Algorithm for Merging Search Interfaces over Hidden Web

Harish Saini[1], Kirti Nagpal[2]
[1]Assistant Professor N.C.College of Engineering Israna, Panipat
[2]Research Scholar, N.C.College of Engineering Israna, Panipat

Abstract: This is the world of information. The size of world wide web [4,5] is growing at an exponential rate day by day. The information on the web is accessed through search engine. These search engines [8] uses web crawlers to prepare the repository and update that index at an regular interval. These web crawlers [3, 6] are the heart of search engines. Web crawlers continuously keep on crawling the web and find any new web pages that have been added to the web, pages that have been removed from the web and reflect all these changes in the repository of the search engine so that the search engines produce the most up to date results.

Keywords: Hidden Web, Web Crawler, Ranking, Merging

I. INTRODUCTION

This is the world of information. The size of world wide web [4,5] is growing at an exponential rate day by day. The information on the web is accessed through search engine. These search engines [8] uses web crawlers to prepare the repository and update that index at an regular interval. Web crawlers continuously keep on crawling the web and find any new web pages that have been added to the web, pages that have been removed from the web and reflect all these changes in the repository of the search engine so that the search engines produce the most up to date results. There is some data on the web that is hidden behind some query interfaces and that data can't be accessed by these search engines. These kind of data is called hidden web [41] or deep web. These days the contents on the hidden web are of higher interests to the users because the contents of hidden web are of higher quality and greater relevance. The contents on hidden web can be accessed by filling some search forms which are also called sometimes search interfaces. Search interfaces [37] are the entry points to the hidden web and make it possible to access the contents on the hidden web. These search interfaces are simply like filling any forms like forms for creating e-mail ids etc. Each search interface has number of input controls like text boxes, selection lists, radio buttons etc.. To access hidden web one needs to fill these search interfaces for which a crawler is required which finds for forms on the web known as form crawler. There can be multiple search interfaces for the same information domain on the web. In that case all those search interfaces are required to be merged or integrated so that crawler[3] finds all data relevant to the user input despite of existence of multiple search interfaces for the same domain. Hidden web crawler is the crawler which continuously crawls hidden web so that it could be indexed by search engines. The major functions which are to be performed by hidden web crawler are getting form filled, finding search query, submitting query and indexing the results. The challenge of the hidden web crawlers are the huge and dynamic nature of the hidden web. The merging[46] or integration of search interfaces over hidden web is required to response queries so that it answers queries at its best. For the integration of search interfaces over hidden web two steps are required

- Finding semantic mapping over the different search interfaces to be merged.
- Merging search interfaces on the basis of semantic mappings found earlier.

A *Ranking the Web Pages by Search Engine*

Search for anything using our favorite crawler-based search engine will sort through the millions of pages it knows about and present you with ones that match your topic. The matches will even be ranked, so that the most relevant ones come first. Of course, the search engines don't always get it right. So, how do crawler-based search engines go about determining relevancy, when confronted with hundreds of millions of web pages to sort through? They follow a set of rules, known as an algorithm. Exactly how a particular search engine's algorithm works is a closely kept trade secret.

One of the main rules in a ranking algorithm involves the location and frequency of keywords on a web page. Call it the location/frequency method. Pages with the search terms appearing in the HTML title tag are often assumed to be more relevant than others to the topic. Search engines will also check to see if the search keywords appear near the top of a web page, such as in the headline or in the first few paragraphs of text. They assume that any page relevant to the topic will mention those words right from the beginning. Frequency is the other major factor in how search engines determine relevancy. A search engine will analyze how often keywords appear in relation to other words in a web page. Those with a higher frequency are often deemed more relevant than other web pages.

B. ***Web Crawler***

A crawler [14,20] is a program that are typically programmed to visit sites that have been submitted by their owners as new or updated. The contents of the Entire sites or specific pages is selectively visited and indexed. The key reason of using web crawlers is to visit web pages and add them to the repository so that a database can be prepared which in turn serves applications like search engines.

Crawlers use graph structure of the web to move from pages to pages. The simplest architecture of web crawler is to start from a seed web page and traverse all hyperlinks encountered in this web pages then all encountered hyperlinks are added to the queue which in turn are traversed. The process is repeated until a sufficient number of pages are identified...

