Journal of Computer Science 3 (6): 436-440, 2007
ISSN 1549-3636

Enhancement of Search Results Using Dynamic Document Seed Reranking Algorithm

[1]Angelina Geetha and [2]A. Kannan
[1]Department of Information Technology, B.S.A. Crescent Engineering College, Anna University, India
[2]Department of Computer Science and Engineering, Guindy College of Engineering,
Anna University, India

Abstract: We proposed an algorithm to improve the precision of top retrieved documents by re-ordering the retrieved documents in the initial retrieval. To re-order the documents, we first automatically extract key terms and key phrases from top N retrieved documents and generate a document index for each document. Using the standard similarity metrics, a document similarity matrix is generated for these documents. The document similarity matrix is considered as an adjacency matrix, where the nodes are documents and the distances are their similarity measures. The objective of this algorithm is, to rerank the documents so as to minimize the similarity mean absolute distance between them. Moreover, the user can choose a document of interest as the seed document and initiate the reranking algorithm by which documents are reranked based on is similarity distance from the seed document. From the experimental results, it is demonstrated that the algorithm reduces the mean absolute difference. Further it is proved that the proposed reranking algorithm minimizes the mean absolute distance between the top N results obtained from search engines and helps users to rerank documents based on any seed document as a query.

Key words: Search result reranking, Similarity metric, Document retrieval

INTRODUCTION

The problem of searching on the World Wide Web (WWW), which is the process of discovering pages that are relevant to a given query. The commonly used tool to search on the web is a search engine. The process of determining the relevance ranking of search results based on a given query is still a challenge. The discomfort faced by users of search engines is two fold. One is the users feel they are unable to clearly specify what they need to search in the form of a query. Second is that though the query is given, the search results given by a search engine, is not well ranked. The objective of this paper is to address these two problems. When a user initiates a search process often he himself is not clear about what exactly he needs from that search process. The user refines his search query based on the initial search results. The user given query is the beginning for the process of searching on the web. Jon. M. Kleinberg[1] has classified queries in the following types.

- Specific queries. Example " Database support by Java using JDBC"
- Broad Topic queries. Example " Find information about Database connectivity"
- Similar queries. Example "Find pages 'similar' to java.sun.com"

The difficulty in handling specific queries is centered roughly, around what could be called the scarcity problem. There are very few pages that contain the required information, and it is often difficult to determine the identity of these pages. For broad topic queries, on the other hand, one expects to find many thousand relevant pages on the web, which may be generated by variants of term matching. The fundamental difficulty lies in what could be termed as 'Abundance problem: The number of pages that could reasonably be returned as relevant is far too large for a human user to digest'. For the third type of query, the challenge is to extract the features of a given page and then initiate the search. The user may feel a particular web document closer to his search and may look for documents similar to it. This can be achieved by

Corresponding Author: Angelina Geetha, Department of Information Technology, B.S.A. Crescent Engineering College, Seethakathi Estate, Vandalur, Chennai - 600 048. TAMIL NADU, INDIA

document reranking based on the features of the selected document. In our work we have developed an algorithm where the user can dynamically choose a particular web document and present it as a seed. The key features of this seed document are heuristically extracted to create an index for that document. Similarly document index is created for all the search result documents. Based on the similarity measure between the seed document and all the other documents, the algorithm reranks the remaining search results in order to minimize the similarity mean absolute distance between them. Most similar documents are ranked higher than the dissimilar documents.

The objective of a ranking function is to match the documents in a text collection against a query and order them in descending order of their predicted relevance. The similarity between a query and a document can be calculated by the widely used cosine measure given by Salton[2]. Documents are then ordered by decreasing values of this measure. In the vector space model, these weights are commonly measured by their statistical properties or statistical features. For example, one of the most widely used statistical features in term weighting strategy is *term frequency* (TF), which measures how many times the term has appeared in the document or query[2]. Another commonly used feature is the *inverse document frequency* (IDF), which can be calculated by log (*N/DF*), where *N* is the total number of documents in the text collection and *DF* is another feature that measures the number of documents in which the term has appeared in the document collection.

