ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

Vol. 4, Issue 10, October 2016

# A Survey on Graph-Based Dynamic Approach to Effectively Rank Documents based Upon Query Term Constraints

Ammar Poonawala[1], Harsh Pereira[2], ChetanVatharkar[3], Advait Trivedi[4], Shraddha Khonde[5]

Student, Dept. of Computer Engineering, Modern Education Society's College of Engineering, Savitribai Phule Pune University, Pune, Maharashtra, India[1,2,3,4]

Assistant Professor, Dept. of Computer Engineering, Modern Education Society's College of Engineering, Savitribai Phule Pune University, Pune, Maharashtra, India[5]

**ABSTRACT:** Document Management System has generated great interest in the business world. A well designed document management system entails effective and efficient retrieval and processing of stored data. The main factor of interest in most IR systems is how to effectively rank the retrieved results. In this paper we have proposed a unique dynamic, graph based approach to effectively score and rank the relevant documents.

**KEYWORDS**: Ranking algorithm, Jaccard's similarity, Support count, Dynamic programming.

## I. Introduction

Information Retrieval is the branch of data-mining which focuses on the text-search in a collection of huge set of document-store. The information retrieval process begins when a user enters a search query. User query is matched in the database and documents are ranked accordingly.

A document basically contains a rich source of information. As the amount of documents required for large-scale and small-scale enterprise business is huge, so we need an efficient and accurate Document Management System.

Document Indexing is a process of tagging information with a file so it can be used for search and retrieval purposes. The relevant documents returned by the IR System have to be ranked in a efficient and effective way, and in this paper we propose a method to do so.

We basically store document sets relating to each individual term present in the query, as Intermediate Results, that can be used by future queries, rather than computing the same every time(Building upon the concept of Dynamic Programming).

Also we add a support count to each term in the query to help eliminate certain documents not having sufficient weight age. The support count is calculated based upon the user historical data.

The end result of our algorithm is a weighted graph showing how the query is associated with relevant documents.

## II. LITERATUR REVIEW

Presently, information retrieval can be accomplished simply and rapidly with the use of search engines. This allows users to specify search criteria to obtain the results. A similarity measurement between keywords and index terms is essentially performed to facilitate users in accessing the required results.[2]. The Jaccard Coefficient is a similarity measurement used to compare similarity between sets of data.

Earlier research shows Term frequencies and inverse document frequencies have been successfully applied in determining weighting for document rankings. Rather than using Boolean retrieval where the presence of terms in documents needs to be recorded in the index, a term can also be assigned a weight that expresses its importance for a particular document.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

