Document Retrieval using Proximity-based Phrase Searching

Kathryn Patterson, Carolyn Watters, Michael Shepherd
Faculty of Computer Science, Dalhousie University
Halifax, Nova Scotia, Canada B3H 3L7
kate@cs.dal.ca

Abstract

Exact phrase matching is a powerful tool to quickly retrieve results when a sufficient section of the text is accurately provided as the query. If the section of the text is not completely accurate, phrase searching will fail. A method must be used which will enforce strict enough conditions to achieve high accuracy while allowing for mistakes in the text provided. Here we develop a method using proximity conditions to search for quotes from movies and compare the results against the Vector Space Model. Initial results show a promising accuracy in excess of 78% for documents being successfully ranked within the top ten results.

1. Introduction

In general, the use of complex queries has little impact on web searches, so simple queries are typically sufficient and search engines currently handle simple queries very well [2]. Moreover, increasing the complexity of queries increases the likelihood of errors [5]. However, this does not imply that users should be restricted to simple queries, as they will not always be sufficient.

A user may know nearly the exact text of a section of the document they are looking for. They have a particular piece of text that they wish to use as a query in order to find the source of the text. For example, using a piece of a song to find the lyrics of a song when the rest of the song, the song title or the author is unknown. The text may be part of a quote from an academic paper and the reference may be lost, so the original article must be found.

In this paper we examine the problems of using phrase searching to find known text and propose a method involving the use of proximity searching to improve search results without making the query syntax more complex for users.

Queries of movies quotes are performed on a corpus of movie scripts, the rank of the correct document is examined and the accuracy of the proposed method is compared to searching with the Vector Space Model.

Users often find difficulty in employing advanced query operators. Even one-input interfaces are challenging for new users [12]. The query languages are too complicated for the users to develop advanced queries [13] which would be suitable. Therefore, we have attempted to minimize the complexity of our queries while improving results.

The results of this study show an improvement of approximately 76% in accuracy over the Vector Space model. These results are promising and suggest further examination of this method on different and larger corpora.

Section 2 of this paper briefly reviews related research, Section 3 describes the methodology of our research, Section 4 discusses the experimental results, and Section 5 summarizes the paper and discusses future research.

2. Related Work

The vector space model, VSM, provides a baseline for the purpose of ranking documents based on a query [1]. The similarity between the terms of the document and the terms of the query are determined based on the frequency of the occurrence of the terms in the document. The higher the frequency of the occurrence of query terms, the higher the document will be ranked.

This method has the advantage of not depending on the order of the keywords. If a user is searching for a document on a particular topic, then the name of the topic is likely to appear many times in a useful result.

Lee et al. [10] described how using index terms and keywords alone generally causes poor retrieval performance. However, given that vector space model is widely used in commercial systems, they decided to attempt to modify and improve the method. They compared the performance of their modifications and found only slight improvement for some methods.

Phrase searching enables users to find documents containing the full text that a user seeks [2] [3]. Search engines, such as Google, allow users to provide exact phrases to search for and will only return documents containing the phrase as it appears with the exception of the addition or exemption of punctuation. The more unique the phrase is, the more likely the correct document will appear in the highly ranked documents. Phrase searching is a very efficient way to locate a desired known text, as it is far more likely to achieve the desired result than performing a keyword search [2] [3]. Phrase searching accounted for approximately 10% of queries in a study on the usage of search engines on WebCrawler (http://webcrawler.com/) and Magellan (http://www.mckinley.com/magellan/) [5]. This may be a relatively low percentage compared to multiple keyword queries (67%), but the availability of phrase searching is a necessity in information retrieval when keyword queries have such a low performance as in [10].

Phrase searching can however fail, if incorrect words are used or they appear in the wrong order. In this case, no results will be found and irrelevant results will be retrieved.

It is highly likely a user will not remember a phrase exactly when they wish to use it as a query unless they have the phrase stored somewhere for exact reference. Thus, the user may get some segments of the phrase correct while incorrectly providing the rest of the phrase. As a result, one may be inclined to use a keyword search method such as the Vector Space Model.