The main goal of the web crawler is to keep the coverage and freshness of the search engine index as high as possible which is not informed by user interaction. For this task the crawler and other parts of the search engine have no communication between them. Because web is a purely dynamic collection of web pages there is a need for crawling cycles frequently to refresh the web repository so that all new pages that have been added to the web are included in the repository similarly the web pages that have been removed from web are deleted from web repository

C ***Hidden Web Crawler***

Current day crawlers crawls only publicly indexable web (PIW) i.e set of pages which are accessible by following hyperlinks ignoring search pages and forms which require authorization or prior registration. In reality they may ignore huge amount of high quality data which is hidden behind search forms. Pages in hidden web are dynamically generated in response to the queries submitted via search forms. Crawling the hidden web is highly challenging task because of scale and the need for crawlers to handle search interfaces designed primarily for human beings. The other challenges of hidden web data are:

- Ordinary web crawlers can't be used for hidden web
- The data in hidden web can be accessed only through a search interface
- Usually the underlying structure of the database is unknown.

The size of hidden web is continuously increasing as more and more organizations are putting their high quality data online hidden behind search forms. Because there are no static links to hidden web pages therefore search engines can't discover and index these web pages.

Because the only entry point to hidden web is search interface the main challenge is how to generate meaningful queries to issue to the site. Hidden web crawler is the one which automatically crawls hidden web so that it can be indexed by search engines. Hidden web crawler is able to allow an average use to explore the amount of information, which is mostly hidden behind search interfaces. The other motive for hidden web crawler is to make hidden web pages searchable at a central location so that the significant amount of time and effort wasted in searching the hidden web can be reduced. One more motive for hidden web crawlers is that due to heavy reliance of web users on search engines for locating information, search engine influence how the users perceive the web..

There are two core challenges while implementing an effective hidden web crawler

- The crawler has to be able to understand and model a query interface.
- The crawler has to come up with meaningful queries to issue to the query interface.

The first of the two challenges was addressed by a method for learning search interfaces. For the latter challenge if the search interface is able to list all possible values for a query with help of some selection list or drop down list in that case solution is straight forward. possible so all of the possible queries can't be exhaustively listed.

D ***Search Interfaces***

While accessing hidden web [49] search query is submitted to the search interface of the hidden web crawler. The selection of web pages and the relevance of results produced from hidden web is determined by the effectiveness of search interfaces [42,43]. The different actions performed by search interfaces are getting input from user, selection of query, submitting query to the hidden web and merging of results produced by hidden web crawlers.

The most important activity of search interface is selection of query which may be implemented in variety of ways. One method is random method in which random keywords are selected from the English dictionary and are submitted to the database. Random approach generates a reasonable number of relevant pages. The other technique is using generic frequency. In this method we find a document and obtain general distribution frequency of each word in the document. Based on this *frequency* distribution we find the most frequent keyword, issue it to the hidden web database and retrieve the result. The same process is repeated with the second most frequent word and so on till the downloaded resources are exhausted. After that it analyses the returned web pages that whether they contain results or not. Another method is adaptive in this method the documents returned from the previous queries issued to the Hidden-Web database is analysed and it is estimated which keyword is most likely to return the most documents. Based on this analysis, the process is repeated with the most promising queries.

E *Problems in Integration of Hidden Web*

Recently keen interest has been developed in the retrieval and integration of hidden web with the intention of having high quality data for the databases. As the size of information in hidden web is growing, search forms or search interfaces are needed to be located which serves as an entry point to hidden web database. But there exists crawlers which focus search on specific database domains and retrieve invariable diversified set of forms like login form, quote request forms and searchable forms from multiple domains. The process of grouping these search interfaces or search forms over the same database domain is called integration of search interfaces.

These search form are input to algorithm which finds correspondences among attributes of different forms. This task is not so easy because of dynamic nature of the web the new information is added to web and new information is being modified or even removed. Therefore a scalable solution suitable for large scale integration is required for the purpose of large scale integration. In addition in a well defined domain there may be variation in both structure and vocabulary of search forms. So in order to obtain information that covers the whole domain it is required to find correspondences among attributes of forms and for that a broad search is needed to be performed.

