A Scalable Approach For Performing Proximal Search For Verbose Patent Search Queries

Sumit Bhatia
Computer Science and Engineering,
Pennsylvania State University
University Park, PA 16802
sumit@cse.psu.edu

Bin He, Qi He, Scott Spangler
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95050
{binhe,heq,spangles}@us.ibm.com

ABSTRACT

Even though queries received by traditional information retrieval systems are quite short, there are many application scenarios where long natural language queries are more effective. Further, incorporating term position information can help improve results of long queries. However, the techniques for incorporating term position information have been developed for terse queries and hence, can not be directly applied to long queries. Though there exist some methods for performing proximal search for long queries, they are not scalable due to long query response times. We describe an intuitive and simple, yet effective technique that *implicitly* incorporates term position information for long queries in a scalable manner. Our proposed approach achieves more than 700% faster query response times while maintaining the quality of retrieved results when compared with a state-of-the-art method for performing proximal search for very long queries.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – Retrieval Models, Search Process

General Terms

Algorithms, Experimentation

Keywords

Verbose queries, long queries, term proximity, proximal search, prior art search, patent search.

1. INTRODUCTION

Majority of information retrieval research focuses on *keyword* queries where the user represents her information need in form of a few keywords that representing the key underlying concept. Likewise, most of the queries received by commercial web search engines are very short – 2-3 keywords on an average. However, there are many important use cases where the user's information need can not be sufficiently described by a few keywords. Longer queries can better express complex and specific information needs [10] and are common in enterprise search systems [5] and search-in-context systems [1]. Further, in many scenarios natural language queries are preferred over keyword queries such as in community question answering services [14]. In prior art search or patent retrieval, it is common to use abstracts of patents as queries to search for similar/related prior patents [12].

Surprisingly, given the vast applications of long queries, modern search engines usually perform poorly with long queries [7]. Further, a majority of commercial web search engines do not provide support for very long, verbose queries. For example, Google has an upper limit of 32 keywords for a search query. One major reason for limiting the size of input queries to the search engine is that the time taken for search and retrieval operations increases with the number of words in the input query. Hence, most of the current techniques for handling verbose queries first transform the long query into a shorter query by either extracting key *concept* phrases from the query [2] or by removing less important terms from the query [7]. This query transformation step increases the time required to retrieve results for the query. Further, many of these approaches require some kind of training data to learn the importance of different terms to distinguish key terms from non-key terms. Such data may not always be available for all the applications and thus, preventing the use of such techniques.

In addition to long response times for verbose queries, the results produced by the present search systems are often not satisfactory and contain a very large number of false positives (i.e., very low precision) for longer queries [12]. The major reason for such a low precision is again the length of input queries. Any document that contains at least one of the query words is a potential match for the query and since the input query is very long and contains a large number of keywords, the number of matching potential documents is very large. Even though commonly used document ranking functions help alleviate this problem to some extent by assigning more weight to more important terms and assigning higher scores to documents containing multiple query terms, the results are still far from being acceptable. This is because the most commonly used ranking functions do not take into account the relative position of query terms and their occurrence in the document. Utilizing term proximity information can help improve retrieval performance in such scenarios [12, 13]. However, most of the current works on incorporating term proximity information in retrieval models have focused on short keyword queries [13, 3, 6] and are not suitable for use with longer queries. Moreover, these involve additional computation to determine the proximity information and coupled with the already long processing time for verbose queries, make the overall search process very slow. In this paper, we describe a technique for incorporating term proximity information for long queries in a scalable manner. Our approach does not involve a query transformation step and also does not require explicit term position computations at query time. We split the documents and input queries into simpler and smaller subunits so that the query length reduction step (for handling long queries) and term position computation (for proximity information) are implicit in the retrieval process. For experiments, we chose patent retrieval as the application domain given the very long input queries that are usually used for patent search (abstracts of patents). We compare our proposed approach with a state-of-the-art approach for performing proximal search for long queries and we achieve more than 700% faster query response times while maintaining, and in many cases, improving the quality of retrieved results.