A keyword search using terms of a phrase can create problems, if there are not enough unique and frequently-occurring terms provided in the query. Sentence terms such as determiners and stop words [8] are among the sorts of terms we typically use in phrases that would not contribute well to a keyword search and so are essentially ignored.

Sadakane et al. [4] examined the use of a k-word proximity search in which *k*-words were provided by the user and they searched for the smallest interval containing all *k* words in an arbitrary order. They looked at two methods for completing this task to determine the efficiency of finding this minimum interval; a plane sweep algorithm and a divide-and-conquer approach. They found that both algorithms were efficient enough for practical use.

Keen [11] examined the use of four methods of proximity searching. He found that a simple method involving setting a range of 5 words resulted in a Recall of 59% and a Precision of 43%. As he increased the complexity of the proximity methods, the Recall increased and the Precision decreased. The results for all four searches are relatively high compared to that of vector space model searches, as in [5].

One method that aids in the problems incurred with exact phrase matching is the proximity search [3] using the NEAR operator [9]. For situations where the sections of a phrase are known, but joining parts are not, a proximity search can be performed.

Proximity searching is the application of a maximum distance and, optionally, an ordering [9]. Proximity queries are written in expressions with two terms as operands and a proximity operator. For example, *trailer NEAR3 boys* means that there may be no more than three words between *trailer* and *boys* and that they can be in any order. In order to specify that *trailer* must come before *boys*, the operator would then become *WITHIN3*.

If the desired text contains the phrase *"Get your head stuck in your niece's dollhouse because you wanted to see what it was like to be a giant and it's Uncle Conan, you went to Harvard!?"*. If the user remembers *"head stuck"*, and *dollhouse* and *"you went to Harvard"*, they could estimate the distance between the segments they recall and submit a query, such as: *"head stuck" NEAR5 dollhouse WITHIN20 "you went to Harvard"*.

Adding conditions such as proximity will improve results if the user has a reasonably accurate recollection of the maximum distance between sentence segments.

Rosolofo et al. examined the use of a term proximity measurement in combination with the Okapi probabilistic systems [6]. They applied proximity scoring to the top document returned through a keyword search to enhance scoring. They reported that the method was stable and that their method generally improved results.

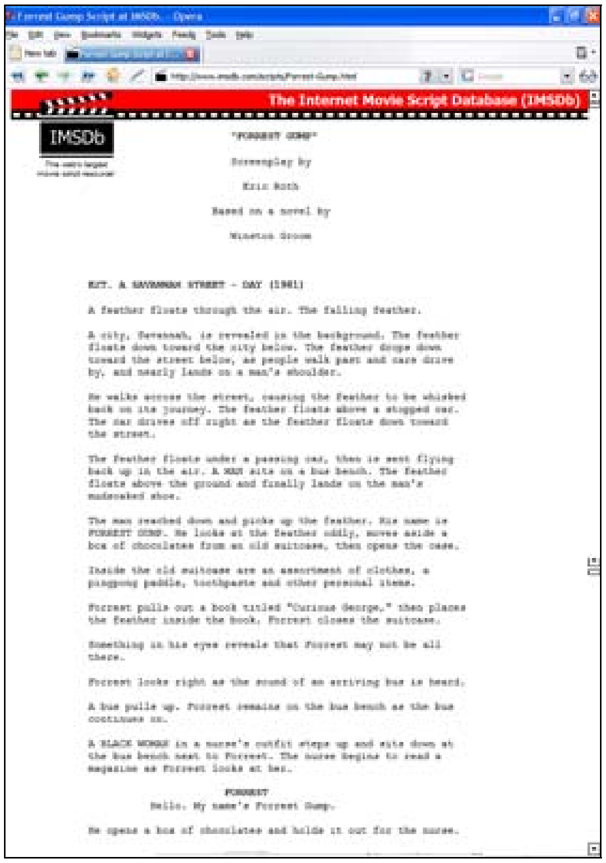
Our interest is in the use of proximity-based phrase search method and testing the retrieval accuracy for retrieving quotes from text.

