International Journal of Computer Applications (0975 8887)
Volume 87 - No. 13, February 2014

Triple Indexing: An Efficient Technique for Fast Phrase Query Evaluation

Shashank Gugnani
BITS-Pilani, K.K. Birla Goa Campus
Goa, India - 403726

Rajendra Kumar Roul
BITS-Pilani, K.K. Birla Goa Campus
Goa, India - 403726

ABSTRACT

Phrase query evaluation is an important task of every search engine. Optimizing the query evaluation time for phrase queries is the biggest threat for the current search engine. Usually, phrase queries are a hassle for standard indexing techniques. This is generally because, merging the posting lists and checking the word ordering takes a lot of time. This paper proposes a new technique called Triple Indexing to index web documents which optimizes query evaluation time for phrase queries by reducing the time for merging the posting lists and checking the word ordering. In addition, a proper procedure has been put forward for document ranking using an extended vector space model. The 4 Universities dataset and Industry Sector dataset of Carnegie Mellon University has been used for experimental purpose and it has been found that using the proposed method with a modern machine, the query time for phrase queries is reduced by almost 50 percent, compared to a standard inverted index.

General Terms:

Web Document Indexing, Phrase Queries

Keywords:

Triple Index, Inverted Index, Query Optimization, Phrase Queries, Vector Space Model

1. INTRODUCTION

Search engines generally use an inverted index [11] to execute text queries. Because inverted indexes are expensive to update, search engines usually recreate the index quite often. The index should be updated as often as possible, so that it reflects the latest data on the web, and gives the best results. A large amount of research on high performance inverted indices has been carried out, because most web search engines use an inverted index to execute text queries. Generally, documents are stored as list of words, where as in inverted indices, each word is stored with the list of documents that the word appears in.
Phrase based querying in inverted indices is highly inefficient as shown by Bahle et al. [4]. This is because each word in the query has to be searched separately in the index and then the document list is merged. In addition, the position of each and every word has to be checked to ensure that the phrase is present as it is in the document. Many indexing methods have been proposed for efficient phrase based querying, like Suffix Tree Method [14] and Document Index Graph Method [7]. This paper proposes a technique for indexing web pages for efficient phrase based querying called Triple Indexing. The paper is organized in the following lines. Section 2 covers related work. Section 3 covers the proposed approach, describing the construction, working, and ranking procedure of the technique. Section 4 describes experiments performed and presents the results. Finally, Section 5 summarizes and concludes the work.

2. RELATED WORK

Many techniques have been proposed for phrase based indexing. Document Index Graph [7] is a directed graph built using the documents and the ordering of words in the documents. Each word is a vertex in the graph and each edge represents a sequence of two words found in the text. By using a graph, redundancy is reduced and phrase querying is made easy. This method provides for an efficient correlation between different words and a coherent document clustering method.
Suffix Trees [14] are efficient data structures, which can be used to index documents, and support searching for any random phrase. Each node in the tree corresponds to a unique phrase. Although suffix tree is criticized for consuming a large amount of space, it is still useful for efficient phrase based querying and document clustering. Space efficient versions of suffix trees have also been proposed which take up space close to the size of the indexed text.
Patil et al. [12] propose a new data structure, Conditional Inverted Lists to further reduce the space for indexing phrases. The key idea is to select certain important nodes in the suffix tree for which an inverted list is maintained. However, not all phrases are indexed, and hence, not all phrase queries can be directly processed by the inverted list.
Lim et al. [10] propose a method to rebuild the inverted index of those indexed documents which have been changed by their content. A landmark-diff method has been used in their approach to reduce the number postings in the inverted index that are necessary to be updated.
Delbru et al. [5] propose a entity retrieval model and introduce a methodology for indexing semi-structured data. They show that their model is scalable and evaluates queries in sub-second time.
Hsu et al. [8] propose a technique known as CIS-X to index XML documents. The idea is to build a summary tree for the XML data tree which will be regarded as the index of the XML document.
In practical situations, retrieving the document list is not enough. Among all the retrieved documents, only the top k documents which are highly relevant to the query attract the user's interest. Thus, there

Table 1. : Sample Documents

Document 1	Document 2	Document 3
weather cold today	very cold	weather today very cold
yesterday hot	hot yesterday	weather very hot yesterday

needs to be a document ranking method to find out the most relevant documents. The Vector Space Model [9], is one of the most popular models for ranking web documents. It uses *tf-idf* value along with a cosine similarity formula to calculate scores for each document. After indexing the web documents, this paper proposes an method to rank the documents, which is an extension of the Vector Space Model.