Vol. 4, Issue 10, October 2016

## III. PROPOSED ALGORITHM

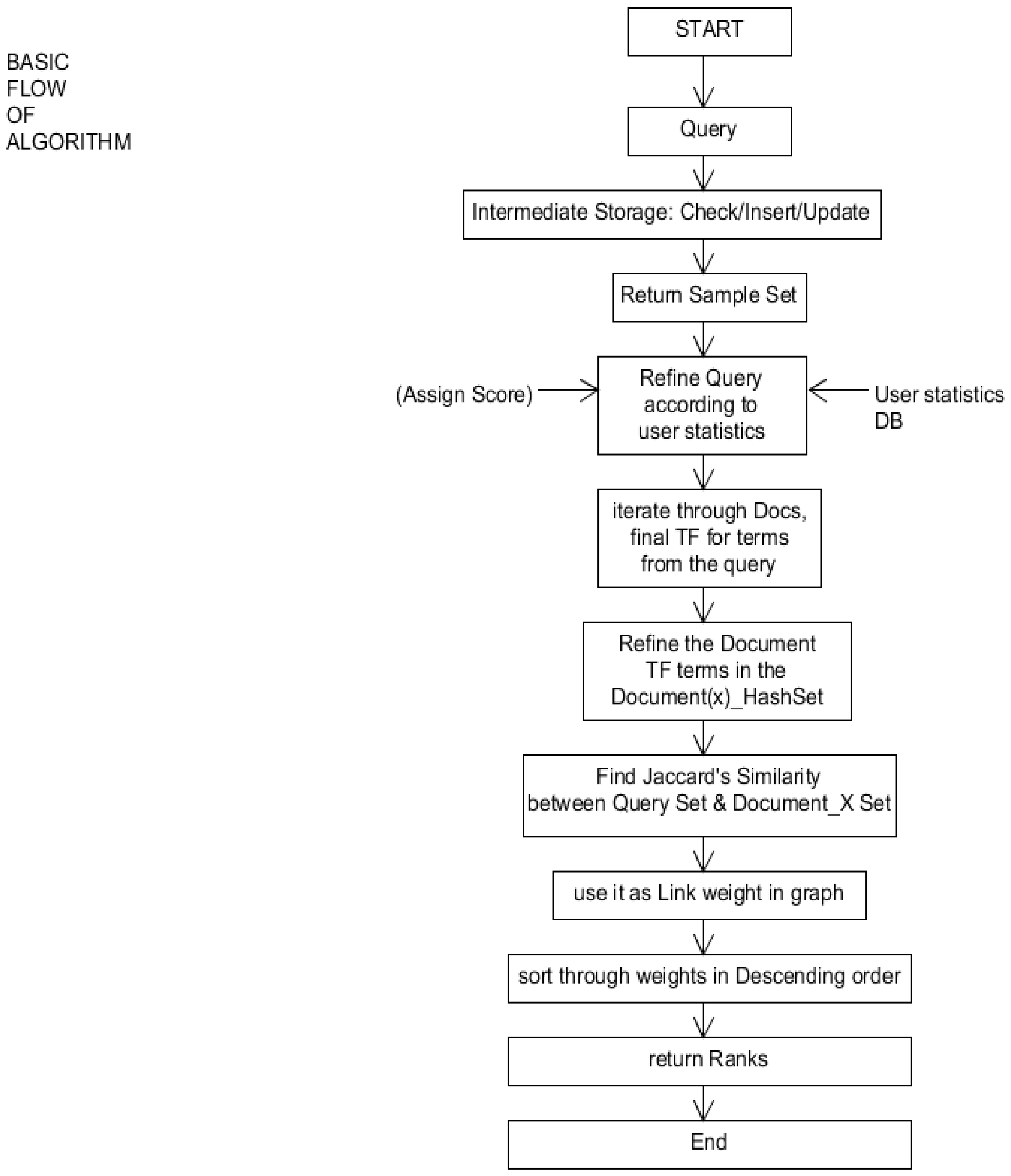

BASIC FLOW OF ALGORITHM
START
Query
Intermediate Storage: Check/Insert/Update
Return Sample Set
(Assign Score)
Refine Query according to user statistics
User statistics DB
iterate through Docs, final TF for terms from the query
Refine the Document TF terms in the Document(x)_HashSet
Find Jaccard's Similarity between Query Set & Document_X Set
use it as Link weight in graph
sort through weights in Descending order
return Ranks
End

Fig.1. Basic Flow of Algorithm

The above diagram explains the basic flow of the algorithm. How it works and how we use the intermediate storage for reducing the time required, and subsequently refining the documents based upon the user query. More in depth explanation of the algorithm is given later in the paper.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

Vol. 4, Issue 10, October 2016

A. ***TERM CONSIDERATION:***

The considered or in question are extracted from the user query as the input from the Front End or the input console.[5].

B. ***SUPPORT COUNT:***

The support Count is a percentage degree of normalization on a number in the range [1,0] that is used as a magnitude that denotes the weight of the term considered in the Query Set & is used by the algorithm for considering or discarding terms based on their term frequency in the tuples for Document Sets.

C. ***TERM FREQUENCY:***

Suppose we have a term “Cancer” & we wish to determine it's term frequency with respect to a document. We might count the number of times “Cancer” occurs in the document, this is called the Raw term frequency. For simplicity we use the Raw frequency for a better understanding and functioning of the Algorithm.