3. Methodology

The methodology used in this study consisted of selecting all web pages containing scripts for movies from Internet Movie Script Database (IMSDb) to form a corpus, obtaining manually generated user queries, cleaning this data, building term-documents matrices, applying Vector Space Model ranking to all documents for each query, building a corpus-term matrix from the IMSDB scripts, building term-term proximity matrices from query terms, applying a proximity-based scoring system to all documents for each query, developing two modified versions of the proximity-based scoring system based on observations during testing and evaluation of resulting document rankings.

3.1 Dataset

A corpus of 712 complete movie scripts was obtained from the Internet Movie Script Database [7]. From the original collection of 732 movie scripts, any files that were not in the HTML format as in Figure 1, such as PDF format, were omitted. The movie scripts had an average length of 23,766 words.


Forrest Gump Script at IMSDb. - Opera
The Internet Movie Script Database (IMSDb)
IMSDb

"FORREST GUMP"

Screenplay by

Eric Roth

Based on a novel by

Winston Groom

EXT. A SAVANNAH STREET - DAY (1981)

A feather floats through the air. The falling feather.

A city, Savannah, is revealed in the background. The feather floats down toward the city below. The feather drops down toward the street below, as people walk past and cars drive by, and nearly lands on a man's shoulder.

He walks across the street, causing the feather to be whisked back on its journey. The feather floats above a stopped car. The car drives off right as the feather floats down toward the street.

The feather floats under a passing car, then is sent flying back up in the air. A MAN sits on a bus bench. The feather floats above the ground and finally lands on the man's mudcaked shoe.

The man reached down and picks up the feather. His name is FORREST GUMP. He looks at the feather oddly, moves aside a box of chocolates from an old suitcase, then opens the case.

Inside the old suitcase are an assortment of clothes, a pingpong paddle, toothpaste and other personal items.

Forrest pulls out a book titled "Curious George," then places the feather inside the book. Forrest closes the suitcase.

Something in his eyes reveals that Forrest may not be all there.

Forrest looks right as the sound of an arriving bus is heard.

A bus pulls up. Forrest remains on the bus bench as the bus continues on.

A BLACK WOMAN in a nurse's outfit steps up and sits down at the bus bench next to Forrest. The nurse begins to read a magazine as Forrest looks at her.

FORREST
Hello. My name's Forrest Gump.

He opens a box of chocolates and holds it out for the nurse.

Figure 1. IMSDB genres and scripts

The proximity search approach used in this study was based solely on words and relative locations of the words within the scripts. All HTML tags and images, etc., were removed from the web pages.

All words were converted to lowercase and punctuation characters were removed. This resulted in 114,501 unique words in total. The total number of occurrences of each unique term in the corpus was calculated for all words, *totalTF*(*w*), where *w* is a unique word.

A collection of 135 queries were obtained from six university students. They were asked to write down quotes from any movie in the form of a complete or incomplete sentence, indicating "small" and "large" gaps between sentence segments with and ellipses and two consecutive ellipses, respectively. The students were not told what the interpreted maximum size of the gaps would be. The students also provided the name of the movie which they claimed that the quote was from. Of the 135 queries received, 44 of the queries did not have a corresponding movie in the corpus and were removed from the query set, as a result. The remaining 91 queries are referred to as valid queries from this point on.

3.2 Proximity-based phrase searching

Here we discuss alternative querying methods to the Vector Space Model for retrieving documents containing the passages provided in the queries. We will refer to these as the PBP methods.