Rorvig[3] studied the impact of ranking / similarity functions on visual information retrieval (IR). In visual IR, not only the similarity between query and document, but also the relationships among documents need to be visualized. Rorvig used multidimensional scaling to visualize document similarities using five different similarity functions. A key finding in all of these studies is that a single ranking function cannot work well for all contexts.

Many methods have been proposed to rerank documents. In the literature, Lee et al [4] proposed a document reranking method based on document clusters. They build a hierarchical cluster structure for the whole document set and use the structure to rerank the documents. In the works of Balinski [5] a document reranking method was proposed, that uses the distance between documents to modify initial relevance weights. Crouch et al [6] used the unstemmed words in the queries to reorder the documents. Xu et al [7] made use of global and local information to do local context analysis and then use the information acquired to rerank documents. Manually built thesaurus was also used to rerank retrieved documents[8]. Each term in a query topic is expanded with a group of terms in the thesaurus. Bear et al [9] used manually crafted grammars for topics to reorder documents by matching grammar rules in some segments of an article. Kamps [10] proposed a reranking method based on assigned, controlled vocabularies. Yang et al [11] used query terms that occurred in both query and top N (N<=30) retrieved documents to rerank documents.

For a given query q, we first obtain a set of documents retrieved and ranked by an external search engine. We propose a document reranking algorithm where the user selects a document as the seed for the reranking procedure. The similarity weightage is calculated based on the importance of query key term weightage, document term frequency and document distance as in the case of vector model[2]. But the importance are not calculated globally for entire search result documents but only for its subset whose members are relevant to the given query q. Consequently, the implementation depends on the search engine used. We have considered Google web search engine for the purpose of our research.

Our algorithm initially accepts a query from the user, extracts the key terms from the query. The top N search results are acquired from any search engine. The dynamic reranking algorithm generates a distance matrix for the top N documents.

Once the user selects a particular document as the seed, based on the document distance metrics, the search results are reranked, in such a way that documents similar to the given document appears closer. The objective of the algorithm is to minimize the similarity mean absolute distance between the documents.

SYSTEM ARCHTECTURE

Figure 1 depicts the system architecture. Initially the user gives a query to search for. This query is given to the search system. The stop words are removed from the query and the key terms are given to any external search engine to search the Internet. From these results the user can browse and choose the seed document.

The Document Index Generator generates the index vector for every web document. The key features are extracted and stored as an index vector for each document. When the user selects a seed document and requests reranking, the Dynamic seed reranker algorithm is initiated. The various similarity metrics as

discussed below are applied and the documents are reranked based on their similarity to the selected seed document. Thus reranked results are given to the user. The user can choose again a new seed and request for reranking again or the user may opt for rephrasing the query itself.