D. ***CREATING TUPLES AND NODES:***

Tuples are the basic structure used in a node in the Graph to Denote a Document Entity.
There are 2 basic nodes i.e. The Query node & the Document node.
The Query & the Documents are denoted as a set with Key:Value pairs.
More accurately, theHashSet Data structure should be used.
Where, Keys are the terms that are considered & the Values are the support count.
For Example,
Query={term1:support,term2:support,...,termN:support}
Document_X={term1:TermFrequency,term2:TermFrequency,...,termN:TermFrequency}

The node contains the above defined tuple as a constituent of itself. The reason of using a node and a Graph Structure is that we would be able to direct one node to another node and the link weight age to denote weight properties like The similarity index as described later.

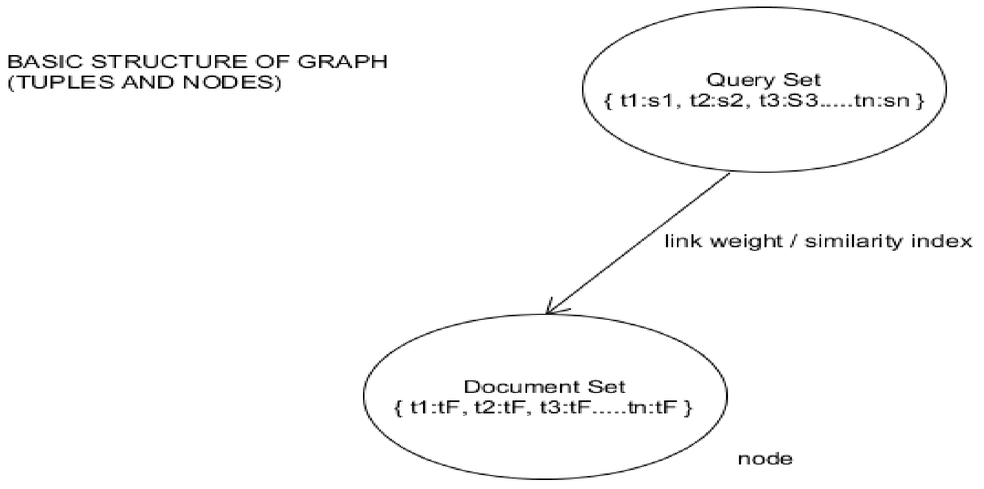

BASIC STRUCTURE OF GRAPH
(TUPLES AND NODES)
Query Set
{ t1:s1, t2:s2, t3:S3.....tn:sn }
link weight / similarity index
Document Set
{ t1:tF, t2:tF, t3:tF.....tn:tF }
node

Fig.2. A Basic Structure of Graph (Tuples and Nodes)

The above diagram shows the structure of each node and the link between two nodes. Each node is an inherent representation of the documents in the form of a set. The links denotes the similarity between two nodes (ideally the query node and the document node as the weight).

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

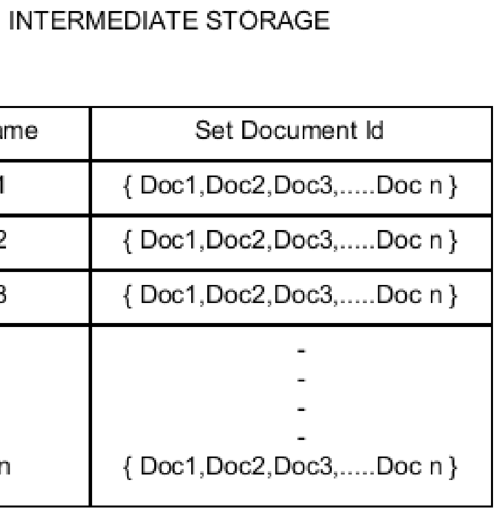
International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 10, October 2016

E. ***THE INTERMEDIATE STORAGE:***

The Intermediate storage plays an important role for reduction in time Complexity & running time of the Algorithm. This may or may not be included depending on the need of the algorithm but is highly recommended for high amounts of data based upon the application.[1].
The Intermediate Storage is basically a table containing indexed documents with respect to previously used frequent terms in the user queries & User Statistics.
For example,



INTERMEDIATE STORAGE

| Term Name | Set Document Id |
|---|---|
| Term1 | { Doc1,Doc2,Doc3,.....Doc n } |
| Term2 | { Doc1,Doc2,Doc3,.....Doc n } |
| Term3 | { Doc1,Doc2,Doc3,.....Doc n } |
| - - - - Term n | - - - - { Doc1,Doc2,Doc3,.....Doc n } |

Fig.3. Intermediate Storage

So as we see above the Intermediate storage stores sets of Document Indexes that can be directly called and used as the sample space set by the similarity Iterations ,thus reducing the size of the sample set & in true reducing the total number of iterations that the Algorithm performs to create the Rank Graph. This reduces a humongous amount of calculation load on the processing side & the total amount of time taken to produce the results.