Queries are given in the form $t_1\ o_1\ t_2\ o_2\ \ldots\ o_{n-1}\ t_n$, where t_i is a term or word and o_i is an operator between the words t_i and t_{i+1}. Each term is a word and each operator is whitespace, an ellipsis or a double-ellipsis. Whitespace operators imply that the adjacent terms ought to be consecutive; however, our constraints will be more relaxed. A single ellipsis implies that there is a small gap between the adjacent terms and a double-ellipsis implies that there is a large gap between the adjacent terms. A few example queries are as follows:

- *here's looking at you kid*
- *Mos Eisley ...... a wretched hive of scum and villainy*
- *My name is ... you kill my father, prepare to die*

Each operand is converted into an allowable distance between the adjacent terms. The following conversions were used:

- Whitespace → maximum of 2 terms apart
- Single Ellipsis → maximum 10 terms apart
- Double Ellipses → maximum 20 terms apart

The distances were chosen arbitrarily and made larger than is likely needed to ensure that the conditions were not too strict. Using these distances, a maximum phrase window size, *M*, is determined in terms of the maximum number of words in the window by adding the total allowable distances and the number of words. For example, the window sizes for the above examples queries are calculated in Figures 2, 3, and 4, resulting in 13, 43 and 36, respectively.

$$\begin{pmatrix} here's \\ \\ looking \\ \\ at \\ \\ you \\ \\ kid \end{pmatrix} \rightarrow \begin{pmatrix} here's \\ NEAR2 \\ looking \\ NEAR2 \\ at \\ NEAR2 \\ you \\ NEAR2 \\ kid \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 2 \\ 1 \\ 2 \\ 1 \\ 2 \\ 1 \\ 2 \\ 1 \end{pmatrix} \rightarrow M = 13$$

Figure 2. Maximum window size calculation for *here's looking at you kid*

$$\begin{pmatrix} Mos \\ \\ Eisley \\ \ldots\ldots \\ a \\ \\ wretched \\ \\ hive \\ \\ of \\ \\ scum \\ \\ and \\ \\ villainy \end{pmatrix} \rightarrow \begin{pmatrix} Mos \\ NEAR2 \\ Eisley \\ NEAR20 \\ a \\ NEAR2 \\ wretched \\ NEAR2 \\ hive \\ NEAR2 \\ of \\ NEAR2 \\ scum \\ NEAR2 \\ and \\ NEAR2 \\ villainy \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 2 \\ 1 \\ 20 \\ 1 \\ 2 \\ 1 \\ 2 \\ 1 \\ 2 \\ 1 \\ 2 \\ 1 \\ 2 \\ 1 \\ 2 \\ 1 \end{pmatrix} \rightarrow M = 43$$

Figure 3. Maximum window size calculation for *Mos Eisley …… a wretched hive of scum and villainy*

For each unique maximum window of size M, a binary vector, f, of size $|t|$ is created which indicates whether each term, t_i, was found in the window. If t_i is found in the window, $f_i = 1$, otherwise $f_i = 0$. Each term, t_i from the phrase may only be counted once, but since terms are not necessary unique, some words may be counted more than once.

The positions of the first and last words that match phrase terms are the first and last words of the actual window, W_i. Given the length of document, *length*, there are *length* – M + 1 windows in that document and each window is denoted by W_j, where j indicates that the first term in the original window of size M is the j^{th} word in the document.

$$\begin{pmatrix} My \\ \\ name \\ \\ is \\ \ldots \\ you \\ \\ kill \\ \\ my \\ \\ father \\ \\ prepare \\ \\ to \\ \\ die \end{pmatrix} \rightarrow \begin{pmatrix} My \\ NEAR2 \\ name \\ NEAR2 \\ is \\ NEAR10 \\ you \\ NEAR2 \\ kill \\ NEAR2 \\ my \\ NEAR2 \\ father \\ NEAR2 \\ prepare \\ NEAR2 \\ to \\ NEAR2 \\ die \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 2 \\ 1 \\ 2 \\ 1 \\ 10 \\ 1 \\ 2 \\ 1 \\ 2 \\ 1 \\ 2 \\ 1 \\ 2 \\ 1 \\ 2 \\ 1 \\ 2 \\ 1 \end{pmatrix} \rightarrow M = 36$$

Figure 4. Maximum window size calculation for *My name is … you kill my father, prepare to die*

For each window, the number of occurrences of adjacent query terms within the window is calculated. The values are stored in a vector, *a*.