3. PROPOSED APPROACH

3.1 Triple Index

A Triple Index is basically a three level index, individually referred to as the primary index, secondary index and tertiary index. The primary index is a normal inverted index with the addition of a secondary index for each entry in the primary index. The secondary index contains words which follow the word in the corresponding primary index. Posting lists contain documents which contain both the words in order. Each secondary index entry has a tertiary index working just like the secondary index. It is built such that any sequence of words from the three indices will be a phrase which is present in the documents given by the tertiary index. In other words, the triple index stores phrases of length up to 3 words.
Why Triple Indexing? According to Zhang et al. [15] the average query length is just 2.4 words. Thus, majority of queries are of 3 words or less, which can be directly processed by the Triple Index without the need for any list merging or word order checking. This improves the query evaluation time by a huge margin and uses less space as compared to a Suffix Tree or a Document Index Graph.

3.2 Construction of Index

The construction begins by searching through each document and adding the new words and phrases to the index as and when found. The construction will be done from the first word by analyzing groups of three words at a time and then moving on to the second word and so on. An example has been made which takes three documents (see Table 1) and builds a triple index from them. Fig. 1 illustrates the completed index. Blue arrows show a link between an element in the primary index and its corresponding secondary index and red arrows show a link between an element in the secondary index and its corresponding tertiary index.

3.3 Searching the Index

Boolean Queries: Boolean Queries are performed as they would be in an inverted index, using only the primary index to get the document lists and then merging them using a zigzag join algorithm [3, 6, 13].
Phrase Queries: The index is searched using a divide and conquer algorithm (see Fig. 2). The search string is divided by two until there are less than or equal to 2 words in each group. Then each group is searched for in the Triple Index and a list of documents containing the group string is returned. After this the returned lists are merged. This is done by first eliminating documents in which the merged string is not present as it is (two group strings may be present separately in the document but the combined string may not be present) and then joining the document lists using a zigzag join algorithm. The algorithm in Fig. 2 describes the searching procedure for phrase based searches. If the query has 3 or less terms, then we can directly search the index without the use of the search algorithm.

Table 2. : TF * IDF (single word terms)

Terms	Document 1	Document 2	Document 3
weather	0.035	0	0.044
cold	0	0	0
today	0.035	0	0.022
very	0	0.044	0.044
hot	0	0	0
yesterday	0	0	0

Table 3. : TF * IDF (two word terms)

Terms	Document 1	Document 2	Document 3
weather cold	0.119	0	0
weather today	0	0	0.068
weather very	0	0	0.068
cold today	0.119	0	0
today very	0	0	0.068
yesterday hot	0.119	0	0
hot yesterday	0	0.058	0.025
very cold	0	0.058	0.025
very hot	0	0	0.068

3.4 Space Time Analysis

Space: Since the index has three levels, it occupies more space than a regular inverted index. A Suffix Tree has levels up to the largest length of a sentence. Hence, the index uses less space than a Suffix Tree. The index uses memory between that used by an Inverted Index and a Suffix Tree.
Time: With three levels of the index and the proposed divide and conquer approach, the index performs well for phrase based searches and has a time complexity of O(n + output), where *n* is the number of words in the phrase searched and the output is the number of documents given as a result.

3.5 Extending the Vector Space Model

To construct a fully functional search engine using the Triple Index, a document ranking procedure is needed. For web document ranking, Vector Space Model (VSM) is one of the most popular technique. This method proposes an extended VSM for retrieving the top k documents.
A standard VSM is constructed for each document. For phrase based querying, a second vector space is constructed for each document on groups of two words instead of one. Table 2 and Table 3 illustrate the two vector space models built on the sample documents in Table 1. The evaluation of phrase queries submitted by the user can be described by a procedure as follows:

(1) All phrase queries submitted by the user will be evaluated using the search algorithm proposed in Fig. 2.
(2) Document ranking will be done using the two word term VSM. The query is divided into two word groups as proposed by the divide and conquer approach and then the query vector is calculated. For odd length queries one group has only one word