For this either full crawl is needed to be performed but this is really inefficient approach as it can take weeks for the exhaustive crawling process. Another alternative can be focus crawling in which only pages relevant to a specific topic are crawled. This has a better quality index as compared to exhaustive crawling. In this case focused crawler that focus solely on the contents of retrieved web pages may not be a very good alternative also because forms are sparsely distributed and thus the number of forms retrieved per total number of pages retrieved may be very low. For tackling this problem a focused crawler has been developed which used reinforcement learning to build a focus crawler that is effective for sparse concepts.

The kind of web crawler which crawl web pages which contain search forms are called form crawlers. Since form crawler find thousand of forms for the same domain. Those different forms even for the same domain may be different in the structure and vocabulary they use. For that semantic mappings are required to be found among all attributes of those forms and after finding semantic mappings those are integrated so that queries involving that domain can be answered suppressing the existence of multiple search forms using different structures and vocabulary. There exist multiple integration techniques which integrates these search forms over the hidden web. Most of those are semiautomatic.

F. Need For Merging Search Interfaces

It seems that there will always be more then one search interfaces even for the same domain. Therefore, coordination(i.e. mapping, alignment, merging) of search interfaces is a major challenge for bridging the gaps between agents with different conceptualizations.

Two approaches are possible:

(1) merging the search interfaces to create a single coherent search interface

(2) aligning the search interfaces by establishing links between them and allowing them to reuse information from one another.

Here we propose a new method for search interface merging.

Whenever multiple search forms are to be integrated the task can be divided into two major steps

- Finding semantic mapping among search forms or interfaces
- Merging search interfaces on the basis of mapping

The first step in merging [46] of search forms or interfaces are finding semantic mapping and correspondences among attributes of those forms so that on the basis of that mapping search interfaces or forms could be merged. This semantic mapping serves as the domain specific knowledge base for the process of merging [47] of search interfaces. The proposed algorithm is semi automatic. It finds similar terms from a look-up table which stores all similar terms in the different search interfaces to be merged along with the relationship between those similar terms so that it may be decided which term will be appearing in the merged search interface. The algorithm makes use of search interfaces in form of a taxonomy tree to cluster the concepts in the search interfaces on the basis of level

II Merging of Search Interfaces

More and more applications are in need to utilize multiple heterogeneous search interfaces across various domains. To facilitate such applications it is urgent to reuse and interoperate heterogeneous search interfaces. Both merge and integration produces a static search interface based on the existing search interfaces. The resulting search interface is relatively hard to evolve with the change of existing search interfaces.

The search interface merge module takes source search interfaces as input and a table that lists all similar terms coming from different search interfaces along with the relationship between those similar terms. The similar terms can have relationship like meronym, hyper name and synonym. The search interface merge process is divided into **several steps.**

Step 1. Find set of similar terms

Step 2. Children of similar terms are merged and point to the same parent.

Table 1.1 Domain specific knowledge base

Concept	Concept	Relation
Optical_drive	OpticalDisk	Synonym
HardDrive	HardDisk	Synonym
DVD	DVD-Rom	Synonym
CDRW	CD-RW	Synonym
CDRom	CD-ROM	Synonym
Computer	Personal Computer	Synonym

The taxonomy trees of search interfaces A and B respectively are as following

A *Merging Observations and Results*

Let us consider two source search interfaces and apply the above stated algorithm to merge those search interfaces as a case study. The source search interfaces under consideration are given along with the domain specific knowledge base.