A proximity-based scoring system was developed that ranked those documents the highest which maximized the number of query terms found within a minimized window. Each document would have a score given to each window and the highest score found would become the score for that document.

The number of query terms which were found is squared, since we placed higher priority on maximizing the occurrence of query terms than any other factor. The score was increased by the number of occurrences of adjacent query terms. Finally, the percentage of terms in the window which were not query terms is subtracted. This scoring system is expressed in the equation for *PBP-reg*:

$$PBP\text{-}reg = \left|f_j\right|^2 + a_j - \frac{W + \left|f_j\right|}{W}$$

where f_j is a binary array representing which query terms were found in the j^{th} window in terms if a 1 or a 0, a_j represents how many co-occurring query terms were present, and W represents the window length in the number of terms it contains.

Documents were ranked using the PBP-reg system. The resulting rankings of the correct documents were found. If the ranking was not first, the queries and the scripts were examined to determine better ways to match the correct text. Observations made while testing PBP-reg motivated the addition of a factor which incorporates the lengths of matched query terms in PBP-length:

$$\textit{PBP-length} = \left|f_j\right|^2 + a_j - \frac{W + \left|f_j\right|}{W} + \sum_k \frac{length(t_k) f_{jk}}{4}$$

where f_j is a binary array representing which query terms were found in the j^{th} window with a 1 or a 0, a_j represents how many co-occurring query terms were present, W represents the window length in the number of terms it contains, and $length(t_k)$ represents the number of characters in t_k where k is the index of a term in the query.

Finally, an attempt at further improvement upon PBP-reg was made by incorporating the frequency of matched query terms relative to the entire corpus in PBP-tf:

$$\textit{PBP-tf} = \left|f_j\right|^2 + a_j - \frac{W + \left|f_j\right|}{W} + \sum_k \frac{f_{jk}}{totalTF(t_k)}$$

where f_j is a binary array representing which query terms were found in the j^{th} window with a 1 or a 0, a_j represents how many co-occurring query terms were present, W represents the window length in the number of terms it contains, and $totalTF(t_k)$ is the number of times t_k occurs within the entire corpus and k is the index of a term in the query.

3.3 Experimental procedure

All 712 documents were converted into term-frequency vectors. All 91 valid queries were converted into term vectors. The similarity between the query term vectors and the documents was calculated and documents were ranked using the VSM. The top 100 ranked documents were recorded. This experiment is referred to as VSM in the results.

Two-thirds of the 91 queries were chosen at random to be issued to the 712 complete scripts. The three PBP systems were tested on all queries over the whole corpus. The top 100 ranked documents for each query were recorded.

The number of top ranked documents to examine was chosen conservatively. Searching through 100 results is certainly more than human searchers have the tolerance to examine. However, in order to determine whether the methods used are having any positive effect in cases where the results are less than desirable, having a large number of documents recorded can prove useful in improving the method used.

4. Results and discussion

Of the 91 valid queries performed, 12 queries failed to generate the correct document within the top 100 ranked results for all methods. Any query resulting in the correct document not ranking within the top 100 is considered a failed query for the method used. There were two causes of this, shown in Figure 5. There were documents where no segment of the text was similar enough to the query (8 queries) and documents that did not contain the text at all (4 queries).

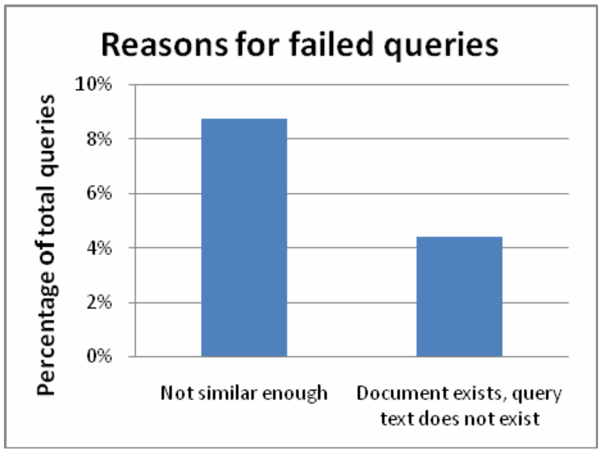


Figure 5. Reasons for failed queries

The results of all four scoring systems are shown in Table 1. Each entry gives the number of queries which successfully ranked the correct document within a top range of documents given for that scoring system.