2. PROPOSED APPROACH

The major purpose of proximal search operation in document retrieval is to ensure that documents containing query terms in close proximity to each other are assigned a higher score. This becomes more important for longer documents. The current methods for proximal search utilize the position information of various terms in the document to assign higher scores to documents that contain query terms in close proximity to each other. The position information for document terms can be pre-computed at index time and distances between query terms present in the document can be computed at run time. For longer queries, these computations can significantly increase the system's response time. Hence, in order to improve query response time, we propose a method that eliminates these distance computations. Figure 1 describes the intuition behind our approach. At index time, we segment the documents in the corpus into much smaller units called *snippets*. For the proposed method, a snippet consists of three consecutive sentences from the original document. Even though we generated snippets that were 3 sentences long, number of sentences in a snippet can be varied depending upon the application. Longer snippets will result in smaller number of overall "documents" to index but a weak proximal search operator whereas shorter snippets favor strong proximity operators at the expense of large number of documents in the index. As suggested by Spangler et al. [11], a snippet length of three sentences offers a fine balance between the two considerations. Thus, a given document is decomposed into a number of snippets that are much shorter in length and each snippet is now treated as a separate document and is indexed by the indexer of the search system. We assign identifiers to each snippet so that all the snippets generated from a given document can be identified. Note that by decomposing a document into snippets and indexing these individual snippets introduces an *implicit proximal search operator*. On receiving a user query, the search system finds snippets containing one or more query terms. These snippets are then ranked using a ranking function that usually assigns higher scores to

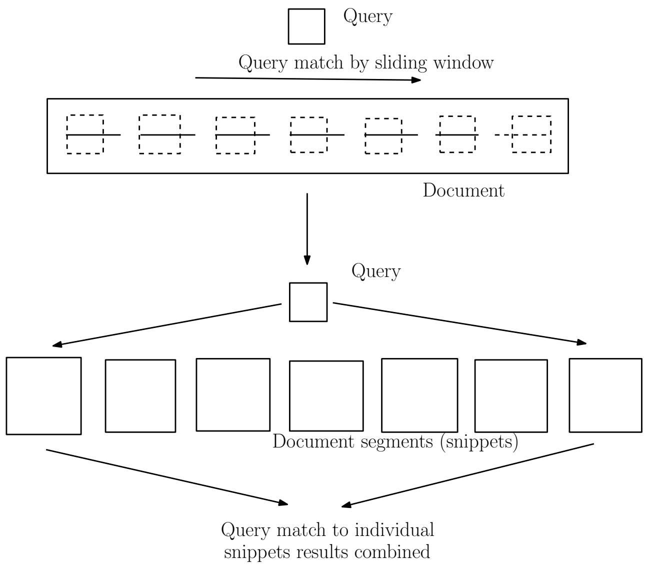
Query
Query match by sliding window
Document
Query
Document segments (snippets)
Query match to individual
snippets results combined

Figure 1: Intuition behind proposed approach for implicitly performing proximal search for long queries. For proximal search, query is matched to different regions of the document in a *sliding window* fashion. We simulate the same effect by splitting the document into shorter sub-documents or *snippets*.

snippets that contain multiple query terms. In this we use the standard query likelihood language model as our ranking function. Thus snippets that contain many query terms are favored over snippets that contain only a few query terms. Note that since each snippet is actually a small continuous segment of the original document text, a snippet containing multiple query term implies that in the original document from which the snippet was generated query terms are present in close proximity to each other. Thus, we have obtained the term proximity information without having to compute the distances between query terms present in the document at query time.

The method described above produces a ranked list of snippets as output. However, the users are interested in obtaining relevant "full documents" rather than snippets. Hence, in order to obtain a ranked list of documents, for each document we add the scores of all the snippets generated by a given document and assign the resulting score to the corresponding document. Thus, documents having multiple matching snippets for a given query are assigned a higher score. We also note that the snippet scores can be merged into document scores in various different ways also depending upon the requirement of the application. The above described method can be extended to accommodate very long queries by decomposing a long query into smaller subqueries. For example, if the input query is a paragraph of text, the paragraph can be decomposed into its constituent sentences and each sentence can then be used as a query. The results of all the sub-queries can then be merged together to create a final ranked list for the paragraph query. The steps involved in our method are summarized below:

1. At index time, split all the documents into smaller sub-units (snippets) and index all the snippets.

2. Split the input long query into its constituent sentences.

3. Split the input query into its constituent sentences[1]. Use each query sentence as an individual query and and select the top K (we use $N = 500$) snippets based on a relevance function (we use standard query likelihood language model).

4. As described above, transform the snippet results into document results for each sentence sub-query by adding scores of all the snippets belonging to the document.

5. Sum scores for each document for each sentence and sort by the final scores. The top N documents can then be shown to the user in decreasing order of their scores.

