An Algorithm for Hyper-cube-based Ontology Construction

Debajyoti Mukhopadhyay, Sounak Chakravorty, Sudarshan Nandy

Web Intelligence & Distributed Computing Research Lab, Techno India Group
West Bengal University of Technology
EM 4/1, Salt Lake Sector V, Calcutta 700091, India
{debajyoti.mukhopadhyay, sounak.bhabook, sudarshannandy}@gmail.com

Abstract

The Web has grown in leaps and bounce in the last decade. The size of the Web has become so large that it has become very difficult to find out the necessary information from the Web. Search engines help us in our quest of finding the required information on the Web. These search engines over the years have also developed themselves in a grand manner. With the advent of ontology based search engine, has only helped the cause. Ontology based search engines are transforming the whole idea of searching into a new league of its own. In this paper we propose a novel approach for proper construction and searching ontology. This proposition of ours will help in the development of a very well organized ontology which in tern will help us in searching the Web in a very useful and organized manner. Thus reducing the unwanted search results.

1.Introduction

With the advent of the Web the exchange of information became a Childs play. As it was very easy to send or receive information from the Web. Thus the Web grew over the years and number of Websites increased day by day. Soon the numbers of Website became so large that the need for some sort of searching technique was felt. Thus the concept of search engine came into being. Soon these search engines became very popular and more and more people used these search engines to get a proper result for their own information requirement. With the need for better searching results increased, so increased the need for development of better search engines. Advent of new techniques for the development of search engine ushered in the times to come. Ontology is one of the techniques to make the Web searching exercise well organized and very much user friendly.

1.1 Search Engine

The concept of search engine almost single handedly has changed the concept of Web searching rather the process of finding any information or user required documents on the net. Search engine is software that helps the net surfer to find out relevant and important information's on the Internet rather the World Wide Web (www).This concept has made the Web very much user friendly and searching information a very matter. Not only searching the process of data sharing has also evolved along with the concept of searching. Though few term it as the two faces of the same coin as searching means getting information which is sharable. The concept of sharing of resources on the Web started from Circa(1993).This was followed by the advent of FTP, Gopher, Archie which became popular for sharing Web resources on the Internet. But these sharing were restricted to file level. After Circa (1994) HTML's and Uri's came into picture and we moved onto text level. But nobody wanted to be restricted at the file or text level but break the barriers and reach the data level. Thus evolved the concept of semantic Web.

2. Semantic Web

The semantic Web works at the data level. Semantic Web documents use metadata to express the meaning of the content encapsulated within them. RDF/XML has been widely recognized as the standard vehicle for describing metadata, an

0-7695-3068-0/07 $25.00 © 2007 IEEE
DOI 10.1109/ICIT.2007.23

enormous amount of semantic data is still being encoded in HTML documents that are designed primarily for user friendliness. Moving on with the concept of the semantic Web the concept of Ontology came into being.

3. The Problem

As ontology was developed using the hierarchical design, it was not able to provide the proper instruction to the crawler in some specific cases. Even though all the information were present in the ontology itself. This was happening due to the lack of interaction between the keywords present in the ontology. Thus the need was felt to design and construct ontology in such a manner such that all keywords remain connected who has similar links between them. Not only this, to make the retrieval faster the need was felt that all the keywords remain connected with the domain under which the ontology is present. This urge led to the concept of constructing hyper-cube ontology with a peer-to-peer connection between the nodes or keywords of the ontology.

4. Algorithm for Hyper-Cube Ontology Construction & Keyword Deletion

The following algorithm creates a hyper-cube with KEYWORDs, which are present in the Ontology. EMPTY specifies that there is no KEYWORD available in the Ontology. ACTIVE means that the KEYWORD is very much alive and can be worked with. NEW is that KEYWORD which has just arrived and wants to get associated with the hyper-cube. LINK denotes the linkage between the KEYWORDs in the Ontology by virtue of similar attributes or properties. MODIFY means that the structure has to be integrated to a higher dimension from the existing one. COPY denotes the replica of already present KEYWORD, which is to be inserted into the structure to make it more stable and have well defined LINK. To start the construction we have to first INITIALIZE a KEYWORD.