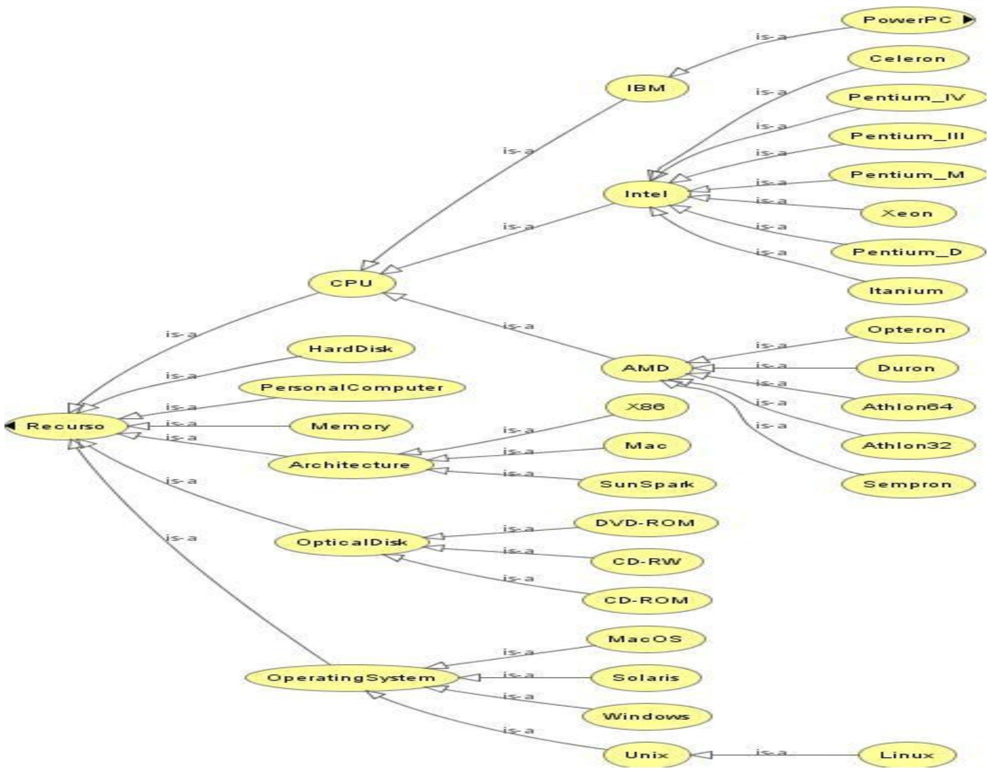

Fig1.1TaxonomytreeforSearchInterfaceS1

The codes assigned to the terms in the source search interface are as follows. Let us consider search interface a first and see the codes assigned

Table 1.2 Codes of terms in search interface A

Term	Code
Recurso	A
CPU	A01
Hard Disk	A02
Personal computer	A03
Memory	A04
Architecture	A05
Optical disk	A06
Operating System	A07
IBM	A0101
Intel	A0102

AMD	A0103
X86	A0501
MAC	A0502
Sunsparc	A0503
DVD-Rom	A0601
CD-RW	A0602
CD-Rom	A0603
Macos	A0701
Solaris	A0702
Windows	A0703
Unix	A0704
PowerPC	A010101
Celeron	A010201
PIV	A010202
PIII	A010203
PM	A010204
Xeon	A010205
Pentium_D	A010206
Itanium	A010207
Opetron	A010301
Duron	A010302
Athlen 64	A010303
Athlen 32	A010304
Semaron	A010305
Linux	A070401