Table 1. Number of documents with highest ranked documents

	VSM	PBP-reg	PBP-length	PBP-tf
Top 1	1	55	57	59
Top 10	3	66	65	69
Top 50	17	72	75	73
Top 100	23	80	78	78

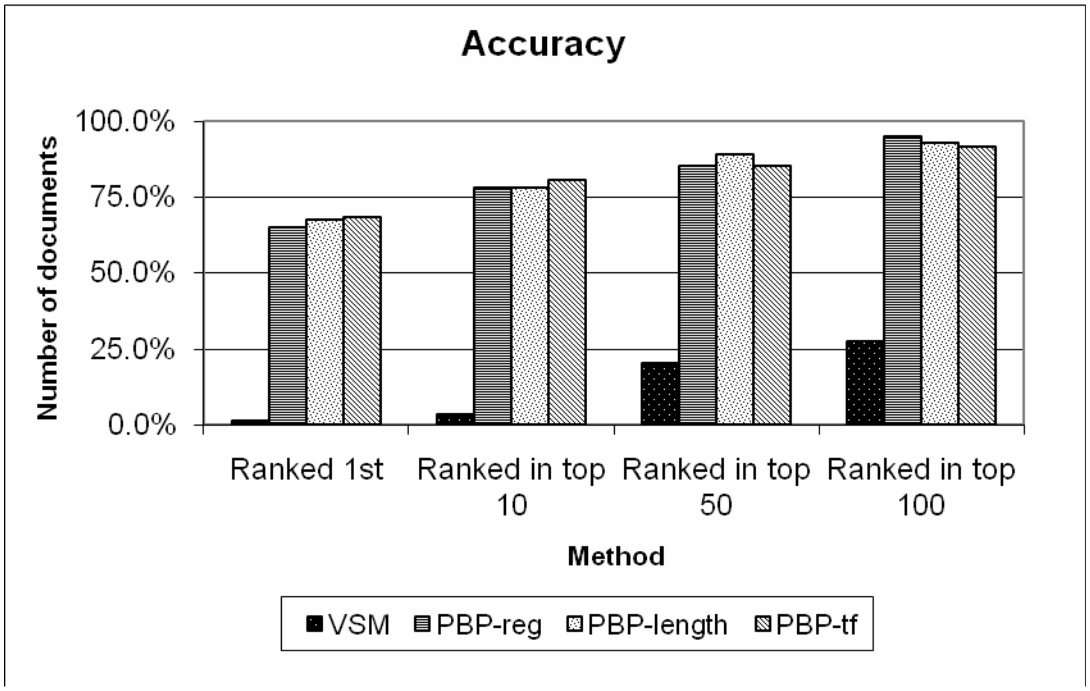


Figure 6. Accuracy

An example of a query which did not rank the correct document as first is *I wasn't even supposed to come in today*. The correct document is *Clerks*. The top five ranked documents from the PBP-tf method matched the following text:

- *Ed-TV:* ***even*** *talks* ***to*** *me and then* ***today I come***
- *Pet-Sematary:* ***wasn't even supposed to*** *be a sprain* ***today****, my friend--that's what* ***I***
- *25th-Hour:* ***supposed to*** *be at work* ***in*** *a couple hours. Christ,* ***I*** *can't* ***even*** *imagine working* ***today***
- *Clerks:* ***even supposed to*** *be here* ***today****!*
- *Wonder-Boys:* ***in*** *Sewickley Heights.* ***I*** *dropped him there once, but... (remembering)* ***Come*** *to think of it, it-****wasn't even***

Looking at the number of matched terms, the lengths of the phrases and guessing the relative frequencies of the words makes it fairly easy to see how the top three ranked documents were placed before the correct document.