Input: *KEYWORD.*

Output: *Linked keywords or Hyper-Cube Structure.*

Begin

Step 1: IF
KEYWORD=0

Step 2: THEN
Return EMPTY

Step 3: IF
KEYWORD=1 and
KEYWORD=ACTIVE

Step 4: THEN
INITIALIZE KEYWORD=k0 or initial node of the hyper-cube.

Step 5: IF
KEYWORD=NEW and k0 is already present.

Step 6: THEN
LINK NEW to k0
Set LINK=0
Set NEW=k1

Step 7: KEYWORD=NEW and k0, k1 already present.

a. {
LINK NEW to k1
// assume NEW connects k1. If NEW connects k0 only the labels will change but the process will be same.
Set LINK=1
Set NEW=k2
LINK k2, k0 as {0, 1}
}

b. MODIFY organization of KEYWORDS from triangle to square.
{
COPY k0
LINK k0, k0=1
LINK k0, k2=0
}

Step 8: KEYWORD=NEW and k0, k1 and k2 already present.

{
Replace COPY k0=NEW
NEW=k3
LINK k0, k3=1
LINK k3, k2=0
}

Step 9: KEYWORD=NEW and k0, k1, k2 and k3 already present

a. {
LINK NEW to k0
// Assume NEW connects k0. If NEW connect others, the labels will change but the process will be same.
Set LINK=2
Set NEW=k4
}

b. MODIFY organization of KEYWORDs from Square to cube.
{

COPY k1, k2, k3
LINK k0, k1=0
LINK k0, k3=1
LINK k1, k2=1
LINK k3, k2=0
LINK k2, k2=2
LINK k3, k3=2
LINK k1, k1=2
LINK k3, k4=1
LINK k1, k4=0
}

Step 10: KEYWORD=NEW and k0, k1, k2, k3 and k4 already present
{
Replace COPY k1=NEW
NEW=k5
LINK k4, k5=0
LINK COPY k2, k5=1
LINK k1, k5=2
}

Step 11: IF k0 is deleted. // To show deletion of a KEYWORD we have chosen k0 as the deleted node.

Step 12: THEN
{
COPY k4 in the place of k0
LINKs remain same as step only replace ko by COPY k4.
}

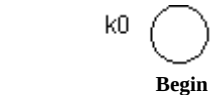
Step 13: KEYWORD=NEW and k1, k2, k3, k4 and k5 already present.
{
Replace COPY k1=NEW
NEW=k6
LINK COPY k2, k6=0
LINK k4, k6=1
LINK k3, k6=2
}

Step 14: KEYWORD=NEW and k1, k2, k3, k4, k5 and k6 already present.
{
Replace COPY k2=NEW
NEW=k7
LINK k6, k7=0
LINK k5, k7=1
LINK k2, k7=2
}

Step 15: KEYWORD=NEW and k1, k2, k3, k4, k5, k6 and k7 already present.
{
Replace COPY k4=NEW
NEW=k8
LINK k1, k8=0
LINK k3, k8=1
LINK k4, k8=2
}

End

4.1 Diagrams Illustrating the Algorithm



Begin

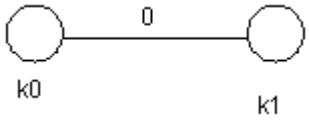


Figure 1

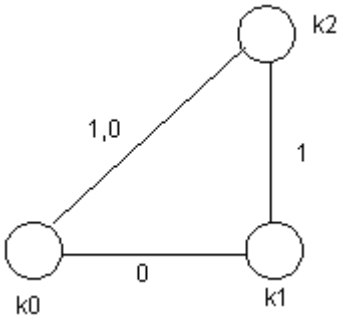


Figure 2

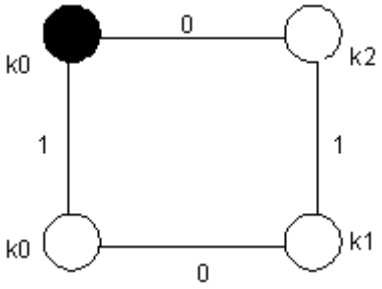


Figure 3

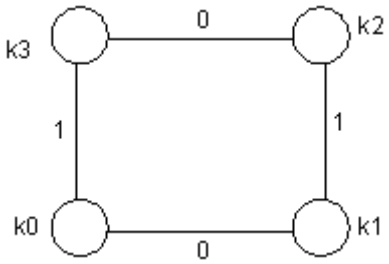


Figure 4

Here the diagrams (Begin, Fig.1, Fig.2, Fig.3, and Fig.4) show the cases for the Begin, step1-step 8 of the algorithm stated above. For all the diagrams the *keyword*s are denoted by empty circles (white) and there copies as filled circles (black) and numbers marked on the lines are the neighbor relationship or the *link* between the two *keyword*s.