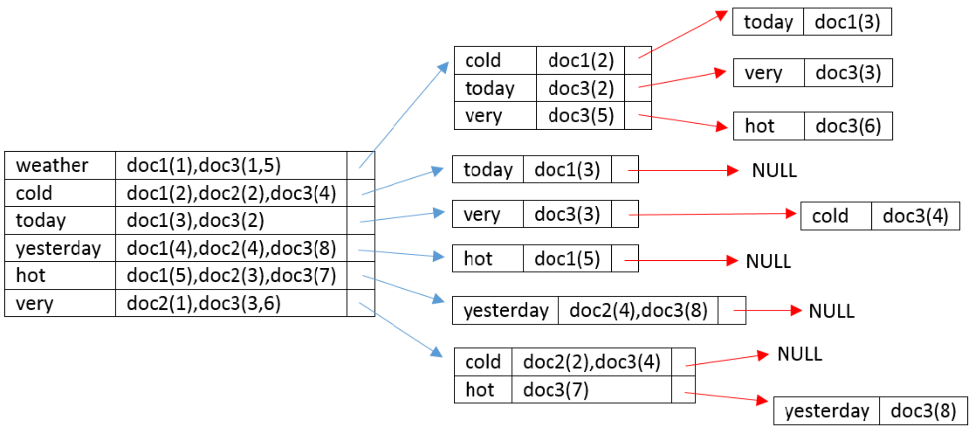

Fig. 1: Triple Index constructed on the sample documents

Algorithm for Phrase Searching

```
Phrase_Search(String s[1 ... n])
begin
  if(n>2)then
    d1<-Phrase_Search(s[1 ... n/2])
    d2<-Phrase_Search(s[n/2+1 ... n])
    if(check(s[1 ... n/2], s[n/2+1 ... n])then //checks if combined string
      dl<-d1 intersection d2                      is in the document
      return dl
    else
      return NULL
  else
    d<-Search(s[1 ... n]) //returns the document list containing the string
    return d
end
```

Fig. 2: Algorithm 1

and the weight for that group is taken from the one word term VSM.

(3) The cosine similarity [9] between a document D_i and a query Q is then calculated for each document using the following formula:

$$Sim\left(Q, D_i\right) = \frac{\sum_{j=1}^{V} w_{Q,j} \times w_{i,j}}{\sqrt{\text{number of terms in } D_i}} \tag{1}$$

where $w_{Q,j}$ is the weight of term j in the query, and is calculated just like $w_{i,j}$ ($tf_{Q,j}$ x idf_j) and V is the dictionary size.
The square root of the number of words in the document is used as the normalization factor. Although, this still favors long documents, the normalization reduces the effect of document size on the cosine similarity. This normalization factor is easier to compute than the standard one, also, pre-computation is possible.

(4) After the similarity has been calculated for each document which has been selected for the result, the documents are ranked based on their similarity measure and the top k documents are returned to the user.

4. COMPUTATIONAL RESULTS

For experimental purpose and to find out the effectiveness of the proposed algorithm, the 4 Universities dataset [1] from the World Wide Knowledge Base project and the Industry Sector dataset [2] of Carnegie Mellon University was used. The datasets combined contain more than 13,000 web documents. The algorithm proposed and the standard inverted index algorithm was run on the dataset to compare both the techniques. A machine with Intel Core 2 Duo Processor, 2.1 GHz, with 4GB main memory and running Ubuntu 13.04 was used to execute the algorithms.
Fig. 3 shows the graph for index construction time v/s number of documents used. It can be observed that a triple index takes more time to construct than an inverted index. However, the time complexity of both the algorithms is same, i.e. $O(NlogN)$, where N is the number of documents.
Since, the proposed algorithm is designed to improve performance for phrase queries, the algorithms were tested for only phrase queries. A set of 50 phrase queries was prepared to evaluate the query evaluation time for each algorithm. It was observed that the average query evaluation time for Triple Index is almost half of the average