A comparison of the accuracy in achieving the correct document within the top ranked documents is shown in Figure 6 and a comparison of the correct document occurring within the top ten ranked documents is shown in Figure 7 in terms of percentage of valid queries. All three of the PBP methods showed a great improvement of accuracy over the accuracy of the VSM method. The PBP methods improved from VSM's accuracy of approximately 3.61% to 79.5%, 78.3% and 81.13% for PBP-red, PBP-length and PBP-tf, respectively.

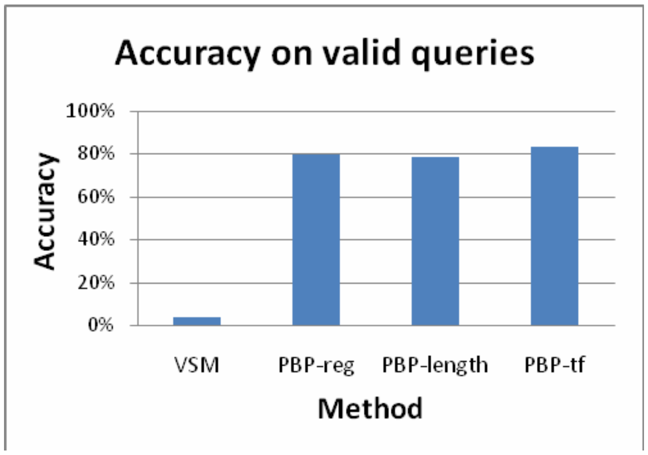


Figure 7. Accuracy on valid queries

The biggest contributing factor is the proximity feature of in all three PBP methods. The purpose of the length and term frequency bonuses in PBP-length and PBP-tf, respectively, was to compensate for the occurrence of multiple documents with the same PBP-reg score and clearly needed to be distinguished based on which words had been matched.

There was no significant difference between the three PBP methods. The PBP-reg method already performed with a high accuracy, as seen in Figure 6, and the modifications made for the PBP-length and PBP-tf methods improved only those few queries which required more factors in order to distinguish the correct result from other close results.

An example of a query that benefited from term frequency bonus is *"It's the genie of the lamp"* which was intended to match the text *"Genie! Of! The Lamp!"* from Aladdin [7]. A number of documents matched *its*, *the*, *of*, and *the* within the same distance of the correct document and since the correct document only matched four terms as well it was not being ranked higher. PBP-reg ranked Aladdin 2nd, PBP-length ranked it as 8th and PBP-tf ranked it 1st. The length of each term is quite similar so it is not surprising that PBP-length did not improve this query result. It should be noted that while PBP-length appeared to disadvantage this query result, the original rank of 2nd was somewhat arbitrary since it shared the same score as eight other documents.

An example of a query which benefited from both the length bonus and the term frequency bonus is *"Excuse me, I believe that's my stapler"* which was intended to match *"Excuse me. I believe you have my stapler?"*. PBP-reg ranked the correct document as 10th, PBP-length at 3rd and PBP-tf as 2nd. *Excuse*, *believe* and *stapler* were given higher priority in both instances as they are both longer and less frequent than the other terms. Putting priority on the word *stapler* through the term-frequency bonus helped push the result up to 2nd place.

We ignored those queries where the document did not exist or for which the text did not exist in the document. A careful examination of the queries which did not have results ranking the correct document within the top ten and the scripts yields a few observations. The first observation is that most of these queries did poorly or failed completely due to the fact that the query had a significant portion written incorrectly. For example, the query *"I could burn the building down"* was intended to match *"I'll set the building on fire"*. While these may have a similar meaning, only one word was matched.

Another observation was the matching text was often spread over a much larger passage than was originally implied by the query.

In terms of efficiency, all three PBP methods outperformed the VSM method, even with the term-document matrices being pre-processed. When tested on five consecutive queries, the PBP methods completed in approximately two-thirds of the time required to complete the same queries using the vector space model.