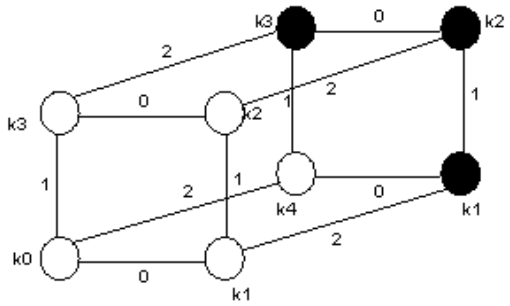


Figure 5

Here the diagram (Fig.5) illustrates the step9 of the algorithm. The black circles are the copies of the original, developed to maintain the cohesiveness of the structure.

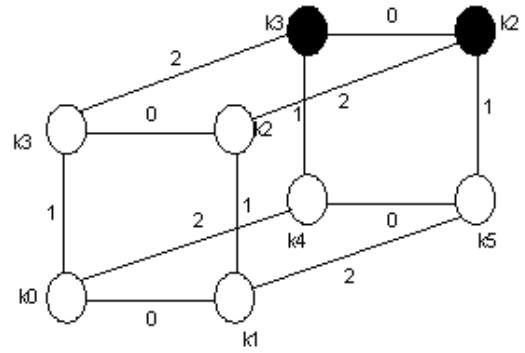


Figure 6

Here the diagram (Fig.6) illustrates the step10 of the algorithm.

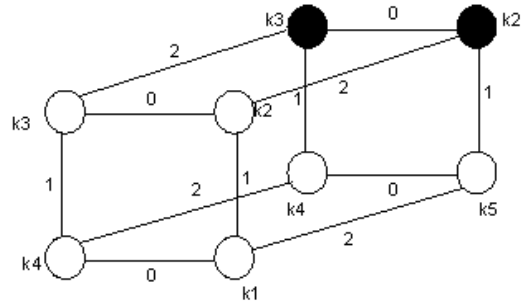


Figure 7

Here the diagram (Fig.7) illustrates the step11 and step12 of the algorithm. This is also the illustration which shows the deletion of keywords from the cube.

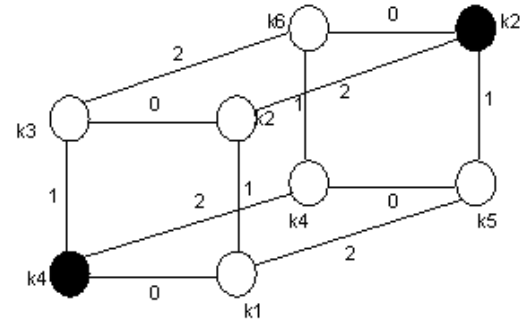


Figure 8

Here the diagram (Fig.8) illustrates the step13 of the algorithm.

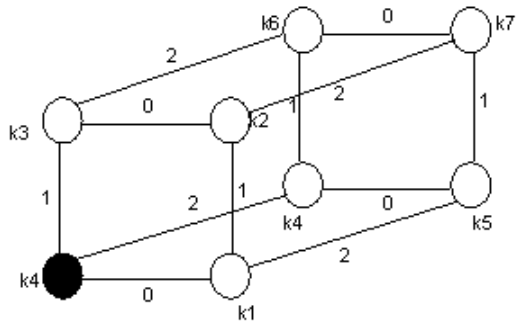


Figure 9

Here the diagram (Fig.9) illustrates the step14 of the algorithm.