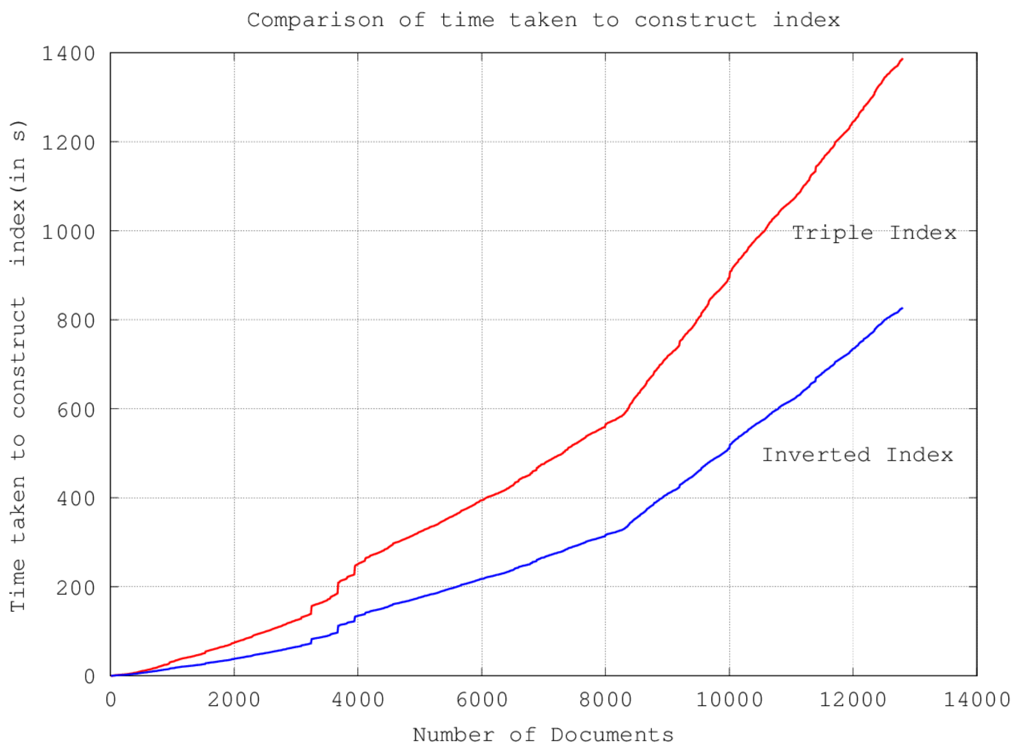

Fig. 3: Index Build Time Comparison

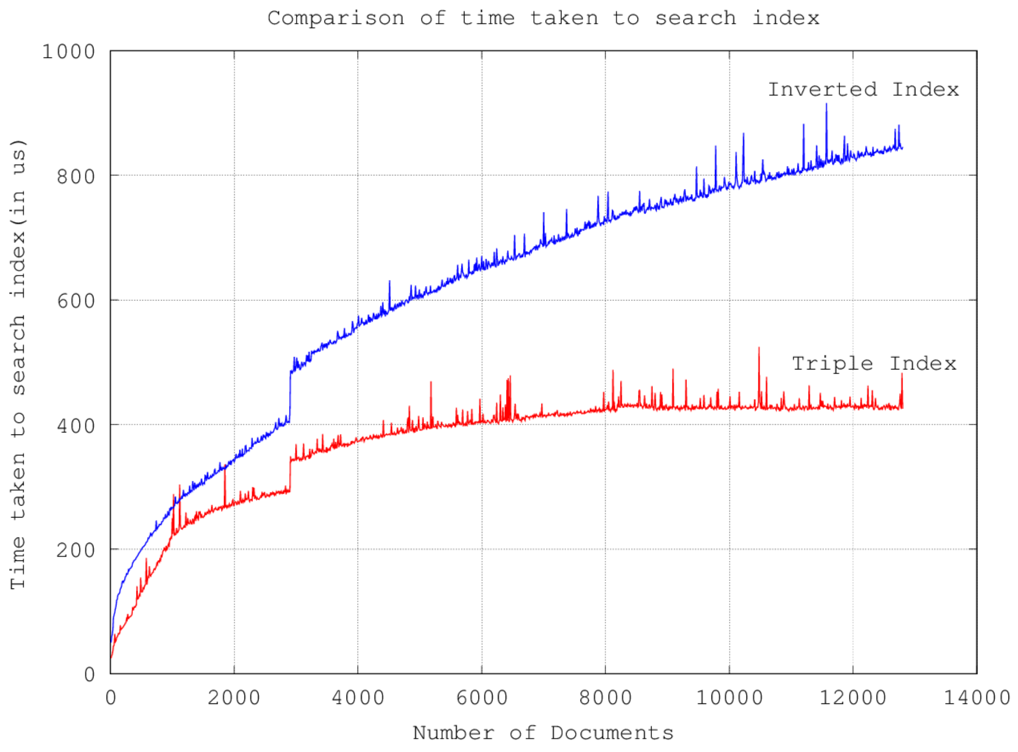

Fig. 4: Search Time Comparison

query evaluation time of an Inverted Index when the number of documents used exceeds 12,000. Fig. 4 shows the graph for average query time v/s number of documents used for both indices.

5. CONCLUSION

This paper highlights the problem of standard indexing algorithms to resolve phrase queries, and proposed an algorithm to solve this issue. In addition, an extended vector space model was put forward to rank the retrieved documents efficiently. A procedure was described to resolve the queries submitted by the user. From experimental results it has been observed that although compared to an inverted index, this approach takes more space and time to construct, but it reduces the query evaluation time of phrase queries by almost 50 percent. Overall, for indexing web documents it is a better algorithm compared to an inverted index.

6. REFERENCES

[1] The 4 universities data set. Available Online at: `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/`.

[2] The industry sector data set. Available Online at: `http://www.cs.cmu.edu/TextLearning/datasets.html`.

[3] Gennady Antoshenkov and Mohamed Ziauddin. Query processing and optimization in oracle rdb. *The VLDB JournalThe International Journal on Very Large Data Bases*, 5(4):229–237, 1996.

[4] Dirk Bahle, Hugh E Williams, and Justin Zobel. Optimised phrase querying and browsing of large text databases. In *Australian Computer Science Communications*, volume 23, pages 11–19. IEEE Computer Society, 2001.

[5] Renaud Delbru, Stephane Campinas, and Giovanni Tummarello. Searching web data: An entity retrieval and high-performance indexing model. *Web Semantics: Science, Services and Agents on the World Wide Web*, 10(0):33 – 58, 2012. Web-Scale Semantic Information Processing.

