.

To find out the performance of the search query, we evaluate the returned patents and compare the claims text with the input text

that generated the query using a cosine method[x]. Table x lists the similarity of the returned results to the input text which was the claims section of the patent. To evaluate the performance of the system we use the sliding ratio method as given in [x]. Table x gives the similarity scores for the results generated by using the claims text of patent number 7,251,748.

Patent Number	Similarity	Sliding Ratio
7,036,729	0.246259140656	0.777135782
6,785,779	0.269319317422	0.879526881
6,606,604	0.316880455599	0.993770032
6,330,347	0.200545157187	0.953011384
6,323,846	0.191944449865	0.928532033
6,138,123	0.226133508433	0.938989953
5,822,712	0.235294541252	0.965901222
5,410,732	0.251477845385	1

Table 2: Sliding Ratio & Similarity

From the table it can be seen that the third patent returned has the highest similarity to the input text. A disadvantage of this measure though is that the tendency to return false positives. To determine the precision of the system, we decided on a threshold of similarity of the returned results. From the results, we can see that the similarity values tend to be on the lower side. This is primarily because the claims section are carefully worded and since we are comparing issued patents, the scores would be lower. To fix the relevant documents without the user's intervention, we utilize an average value that improves our results. Since the entire documents are compared, the similarity scores give us a good sense of estimation of which documents are relevant to our search. We use the average of the similarity scores returned to fix a precision threshold. Similarity scores less than the average are automatically labeled non relevant. The precision values calculated using this method is given in table 3. We again used claims sections of the input patent.

Patent Number	Query Length	Precision
5095480	7	0.416
7251748	5	0.5
6071203	19	0.57

Table 3: Search Precision

Expected search length is the number irrelevant documents that need to be examined before the first n relevant document is found. Table 4 gives the expected search lengths for various lengths of n. It can be seen that the queries that were generated worked reasonably well for short lengths. For the patent 5095480 the ESL tended to be high since the document contains a higher percentage of generic terms related to the domain of the invention. This is very clear when we see that the patent has been referenced by more than 200 other patents.

Patent Number	ESL (n = 2)	ESL (n = 5)	ESL (n = 10)
5095480	2	8	9
7251748	1	1	7
6071203	0	0	7

Table 4: Expected Search Lengths

Our approach attempts to reduce the cost and time required for searching patents. A strategy that can be adopted while searching for patents is outlined in [10]. Using our approach we can reduce the time and cost overhead for some of the steps involved in this strategy. The primary reduction in time can be achieved during the classification process by automatic suggestion of potential keywords and identification of relevant classes from the keywords. Also the system can be used to query USPTO to retrieve patents by classification and rank the patents based on their relevance to the input text. This has a potential of greatly reducing the time taken to find prior art within the patent domain.

7. FUTURE WORK

In this paper we presented a non intrusive method for query generation. A patent search engine was used to demonstrate the concept. We used a patent document to generate queries. These queries were submitted to the search engine and the results returned were ranked using vector model by comparing agaist the text that generated the query. The results were evaluated using identified metrics. We would like look at a way of suggesting additional keywords using NLP techniques that would try to improve the results. NLP techniques can also be used to improve the Boolean queries by making synonyms among the query terms as disjunctions. Also, comparative testing between the system and human users would highlight the instances when the system falls short. While avoiding intrusive relevance feedback is a primary, we would look at a way of introducing this system into a Content Management System that could improve upon the keywords currently generated by making implicit measurements against queries.

8. REFERENCES

[1] Y. Matsuo and M. Ishizuka. "Keyword Extraction from a Single Document using Word Co-occurrence Statistical Information", *International Journal on Artificial Intelligence Tools,* Vol. 13, No. 1 (2004) 157-169

[2] George Almpanidis and Constantine Kotropoulos, "Combining Text and Link Analysis for Focused Crawling", *Information Systems*, Volume 32, Issue 6 (September 2007), Pages 886-908

[3] Hoyeon Ryu, et al., "n-Keyword based Automatic Query Generation", *ICHIT, Proceedings of the 2006 International Conference on Hybrid Information Technology*, Volume 02 (2006),Pages 90-96

[4] V. A. Kulyukin, "Automated query generation for embedded information retrieval." [Online]. Available: http://citeseer.ist.psu.edu/449311.html

[5] Burke, R., Hammond, K. & Cooper, E. "Knowledge-based navigation of complex information spaces". In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 462-468, AAAI, 1996

[6] L. S. Larkey, "A patent search and classification system," in *Proceedings of DL-99, 4th ACM Conference on Digital Libraries*, 1999, pp. 179-187.

[7] I. Dagan, F. C. N. Pereira, and L. Lee, "Similarity-based estimation of word cooccurrence probabilities," in *Meeting of the Association for Computational Linguistics*, 1994, pp. 272-278.

[8] Howard B. Rockman, “Intellectual Property Law for Scientists and Engineers”, *IEEE Press,* 2004

[9] USPTO, “List of Stopwords”, [Online]. http://www.uspto.gov/patft/help/stopword.htm

[10] USPTO, “The 7-Step U. S. Patent Search Strategy”, [Online], http://www.uspto.gov/go/ptdl/step7.htm

[11] USPTO, “Tips on Fielded Searching”, [Online] http://www.uspto.gov/patft/help/helpflds.htm

[12] M. F. Porter, "An algorithm for suffix stripping" in *Readings in information retrieval*, Morgan Kaufmann Multimedia Information And Systems Series, 1997, pp. 313-316

[13] Gerard Salton and Michael J. McGill, “Introduction to Modern Information Retrieval”, McGraw-Hill, 1983.

[14] Get the Patent, http://www.getthepatent.com/.

[15] IP-Discover, http://www.ipdiscover.com/ .

[16] SurfIP, http://www.surfip.com/ .

[17] PatentHunter, http://www.patenthunter.com/ .

[18] Google Patent Search, http://www.google.com/patents .

[19] FreePatentsOnline, http://www.freepatentsonline.com/ .