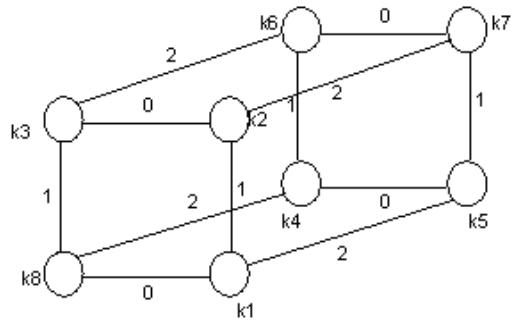


Figure 10

Here the diagram (Fig.10) illustrates the step15 of the algorithm and the final position of the cube when all the eight keywords have been included. The Hyper-cube structure.

4.2 The Final Structure

As we arrive to the hyper-cube structure of the ontology we get all the keywords of the ontology arranged in a cubical structure and the domain have links to all the keywords. Thus whenever the user selects the domain after specifying the keyword the search word will be broadcasted to all the keywords present in the ontology. Below is given the diagram which illustrates the structure of the ontology and the connection between all the keywords present in the ontology.

Here it can be seen that the user denoted by the blue rectangle when specifies the domain (d1, d2, d3, d4 and d5) he gets access to the ontology prepared under that certain domain. The query gets broadcasted to all the nodes of the ontology. Wherever the specified information is matched, the URL attached to the specified information is sent to the respective crawler for the downloading of that specific page.

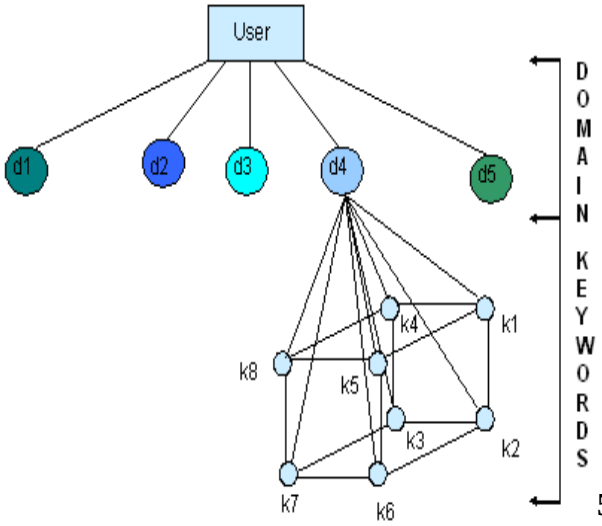


Figure 11 The final structure of the hyper-cube ontology

Here we have shown the structure considering eight nodes or keywords. But the number may be more than this. Thus to increase the number of keywords in the ontology we will follow the same algorithm which we have designed for the hyper-cube ontology.

Again the second level of keywords will also be managed by the same algorithm mentioned above. Thus it can be said that by calling the hyper-cube algorithm recursively we will get all the keywords properly structured under certain ontology.

4.3 The Searching

As the structure has been fully developed the next thing which comes along is the searching. As the structure of the ontology is very well organized so the searching becomes a very easy but effective and pinpoint procedure. The searching should be carried out in the following steps. They are as follows:

Step1: Get the users domain specification.
Step2: Search for the words present in the user query in the domain specified by the user.
Step3: Broadcast the words to all the keywords of the ontology.
Step4: Return the URL attached to the keywords which matched with the words of the user query.

After following these steps we will easily get the required search result from our crawler.

4.4 Flowchart

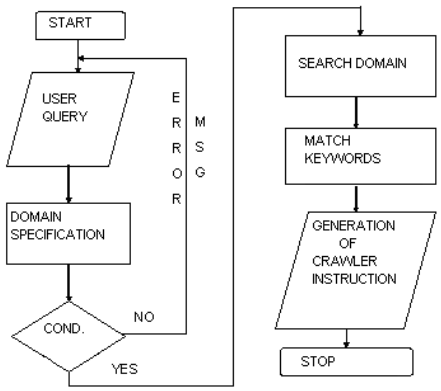


Figure 12 Flowchart on the working procedure

5. Result

Here we see the performance of our searching in comparison with other established search engines. To show the performance of our search engine we carried out numerous searches and got very encouraging results. Among these highly appreciable results we will be providing one of them and will also provide the comparison graph. Here we have specified the *utility* of the URL which is to be downloaded in comparison with the requirement of the user query. We have specified *utility* in three *categories* and set values for each of them and plotted graph with those values.