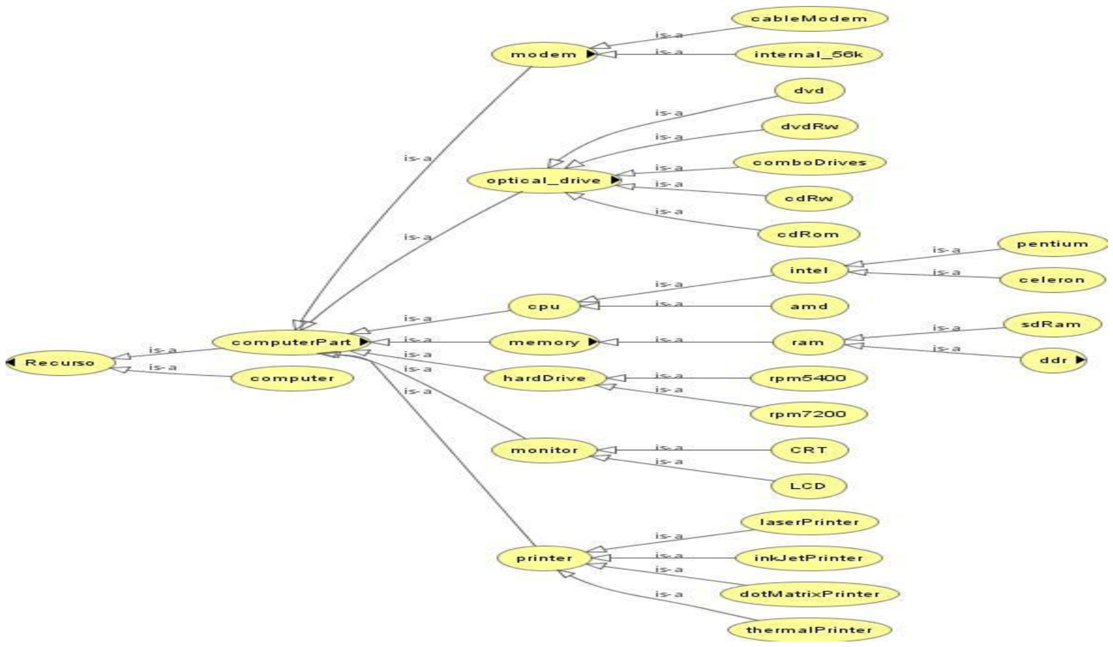

Fig 1.2 Taxonomy tree for Search Interface S2

Similarly the second search interface O2 under consideration becomes as follows

Table 1.3 Codes of terms in search interface B

Term	**Code**
Recurso	B
Computer part	B01
Computer	B02
Modem	B0101
Optical_drive	B0102
CPU	B0103
Memory	B0104
Hard Drive	B0105
Monitor	B0106
Printer	B0107
Cable Modem	B010101
Internal 56K	B010102
DVD	B010201
DVD RW	B010202
Combo-drive	B010203
CDRW	B010204
CDROM	B010205
Intel	B010301
Amd	B010302
RAM	B010401
Rpm5400	B010501
Rpm7200	B010502
CRT	B010601
LCD	B010602
Laser	B010701
Inkjet	B010702
Dotmatrix	B010703
Thermal	B010704
Pentium	B01030101
Celeron	B01030102
Sdram	B01040101
DDRam	B01040102

Now depending upon the length of codes assigned to each term search interface can be divided into clusters so that all terms having codes of same length reside in the same cluster. The methodology used for clustering has been defined earlier. The clusters becomes as follows for each search interface.

The algorithm suggested above makes use of recursive calls so that all children of the terms under consideration are also checked for the similarity. We take example of term Intel in search interface o1 and the same term Intel in search interface O2. Then we try to combine the children of both terms. This term has been selected for the purpose of clarity and simplicity only so that it doesn't become too complex.

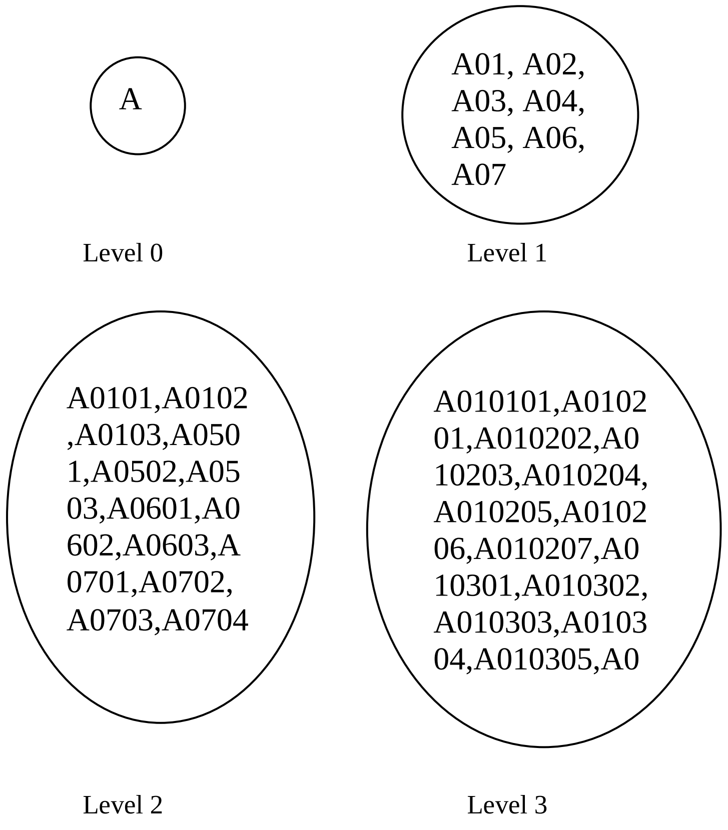
A
A01, A02, A03, A04, A05, A06, A07
Level 0
Level 1
A0101,A0102,A0103,A0501,A0502,A0503,A0601,A0602,A0603,A0701,A0702, A0703,A0704
A010101,A010201,A010202,A010203,A010204, A010205,A010206,A010207,A010301,A010302, A010303,A010304,A010305,A0
Level 2
Level 3