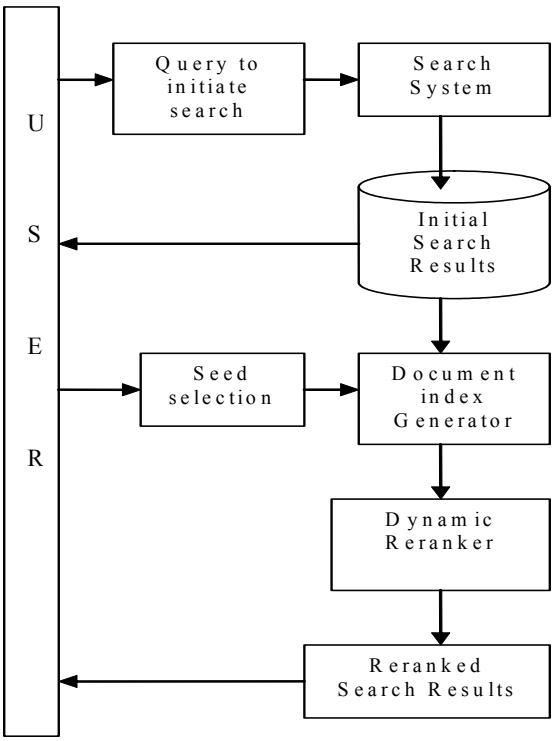

Fig. 1: System Architecture

Search System: Keywords are extracted from the given query. The extracted keywords are converted into a string by placing '+' symbol between them and this is given to an external search engine (say Google) and the search is triggered. The results from the search engine are captured and the system stores the URLs of the search result documents in a database for further use by the index generator.

Document Index Generator : In this module, the URLs of the search result documents are retrieved from the database. Every web document is retrieved and detagged. We have restricted our work to only web documents of HTML format and text format. The task of feature extraction focuses on the key term extraction. All the stop words are removed. Stemming of words is also considered. For example 'network', 'networking', networked' are considered alike. The following three parameters are calculated.

The term frequencies of the key terms are tabulated. Term frequency (TF) is how many times a particular key term has occurred in the document or query[1]. For similarity measure we define a heuristic technique which states the density of key term distribution reflects on the importance of that term in the document. Hence the Term Density measure is also calculated. Term Density Measure (TDM) is the mean distance between the successive occurrences of the term. Let x_1, x_2, x_3... x_n be the occurrences of the keyword x in the document. Then the mean distance is calculated as

$$TDM = \sum_{i=1}^{TF-1} dist(x_i, x_{i+1}) / (TF-1)$$

Where, dist (x_i, x_{i+1}) is the number of words in between the successive occurrences of a particular key term. The maximum inter term distance measure is limited to a cutoff value, in our case it is set as 8. For documents with TF=1, TDM is set to zero. The lesser the TDM, the closer they appear in the document. Hence for every document a term index is generated with terms whose TDM is below the cutoff value.

In this algorithm we have considered the key phrases also. Considering the time delay we have restricted our key phrases to a length of two words only. If there are n key terms a Key Phrase Matrix (KPM) of size n x n is generated, and the frequency of the occurrences of the key phrases are computed and stored in this matrix.

$$KPM(i,j) = x$$

indicates that key terms words i and j occur next to each other x times. We have considered KPM (i,j) as equal to KPM (j,i). For example the phrase 'Programming in network' is considered as the same key phrase as 'network and programming'. Hence the upper diagonal matrix alone has to be calculated. KPM (i,i) is ignored. Though the key terms are very high, we found that the KPM is highly sparse and does not need very high memory storage, since we considered storing only the nonzero elements of the matrix.

Document Seed reranking Metrics: The key features of each document are indexed by the index generator. The following metrics are applied and the overall similarity measure of each document with respect to the given seed document is calculated. The vector of key terms of seed document is taken as X. Y is the vector of

the document (from the rest N-1 documents) to which the similarity to the seed document is to be calculated.

Matching Coefficient (MC):

The Matching Coefficient is a simple vector based approach which counts the number of terms, (dimensions), on which both vectors are non zero. So for vector set X of document A and vector set Y of document B, the matching coefficient is |X & Y|. This can be seen as the vector based count of co-referent terms. For this the position of occurrence of terms is not taken into account. Hence for any two documents A and B, the Matching coefficient (MC) based on terms is denoted as $MC_{t(A,B)}$

Dice Coefficient (DC):

Dice coefficient is a term based similarity measure (0-1) whereby the similarity measure is defined as twice the number of terms common to compared entities divided by the total number of terms in both tested entities. For any two documents A and B, the Dice coefficient (DC) based on terms is denoted as $DC_{t(A,B)}$

Jaccard Similarity (JS):

Jaccard Similarity uses word sets from the comparison instances to evaluate similarity. The Jaccard similarity penalizes a small number of shared entries (as a portion of all non-zero entries) more than the Dice coefficient. Each instance is represented as a Jaccard vector similarity function.