[6] Tingjian Ge. Join queries on uncertain data: Semantics and efficient processing. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 697–708. IEEE, 2011.

[7] Khaled M Hammouda and Mohamed S Kamel. Efficient phrase-based document indexing for web document clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 16(10):1279–1296, 2004.

[8] Wen-Chiao Hsu and I-En Liao. Cis-x: A compacted indexing scheme for efficient query evaluation of {XML} documents. *Information Sciences*, 241(0):195 – 211, 2013.

[9] Dik L Lee, Huei Chuang, and Kent Seamons. Document ranking and the vector-space model. *Software, IEEE*, 14(2):67–75, 1997.

[10] Lipyeow Lim, Min Wang, Sriram Padmanabhan, Jeffrey Scott Vitter, and Ramesh Agarwal. Efficient update of indexes for dynamically changing web documents. *World Wide Web*, 10(1):37–69, 2007.

[11] Ajit Kumar Mahapatra and Sitanath Biswas. Inverted indexes: Types and techniques. *International Journal of Computer Science*, 8.

[12] Manish Patil, Sharma V Thankachan, Rahul Shah, Wing-Kai Hon, Jeffrey Scott Vitter, and Sabrina Chandrasekaran. Inverted indexes for phrases and strings. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 555–564. ACM, 2011.

[13] Guoren Wang, Ge Yu, Kunihiko Kaneko, and Akifumi Makinouchi. Comparison of parallel algorithms for path expression query in object database systems. In *Database Systems for Advanced Applications, 2001. Proceedings. Seventh International Conference on*, pages 250–257. IEEE, 2001.

[14] Ruilong Yang, Qingsheng Zhu, and Yunni Xia. A novel weighted phrase-based similarity for web documents clustering. *Journal of Software*, 6(8):1521–1528, 2011.

[15] Yuye Zhang and Alistair Moffat. Some observations on user search behavior. *Australian Journal of Intelligent Information Processing Systems*, 9(2):1–8, 2006.