5.1 Utility

a. Category [1.0]: Matched perfectly.
b. Category [0.5]: Matched up to some extent.
c. Category [0.0]: Not Matched.
Search Topic/Query: Sophocles Script

Table 1. GOOGLE Search

NO	URL	UTILITIY
1	www.**sophocles**.net	0.0
2	www.**sophocles**.net/links.asp	0.0
3	writingforstagescreen.suite101.com/article.cfm/**sophocles_script_**software	0.0
4	classics.mit.edu/**Sophocles**/antigone.html	1.0
5	www.**script**frenzy.org/eng/node/401417	0.5

Table 2. Hyper Cube Based Ontology Searching:

NO	URL	UTILITY
1	http://classics.mit.edu/Sophocles/ajax.html	1.0
2	http://classics.mit.edu/Sophocles/antigone.pl.txt	1.0
3	http://classics.mit.edu/Aristophanes/acharnians.pl.txt	1.0
4	http://classics.mit.edu/Sophocles/electra.pl.txt	1.0
5	http://www.theatrelinks.com/playscripts.htm	1.0

Table 3. YAHOO Search:

NO	URL	UTILITY
1	www.**sophocles**.net	0.0
2	www.**sophocles**.net/download.asp	0.0
3	writingforstagescreen.suite101.com/**...**/**sophocles_script**_software	0.0
4	classics.mit.edu/**Sophocles**/electra.htm	1.0
5	www.writerdirector.com/articles/**sophocles**.htm	0.0

5.2 Comparison Graph

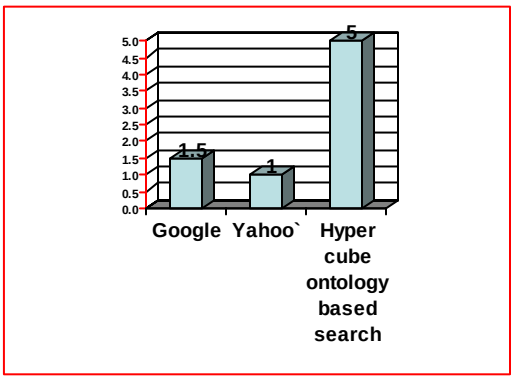


Fig. 13 Comparison Graph among Google, Yahoo and HyperCube based ontology searching

The above graph shows the comparison of the search results. Here we can see as far as utility is concerned our searching technique is far ahead of the other established search engines. Our search engine gives the most specific result.

6. Conclusion

Though our searching technique is working perfectly and also producing very good and specific results but still it can't be termed as an intelligent search engine, as queries like *"a bank on the river bank"* cannot be handled properly by our technique of our searching. In our coming works or propositions we will give a solution for this sort of problems.

References

[1] DAML Services Coalition.DAML-S: Web Service Description for the Semantic Web. In *The First International Semantic Web Conference (ISWC)*, June 2002.

[2] Les Carr, Wendy Hall, Sean Bechhofer, and Carole A. Goble. Conceptual linking: ontology-based open hypermedia. In *World Wide Web*, pages 334–342, 2001.

[3] Stefano Ceri, Piero Fraternali, and Aldo Bongio. Web Modeling Language (WebML): a modeling language for designing Web sites. *Computer Networks (Amsterdam, Netherlands: 1999)*, 33(1–6):137–157, 2000.

[4] S. Ratnasamy, S. Shenker, I. Stoica. Routing Algorithms for DHTs: Some Open Questions. In *Proc. of 1st International Workshop on P2P Systems*, March 2002.

[5] I. Stoica et al. Chord: A scalable P2P lookup service for Internet applications. In *Proc. of ACM SIGCOMM*, August 2001.

[6] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic Web. *Scientific American* May 2001.

[7] M. Uschold and M. Grüninger. Ontologies: Principles, Methods and Applications. In *Knowledge Engineering Review, 11(2)*, 1996.

[8]. Debajyoti Mukhopadhyay, Debasis Giri, Sanasam Ranbir Singh; *An Approach to Confidence Based Page Ranking for User Oriented Web Search;* ACM SIGMOD Record, ACM Press, New York, USA; Vol. 32, No. 2, June 2003.

[9]. Debajyoti Mukhopadhyay, Debasis Giri, Sanasam Ranbir Singh; *A Confidence Based Methodology to Deduce User Oriented Page Ranking in Searching the Web;* ITPC-2003.