3. EXPERIMENTAL PROTOCOL

3.1 Data Description

3.1.1 *Corpus*

We use the publicly available *Marec 400.000* patent collection that was used in AsPire'10 workshop[2] collocated with ECIR 2010. The dataset consists of 400,000 randomly selected patent documents, with 100,000 patents belonging to EPO (European Patent Office), USPTO (US Patent Office), JPO (Japanese Patent Office) and WIPO (World Intellectual Property Organization), respectively. For our experiments, we only selected patents belonging to the USPTO set as the patents belonging to other patent offices were written in languages other than English. Thus the final dataset consists of 100,000 patent documents.

3.1.2 *Queries and Relevance Judgments*

For our experiments, we created queries and relevance judgments using an approach common in patent retrieval community that has been followed in NTCIR [4] and TREC Chemical track [8]. We select 50 patents from the dataset having the most number of in-network citations. The abstract of these patents were used as the input query for different retrieval algorithms with an average query length of 61.23 words and the patents cited by *query patent* present in the dataset constitute the set of relevant documents.

3.2 Baseline Approach

We compare our proposed approach with a state-of-the-art method for exploratory patent analysis as offered by IBM's SIMPLE platform [12]. The search algorithm as used in the SIMPLE platform first breaks the input long query into multiple shorter sub-queries by extracting multiple "keywords" and "phrases" from the original query. It then uses the shorter "sub-queries" with appropriate proximity operators and constructs the final result by combining the results of each individual sub-query. Further, the final ranked list of result is determined based on two factors – *(i)* documents returned by very specific sub-queries should be assigned a higher weight and *(ii)* documents that are returned by many

[1]For extreme cases where the query is a very long sentence, it could be split up into sub-queries of equal length (say 10 words each)
[2]http://www.ir-facility.org/aspire-10

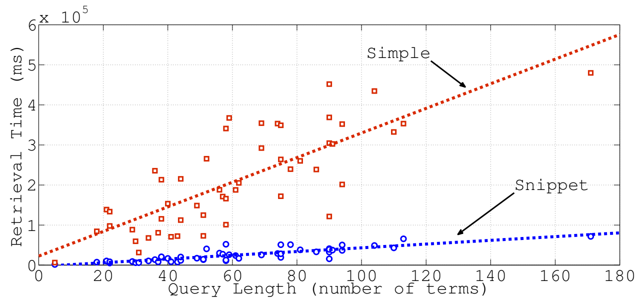

Figure 2: Scatter plot showing query retrieval times with query length (number of words).

different sub-queries are assigned a higher weight. We refer the interested reader to the original paper [12] for further details. We also note that even though there exists methods that tackle the problem of proximal search and verbose queries in isolation, comparing our proposed approach with such techniques is not a fair comparison as our approach targets specific scenarios where it is required to perform proximal search for very long queries.

3.3 Evaluation Metrics

As reported by Magdy and Jones [9], precision oriented metrics such as MAP are not well suited for patent retrieval and can be misleading given the different requirements of patent retrieval task where recall is more important and the user is willing to examine many more documents as compared to a traditional information retrieval setting. Hence, for evaluation we use following recall oriented metrics.

1. **Patent Retrieval Evaluation Score (PRES):** This metric, as proposed by Magdy and Jones [9], evaluates a retrieval algorithm from a recall oriented perspective and takes into account expected search effort from the user. A higher value of the metric indicates that the retrieval algorithm was able to achieve higher recall values at earlier ranks as compared to an algorithm with lower PRES score. Mathematically, PRES for a query Q is computed as:

$$PRES = 1 - \frac{\frac{\sum r_i}{n} - \frac{n+1}{2}}{N_{max}} \quad (1)$$

where, r_i is the rank at which the i^{th} relevant document is retrieved, n is the total number of relevant documents in the corpus for Q, and N_{max} is the maximum number of documents to be examined by the user.

2. **Average Recalls:** We also report recall values at ranks 10, 20, 30, 50, 75 and 100 for different search methods.

All experiments were performed on a machine with four GB of RAM, intel Core-2 quad CPU with 4 cores (2.40 GHz each) and running Ubuntu 10.4 operating system with 2.6.32-41 Linux kernel. Implemetation was done in Java and we used Indri [3] as the base retrieval toolkit.