The Intermediate Storage can also find the intersections of singular terms thus increasing the reach of the storage in case the user query extends over the intersection of any 2 given terms. The Storage can again return the sample space containing more relevant documents thus reducing the total number of calculations and iterations, yet again

F. ***QUERY REFINING USING USER STATISTICS:***

The User query basically is an unrefined set of terms. Now we need to filter out irrelevant terms from any respective document that is related to the query and in turn also reduce the similarity of the document with the query.

The Support count is used in the refining procedure, thus imparting a magnitude to each term. And if the Term Frequency of the term under consideration in the respective document is under the assigned magnitude then the term is discarded from the document tuple. Thus making the document less similar to the fed query.
For Example,

Query={Term1,Term2,...,termN} => refining refining function =>Query_refined={term1:support,term2:support,...,termN:support}
The User statistics is generally a database containing the historic user terms

G. ***THE JACCAD SIMILARITY BETWEEN DOCUMENTS:***

We use the Jaccard Similarity Index to find the similarity between the document and the given query and the document that is under check.[3],[4].

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

**Vol. 4, Issue 10, October 2016**

Both the Query and the Document are represented as a hash-set.
The Jaccard's index is calculated as follows,

J(Query_refined,Document_X)= n(Query_refined.intersection(Document_X)) /
n(Query_refined.union(Document_X))

where, Query={term1:support,term2:support,...,termN:support}
Document_X={term1:TermFrequency,term2:TermFrequency,...,termN:TermFrequency}

This similarity is then used as the link weight age between the respective document node & the Query node. Later this index weight is used as a measure for the ranking

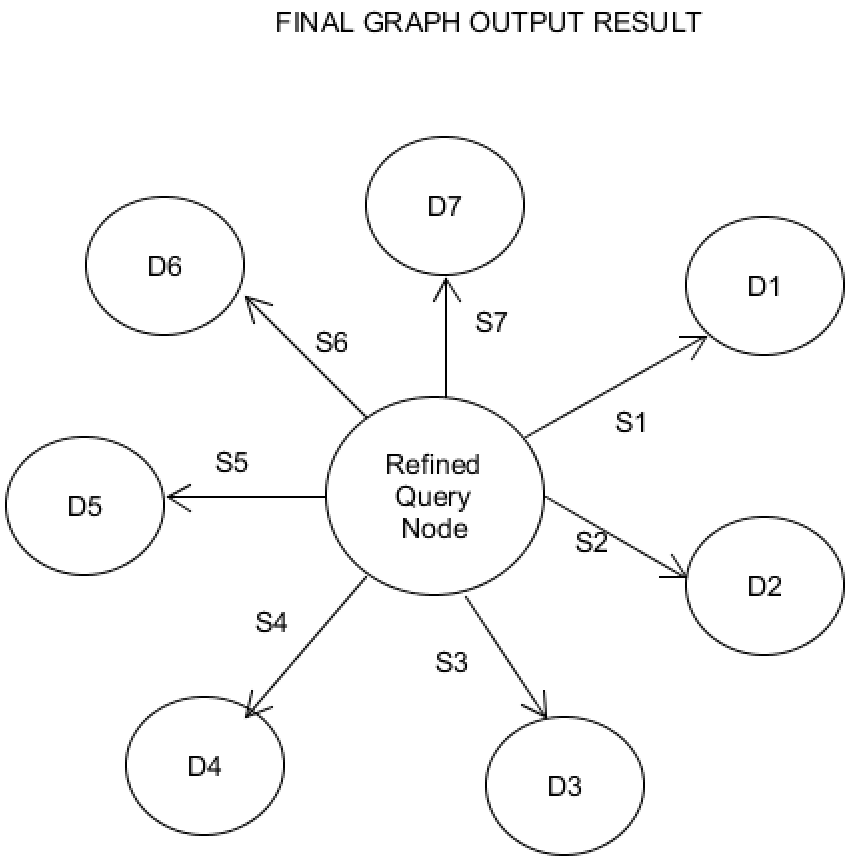


Fig.4. Final Graph

The above image shows the output graph, which is the resultant graph of the algorithm. This way the algorithm displays the end result as a graph with the query node at the centre and each document linked to it with the weights on the links.