5. Summary and future research

Keyword searching and phrase searching alone do not provide the flexibility and accuracy needed to retrieve documents where users have an imperfect recollection of a passage from the text. We have presented a method which has impressive accuracy rates within the top ten when the correct document exists. All three PBP methods achieved accuracies in excess of 78%. The query formulation remains very simple for the user and, in particular, does not require any exact guesses of distance.

It is quite clear that the application of proximity constraints is significantly more effective than the Vector Space Model for finding the source of a movie quote or passage. The strong results achieved here suggest that this method is very promising for future experiments on phrase source searching.

Also notable is the efficiency of the PBP methods; taking only two-thirds of the time to run that VSM required. This leaves open the potential for a more complex and effective method that is at least as efficient as VSM.

We have shown here a simple and effective method of phrase searching. Given the troubles that users have formulating complex queries, having search methods as simple as those used in our study are more necessary in information retrieval.

The corpus is a relatively small size. While the documents are each very long, with lengths averaging 23'766 words, many other corpora will contain an order of magnitude more documents, especially on the web. Testing will be done in the future to compare the accuracy of the PBP method to that of keyword search on much larger corpora. From this, we hope to determine the extent of the applicability of this method.

One area for improvement is the errors uses make when supplying their queries. The user is likely to write something that at least has a very similar meaning to the text they had intended. We will investigate methods that retrieve synonyms for individual query words during word matching.

A potential future improvement would be overcoming the constraints imposed by a strict window size by either relaxing the window size further or by providing users with a larger variety of distances to specify.

The proximity method used in this study only applies a maximum window size when a stricter proximity evaluation could be undertaken. We have already calculated the distances between each query to determine the window size and this can also be used to determine the maximum distance between any two terms in the query. It is not yet clear whether applying restrictions to subsections of each window would improve results, for queries with terms that are very general. We hope to develop a method that will perform this evaluation reasonably efficiently to test its usefulness.

Our goal is to build upon the PBP method to further improve its accuracy and ensure users are able to perform phrase searches on larger corpora without increasing complexity to a significant degree.

6. References

[1] Baeza-Yates, R., Ribeiro-Neto, B. Modern Information Retrieval. ACM Press. New York. 1999. pp. 25—30.

[2] Jansen, B. J.. The effect of query complexity on Web searching results. Information Research, Volume 6 No. 1, October 2000.

[3] Eastman, C. M., Jansen, B. J.. Coverage, Relevance, and Ranking: The Impact of Query Operators on Web Search Engine Results. ACM Transactions on Information Systems (TOIS), 2003.

[4] Sadakane, K., Imai, H.. Text Retrieval by using k-word Proximity Search. Database Applications in Non-Traditional Environments, 1999.

[5] Jansen, B., Pooch, U., (In Press). Web user studies: a review of and framework for future research, Journal of the American Society for Information Science, 2000 (draft)

[6] Rasolofo, Y., Savoy, J.. Term Proximity Scoring for Keyword-Based Retrieval Systems. Advances in Information Retrieval: 25th European Conference on IR Research, ECIR 2003, Pisa, Italy, April 14-16, 2003. Proceedings.

[7] The Internet Movie Script Database (IMSDb) site: accessed 2007. http://www.imsdb.com/

[8] SMART FTP site: accessed May, 2007. ftp://ftp.cs.cornell.edu/pub/smart

[9] Proximity searching – Rhodes University Library site: accessed May, 2007. http://www.ru.ac.za/library/infolit/proximity.html

[10] Lee, D. L., Chuang, H., Seamons, K. Documents Ranking and the Vector Space Model. Software, IEEE, Vol. 14, Issue 2, Mar. 1997.

[11] Keen, E.M. Some aspects of proximity searching in text retrieval systems. Journal of Information Science 18, pp. 89-98, 1992.

[12] Lieberman, H., Fry, C., Weitzman, L., Exploring the Web with reconnaissance agents, ACM Press, Vol. 44 No. 8, New York, NY, USA, p. 69-75, 2001.

[13] Hoang, H.H., Tjoa, A., The Virtual Query Language for Information Retrieval in the SemanticLIFE Framework, proceedings of International Workshop on Web Information Systems Modeling (CAiSE), 2006.