The Jaccard similarity between two vectors X and Y is

$$(X*Y) / (|X||Y|-(X*Y))$$

Where (X*Y) is the inner product of X and Y, and |X| = (X*X)^1/2, i.e. the Euclidean norm of X. The Jaccard similarity between two documents A and B denoted by term vectors X and Y respectively is denoted by $JS_{t(A,B)}$. For key phrase similarity, it is denoted as $JS_{kp(A,B)}$.

Document Seed Reranking Algorithm: The similarity between two documents are measured based on term similarity (TS) and key phrase similarity (KPS) as given below. The overall document similarity metric is computed by giving additional weightage for term similarity over key phrase similarity

$$TS_{(A,B)} = ((MC_{t(A,B)} + DC_{t(A,B)} + JS_{t(A,B)})/3) * 100$$
$$KPS_{(A,B)} = JS_{kp(A,B)} * 100$$
$$DocSim(A,B) = (3*TS_{(A,B)} + KPS_{(A,B)}) / 4$$

Based on this, a document similarity (DS) matrix of size NxN is generated, where N is the number of documents considered. DS(A,B) indicates the cell denoted by the A^{th} row and B^{th} column which specifies the similarity between two documents A and B. Note that DS(A,B) is not same as DS(B,A). For a seed document D, the D^{th} column of the matrix, a linear array x_1 to x_N is extracted from the matrix. The documents are reranked in descending order of their closeness of similarity with the seed document D. This ordering minimizes the overall similarity mean absolute distance (MAD) between documents. The similarity mean absolute distance (MAD) between documents for a given query Q and N search result documents, is defined as,

$$MAD_Q = [\sum \left| (x_{i+1} - x_i) \right| \text{ for } 1 \le i \le (N-1)] \ / \ N$$

where, x denotes the similarity distance between two successive documents.

Algorithm: Initiate the search process using the query given by the user.

For the top N search result documents

Compute the document index consisting of terms based on TF and TDM

Generate the key phrase matrix (KPM)

For each web document (i = 1 to N) do

Let A = i

For each web documents

(j = 1 to N) AND (j != i) do

Let B = j

Compute SimVal(A,B)

$TS_{(A,B)} = (MC_{t(A,B)} + DC_{t(A,B)} + JS_{t(A,B)}) / 3$

$KPS_{(A,B)} = JS_{kp(A,B)}$

$DocSim(A,B) = (3*TS_{(A,B)} + KPS_{(A,B)}) / 4$

Fill the Document Similarity (DS) matrix (i,j) with DocSim(A,B)

End for

End for

Accept the seed document S, extract the linear array of S from DS

Rerank in decreasing order of their similarity distance.

RESULTS AND DISCUSSION

For the search result obtained from Google for the query "data structures and algorithms", we have generated the Document Similarity matrix for top 10 search results. Considering the first search result document as the seed document, the reranking based on this seed document is given in graph 1. Table 1 gives the computation of MAD before reranking and table 2 gives the computation of MAD after reranking. By applying our algorithm, it is found that, there is significant improvement in mean document distance.

Table 1: MAD without reranking

Document links	Measure of similarity	$\mid X_i - X_{i+1} \mid$
1-2	5.55	
1-3	3.79	1.76
1-4	2.87	0.92
1-5	4.19	1.32
1-6	27.77	23.53
1-7	3.78	23.99
1-8	7.8	4.02
1-9	14.13	6.33
1-10	18.49	4.36
$MAD_Q = [\sum \mid (x_{i+1} - x_i) \mid$ for $1 \le i \le (N-1)] / N$		6.623