Fig 1.3 Clusters for Search Interface S1

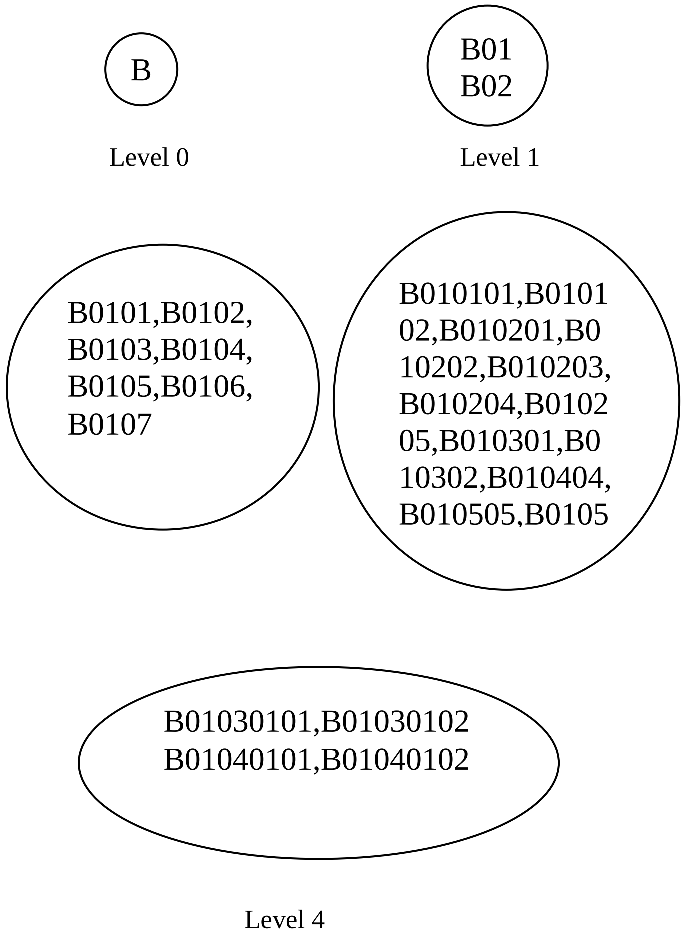
B
B01 B02
Level 0
Level 1
B0101,B0102, B0103,B0104, B0105,B0106, B0107
B010101,B010102,B010201,B010202,B010203, B010204,B010205,B010301,B010302,B010404, B010505,B0105
B01030101,B01030102 B01040101,B01040102
Level 4

Fig 1.4Clusters for Search Interface S2

To see how the process of merging of search interfaces will take place we consider example of term **CPU** which appears in both search interfaces. The process of merging will be followed as per the algorithm suggested above.First of all both search interfaces O1 and O2 will be passed to function Integrate. The two tables table 1 and table 2 are already available with us as given above. While scanning through these tables as given in the algorithm it will be detected that cpu appearing in table 1 is same as cpu appearing in table 2 and whenever two same or similar terms are encountered the codes of both terms are fetched. From tables the fetched codes of both terms in different search interfaces are A01 and B0103 in search interface 1 and 2 respectively. These two codes A01 and B0103 will be passed to function merge. Function merge will make two queues from these two codes one for each code having all its descendants upto any level. So that those queues can be passed to function combine and queues can be merged. In function merge q1 and q2 will be created having code A01 and B0103 in q1 and q2 respectively. The color will be assigned to each term in the queue just to check whether immediate children of the term have been added to the queue or not. Initially term will be of white color and the color will be changed to black when its children has been added to the queue. The process will be repeated until all terms in the queue are black which represents that children of each term has been added to the queue.