## IV. CONCLUSION

In this paper, we describe a graph-based algorithm for ranking documents based on the user's query. Jaccard's Similarity is used for finding the frequency of terms in a particular document. We are developing a prototype system based on our method. Further we will try to improve our system by efficient storage for graph approach.

## V. FUTURE WORK

To increase the speed of the document retrieval we propose a modification in the Intermediate Stage, Currently the Intermediate Storage is only being updated when a query is issued to the IR System, But rather we can periodically update it. We propose to do so in the following way
We consider the various records already present, and create new combinations of these records. We precede so in such a way till no new unique records can be generated.
Thus the chance that the set of documents we are interested in being present in the Intermediate System greatly increases, and the document processing is done at a much faster rate.

## REFERENCES

1. Sandip R. Pandit, M. A. Potey 'A Query Specific Graph Based Approach to Multi-document Text Summarization: Simultaneous Cluster and Sentence Ranking', IEEE,ICMIRA, Issue no.978-0-7695-5013-8/13, 2013
2. Jingyong Wan, Beizhan Wang, Wei Guo, Kang Chen, Jiajun Wang 'A Distributed Search Engine Based on a Re-ranking Algorithm Model', IEEE, Issue no. 978-1-4799-6600-4/15, 2015
3. SuphakitNiwattanakul, JatsadaSingthongchai, EkkachaiNaenudorn and SupachanunWanapu 'Using of Jaccard Coefficient for Keywords Similarity' , Proceedings of the International MultiConference of Engineers and Computer Scientists 2013,Vol I, IMECS 2013, March 13 - 15, 2013, Hong Kong
4. VikasThada, DrVivekJaglan 'Comparison of Jaccard, Dice, Cosine Similarity Coefficient To Find Best Fitness Value for Web Retrieved Documents Using Genetic Algorithm' , International Journal of Innovations in Engineering and Technology (IJIET) ,Vol. 2, Year 2013
5. S. Florence Vijila and Dr. K. Nirmala'Quantification of Portrayal Concepts using tf-idfWeighting', International Journal of Information Sciences and Techniques (IJIST) , Vol.3, No.5, September 2013
6. Yadong Zhu, YanyanLan, JiafengGuo, Pan Du, Xueqi Cheng, 'A Novel Relational Learning-to-Rank Approach for Topic-Focused Multi-DocumentSummarization', IEEE 13th International Conference on Data Mining,No.1550-4786, 2013
7. Sengolmary J, Usha S, 'Web Based Document Management System in Life Science Organization',Online International Conference on Green Engineering and Technologies (IC-GET 2015), No. 978-1-4673-9781-0/15, 2015
8. RajniJindal,ShwetaTaneja, 'Ranking in Multi Label Classification of Text Documents Using Quantifiers', IEEE International Conference on Control System, Computing and Engineering, No.162-166, 2015
9. AzamFeyznia, Mohsen Kahani, Reza Ramezani, 'A Link Analysis Based Ranking Algorithm for Semantic Web Documents', 6th Conference on Information and Knowledge Technology (lKT 2014),Pg. no. 123-127, 2014
10. Liana Ermakova, JosianeMothe, 'Document Re-ranking Based on Topic-Comment Structure',IEEE,ICMIRA,No. 978-1-4799-8710-8/16, 2016
11. HediyehBaban,SalimahMokhtar, 'Online Document Management System for Academic Institutes', 3rd International Conference on Information Management, Innovation Management and Industrial Engineering, No. 315-319, 2010
12. ShyamaleshKhan,B V N Prasad,SSelvi,Usha Rani, 'Document Management System: An Explicit Knowledge Management System',2nd International Conference on Computing for Sustainable Global Development, No.402, 2015