Table 2: MAD with reranking

Document links	Measure of similarity	$\mid X_i - X_{i+1} \mid$
1-6	27.77	
1-10	18.49	9.28
1-9	14.13	4.36
1-8	7.8	6.33
1-2	5.55	2.25
1-5	4.19	1.36
1-3	3.79	0.4
1-7	3.78	0.1
1-4	2.87	0.91
$MAD_Q = [\sum \mid (x_{i+1} - x_i) \mid$ for $1 \le i \le (N-1)] / N$		2.499

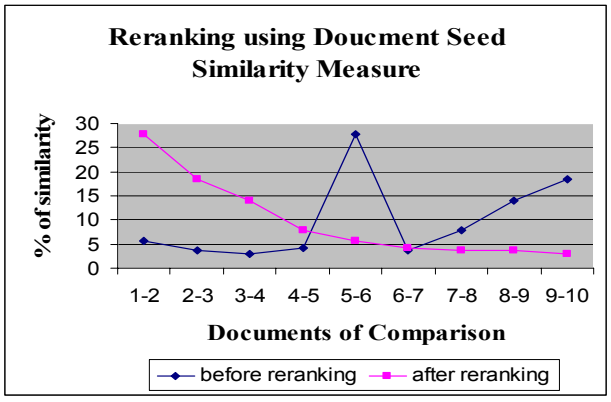

Fig. 1: Similarity between documents before and after reranking

CONCLUSIONS

Search engines are enhancing their search algorithms so as to answer the queries of the user. Various ranking and reranking algorithms focus on similarity between the user query and the search results. In this paper we have developed an algorithm which accepts a web document as seed document extracts the features from the search result documents and reranks the documents based on their degree of similarity with the seed document. This is useful for users who are not very specific about their search process and would like to explore from the initial search documents. The future work is focused on implementing an algorithm which lists the features of the seed document and the user can choose the features of his interest to initiate the reranking process. The work can be extended for multiple document seeds.

REFERENCES

1. Kleinberg J. M., 1999. Authoritative Sources in a Hyperlinked Environment. Journal of the ACM, 46 (5) : 604 - 632.
2. Salton, G., and Buckley, C., 1988. Term-Weighting Approaches in Automatic Text Retrieval. Information Processing and Management, 24 (5) : 513-523.
3. Rorvig M., 1999. Images of Similarity: A Visual Exploration of Optimal Similarity Metrics and Scaling Properties of TREC Topic-document Sets. Journal of the American Society for Information Science, 50 (8) : 639-651.
4. Lee K., Park Y., Choi K. S., 2001. Re-ranking Model Based on Document Clusters. Information Processing and Management, 37 (1) : 1-14.
5. Balinski J., Danilowicz C., 2005. Reranking Method Based on Inter Document Distance. Information Processing and Management, 41 (4): 759-775.
6. Crouch C., Crouch D., Chen Q., Holtz S., 2002. Improving the Retrieval Effectiveness of Very Short Queries. Information Processing and Management, 38 (1) : 1-36.
7. Xu J, Croft W. B., 2002. Improving the Effectiveness of Information Retrieval with Local Context Analysis. ACM Transactions on Information Systems, 18 (1) : 79-112.
8. Qu, Y. L., Xu, G. W., and Wang, J. 2000. Rerank method based on individual thesaurus. In Proceedings of the NTCIR Workshop 2, pp 151 – 161.
9. Bear J, Israel D., Petit J, Martin D., 1997. Using Information Extraction to Improve Document Retrieval. In Proceedings of the Sixth Text Retrieval Conference (TREC-6), pp 376-378.
10. Kamps J., 2004. Improving Retrieval Effectiveness by Reranking Documents Based on Controlled Vocabulary. Advances in Information Reranking Documents Based in Conference on IR Research (ECIR 2004), Volume 2997 of Lecture Notes in Computer Science, pp 283-295.
11. Yang L. P., Ji D. H., Zhou G. D., Nie Y., 2005. Improving Retrieval Effectiveness by Using Key Terms in Top Retrieved Documents. In Proceedings of the 27th European Conference on Information Retrieval, pp 169-184.