[3]lemurproject.org

	Recall @ Rank					
	10	20	30	50	75	100
Simple	0.055	0.096	0.129	0.153	0.171	0.184
Snippet	0.064	0.098	0.120	0.150	0.180	0.195

	PRES @ Rank					
	10	20	30	50	75	100
Simple	0.053	0.074	0.090	0.113	0.132	0.144
Snippet	0.064	0.080	0.091	0.112	0.130	0.142

Table 1: Recall and PRES scores at various ranks for the two methods.

4. RESULTS AND DISCUSSIONS

4.1 Time Performance

For each of the two retrieval methods, we ran each query five times and computed the average time taken by each method. This was repeated for all the 50 queries in the dataset giving us average runtime for each method. On an average, the baseline search algorithm (SIMPLE) took 210.854 seconds per query whereas our proposed method took an average of 25.215 seconds – an improvement of more than 700%. Further, Figure 2 shows the average time taken by two methods for each individual query. The figure shows a scatter plot where x-axis represents query length (in number of words) and y-axis represents time. Thus, a point (x, y) on the plot indicates that for a query consisting of x words, the response time is y milliseconds. The dotted lines in the plot represent a least square fit for the two methods – blue for proposed method and red for the baseline method. We note from the figure that our proposed method consistently achieves much lower query response times as compared to the baseline method. Further, the slope of the line representing the baseline approach is much steeper (22536 ms per query word) when compared to the slope for the proposed method (465.74 ms per query word). This indicates that our proposed approach is highly *scalable* as the rate at which the query response times increases with increasing query length is much lower than the baseline approach.

4.2 Retrieval Quality

Table 1 reports PRES and recall values at different ranks for the two methods. We note from the Table that when compared to the baseline approach, our proposed method achieves higher recall at all the ranks except at rank 50. In terms of PRES, even though the PRES values at higher ranks are slightly lower than the baseline method, both PRES and recall at ranks 10, 20 and 30 are higher than the baseline method. This behavior is desirable as it indicates that a larger number of relevant documents are being shown to the user at earlier ranks.

5. CONCLUSIONS AND FUTURE WORK

In this work we described a retrieval method to perform proximal search for very long patent queries in a scalable manner. Our proposed approach achieves 700% faster query response times as compared to a state-of-the-art method while maintaining, and in many cases, improving the quality of returned search results. Even though the problem is motivated from prior art search, the techniques proposed are general enough and can be easily applied to perform proximal search in other domains with very long input queries. In this work, the results for individual sub-queries are combined by using a simple addition function. Our future work will focus on exploring various possible techniques for combining results of sub-queries for improving retrieval performance.

6. REFERENCES

[1] Placing search in context: the concept revisited. *ACM Trans. Inf. Syst.*, 20(1):116–131, Jan. 2002.

[2] M. Bendersky and W. B. Croft. Discovering key concepts in verbose queries. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 491–498, 2008.

[3] S. Büttcher, C. L. A. Clarke, and B. Lushman. Term proximity scoring for ad-hoc retrieval on very large text collections. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 621–622, 2006.

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

[5] D. Hawking. Challenges in enterprise search. In *Proceedings of the 15th Australasian database conference - Volume 27*, ADC '04, pages 15–24, 2004.

[6] D. Hawking and P. Thistlewaite. Proximity operators - so near and yet so far. In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 131–143, 1995.

[7] S. Huston and W. B. Croft. Evaluating verbose query processing techniques. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 291–298, 2010.

[8] M. Lupu, F. Piroi, X. Huang, J. Zhu, and J. Tait. Overview of the trec 2009 chemical ir track. In *TREC-18*, 2009.

[9] W. Magdy and G. J. Jones. PRES: a score metric for evaluating recall-oriented information retrieval applications. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 611–618, 2010.

[10] N. Phan, P. Bailey, and R. Wilkinson. Understanding the relationship of information need specificity to search query length. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 709–710, 2007.

[11] S. Spangler, J. T. Kreulen, and J. Lessler. Generating and browsing multiple taxonomies over a document collection. *Journal of Management Information Systems*, 19(4):pp. 191–212, 2003.

[12] S. Spangler, C. Ying, J. Kreulen, S. Boyer, T. Griffin, A. Alba, L. Kato, A. Lelescu, and S. Yan. Exploratory analytics on patent data sets using the simple platform. *World Patent Information*, 33(4):328 – 339, 2011.

[13] T. Tao and C. Zhai. An exploration of proximity measures in information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 295–302, 2007.

[14] X. Xue, J. Jeon, and W. B. Croft. Retrieval models for question and answer archives. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 475–482, 2008.