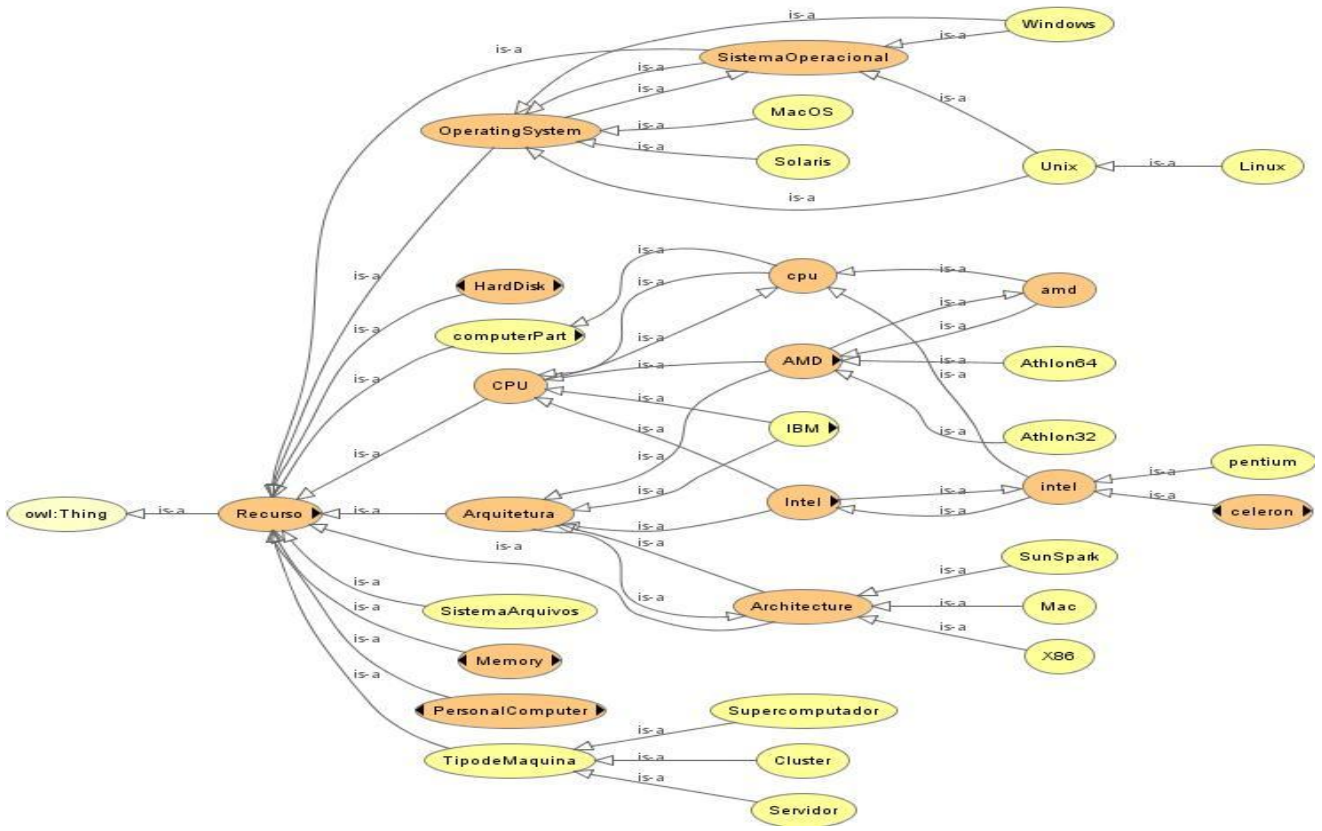

Fig 1.5 Taxonomy tree for Search Interface after merging S1 and S2

This way merging process proceeds in the algorithm. The same process will be repeated for any term in the search interface. And after merging two search interfaces result will be stored in form of a queue only. The result obtained after merging the two search interfaces above the resulting search interface becomes as follows. The case study has shown that how the proposed algorithm is implemented for merging search interfaces. It is seen from the Fig 3.6 that all similar terms appear only once in the resulting search interface and none of the concept is left. The same algorithm can be implemented for merging more than two search interfaces.

REFERENCES

[1] Brian Pinkerton, "*Finding what people want: Experiences with the web crawler*." Proceedings Of WWW conf., 1994.

[2] Alexandros Ntoulas, Junghoo Cho, Christopher Olston "*What's New on the Web? The Evolution of the Web from a Search Engine Perspective.*" In Proceedings of the World-Wide Web Conference (WWW), May 2004.

[3] Vladislav Shkapenyuk and Torsten Suel, "*Design and Implementation of a High performance Distributed Web Crawler*", Technical Report, Department of Computer and Information Science, Polytechnic University, Brooklyn, July 2001.

[4] Junghoo Cho, Narayanan Shivakumar, Hector Garcia-Molina *"Finding replicated Web collections." In Proceedings of 2000 ACM International Conference on Management of Data (SIGMOD)*, May 2000.

[5] Douglas E. Comer, "*The Internet Book*", Prentice Hall of India, New Delhi, 2001.

[6] A. K. Sharma, J. P. Gupta, D. P. Agarwal, "*A novel approach towards efficient Volatile Information Management*", Proc. Of National Conference on Quantum Computing, 5-6 Oct.' 2002, Gwalior.

[7] AltaVista, "*AltaVista Search Engine*," WWW, http://www.altavista.com.

[8] Shaw Green, Leon Hurst , Brenda Nangle , Dr. Pádraig Cunningham, Fergal Somers, Dr. Richard Evans, "*Sofware Agents* : A Review", May 1997.

[9] Cho, J., Garcia-Molina, H., Page, L., "*Efficient Crawling Through URL Ordering*," Computer Science Department, Stanford University, Stanford, CA, USA, 1997.

[10] O. Kaljuvee, O. Buyukkokten, H. Garcia-Molina, and A. Paepcke. "Efficient web form entry on pdas". Proc. of the 10th Intl. WWW Conf., May 2001.

[11] Ching-yao wang, ying-chieh lei, pei-chi chang and shian- shyong tsen National chiao-tung university, *"A level wise clustering algorithm on structured documents"*

[12] Benjamin Chin Ming Fung, *"Hierarchical document clustering using frequent itemsets"*

[13] Francis Crimmins, "*Web Crawler Review*".

[14] A.K.Sharma, J.P.Gupta, D.P.Agarwal, "*An Alternative Scheme for Generating Fingerprints of Static Documents*", accepted for publication in Journal of CSI..

[15] Allan Heydon, Marc Najork, 1999," Mercator: "*A Scalable, Extensible Web Crawler*", Proceedings of the Eighth International World Wide Web Conference, 219-229, Toronto, Canada, 1999.

[16] F.C. Cheong, "*Internet Agents: Spiders, Wanderers, Brokers and Bots* ", New Riders Publishing, Indianapolis, Indiana, USA, 1996.

[17] Frank M. Shipman III, Haowei Hsieh, J. Michael Moore, Anna Zacchi , "Supporting Personal Collections across Digital Libraries in Spatial Hypertext"

[18] Stephen Davies, Serdar Badem, Michael D. Williams, Roger King, *"Google by Reformulation": Modeling Search as Successive Refinement*

[19] Gautam Pant, Padmini Srinivasan1, and Filippo Menczer, "*Crawling the Web"*

[20] Monica Peshave ,"How Search engines work *and a web crawler application"*

[21] Ben Shneiderman ," *A Framework for Search Interfaces"*

[22] Danny C.C. POO Teck Kang TOH, " Search *Interface for Z39.50 Compliant Online Catalogs Over The Internet"*

[23] David Hawking CSIRO ICT Centre, "*Web Search Engines: Part 1"*

[24] M.D. Lee, G.M. Roberts, F. Sollich, and C.J. Woodru, "*Towards Intelligent Search Interfaces"*

[25] Sriram Raghavan Hector Garcia-Molina, "*Crawling the Hidden Web"*

[26] Jared Cope, "*Automated Discovery of Search Interfaces on the Web"*

[27] IVataru Sunayama' Yukio Osan-a? and Iviasahiko Yachida', "*Search Interface for Query Restructuring with Discovering User Interest"*

[28] Stephen W. Liddle, Sai Ho Yau, and David W. Embley, "*On the automatic Extraction of Data from the Hidden Web"*

[29] Gualglei song, Yu Qian, Ying Liu and Kang Zhang, *"Oasis: a Mapping and Integrated framework for Biomedical ontologies"*

[30] Miyoung cho, Hanil Kim and Pankoo Kim, " *A new method for ontology merging based on the concept using wordnet"*

[31] Pascal Hitzler, Markus Krotzsch, Marc Ehrig and York Sure, " *What is ontology merging"*

[32] Natalya Fridman Noy and Mark A. Musen, "*PROMPT: Algorithm and tool for automated ontology merging and alignment"*

[33] Dinesh Sharma, Komal Kumar Bhatia, A.K Sharma, "*Crawling the Hidden web resources", NCIT-07 , Delhi*

[34] Dinesh Sharma, *"Crawling the hidden web resources using agent" ETCC-07. NIT- Hamirpur*