

GENAUM: NEW SEMANTIC DISTRIBUTED SEARCH ENGINE

ICHRAK SAIF ABDELAZIZ SDIGUI DOUKKALI ADIL ENAANAI

ENSIAS, Mohammed V University

Rabat, Morocco

s_ichrak@yahoo.fr doukkali.ensias@gmail.com enaanai@gmail.com

EL HABIB BENLAHMAR

Faculty of science Ben M'sik, Hassan II University

Casablanca, Morocco

h.benlahmar@gmail.com

The rapid development of services based on distributed architectures is now emerging as important items that transform mode of communication, and the exponential growth of the Web makes a strong pressure on technologies, for a regular improvement of performance, so it's irresistible to use distributed architectures and techniques for the search and information retrieval on the Web, to provide more relevant search result, in minimum possible time. This paper discuss some solutions researchers are working on, to make search engines more faster and more intelligent, specifically by considering the semantic context of users and documents, and the use of distributed architectures. This paper also presents the overall architecture of GENAUM; the collaborative, semantic and distributed search engine, based on a network of agents, which is the core part of the system. The functionality of GENAUM is spread across multiple agents, to fulfill user's performance expectations. At the end of this paper, some preliminary experimental results are presented, that attempts to test the user modeling process of GENAUM, using reference ontology.

Keywords: Web search, semantic search engine, distributed search engine, user profile, genaum

1 Introduction

Information search is the most popular activity on the Web, and it is estimated that more than 80% of users use search engines to find their information needs [1]. When a user sends a request to a search engine, usually he gets a long list of results and spends a lot of time to choose the documents that really meet his needs. However, the user does not want to waste time with inappropriate or irrelevant search results. The extraction of relevant information from the web has become a challenge for all current search engines [2]. Search engines require users to translate their information needs into an expression of a formal query language, and they are encouraged to enter short queries by the fact that long queries do not produce good results. But users may find it difficult to express by a few keywords a complex information need. The handicap of computer is mostly due to technological difficulties to understand natural language or document content. Even if the information is conspicuous to a human reader, computer may not be able to see anything else of it, other than a string of characters. In that case, it can

compare documents and keywords provided by the user, but without any understanding of what those keywords would mean [3]. With the emergence of the Semantic Web, this situation can be improved if machines could “understand” the content of Web pages and the context of the information surrounding queries and documents, which allows more accurate search results that better reflect the user’s actual intentions. The query context can affect the user, its environment, its need for information and also the search engine itself. Thus, according to its interests, a user may have a larger or a smaller number of documents, a particular document type, a particular area of expertise... etc. Generally; a result is relevant if and only if it satisfies the constraints and specific needs of the user.

The number of documents available in the Web is increasing massively. According to IDC [4], the digital data universe is projected to reach 44 zettabytes (44 trillion gigabytes) by 2020. The indexing and searching process for search and information retrieval on the Web, will be very difficult to achieve given the bandwidth limitation and response time constraints.

The remainder of the paper is structured as follows. In Section 2, we outline the architecture of search engine and its processes. Section 3 and section 4, present a targeted survey of works on semantic and distributed search engines. Section 4 presents the overall architecture of GENAUM. Some experimental results are presented in section 5. The final section ends with a conclusion and gives an overview of our prospects.

2 Search Engine Process

A search engine is basically made up of four main processes front-end and back-end (Fig. 1):

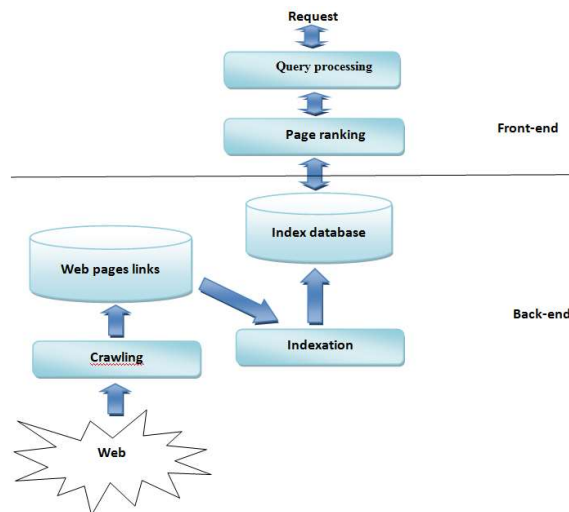


Figure 1 A simplified search engine Architecture

- **Crawling process:** This process is based on robots, also called spiders or crawlers that browse the World Wide Web in a methodical and regular manner, to discover new addresses (URLs) judged interesting. They follow the hyperlinks that connect to each other pages.

- Indexing process: The objective of this process is to extract keywords considered significant in each page identified by crawlers, and store them in a database, called index database. The purpose of indexing is to facilitate a fast and an accurate information retrieval.
- Page ranking process: This process allows ranking and organizing search results, and making the most relevant documents in top of the list.
- Query processing process: It's the process that submits the query and returns the results. Linguistic techniques such as stemming, removal of articles and auxiliary verbs and spelling, are used to reduce noise and increase the recall. Also, algorithms are applied to identify in the index database, documents that best match the user's query.

3 Semantic Search Engines

Semantic search engines try to solve the limitations of keyword search [5]. Their goals are to improve search accuracy and to generate more relevant results, by understanding the objective and the contextual meaning of terms. They integrate the conventional technologies of a search engine and the Semantic Web [6]. Unlike traditional search engine, a semantic search engine provide fewer number of results and more relevant documents, as it considers the semantic context of the given query [7], that saves time for the user.

In the Semantic Web, every query written has syntax and semantic, which is the meaning conveyed. The idea is to provide to machines data that they could understand and from which they could learn new knowledge, and provide a solution for the problem of interoperability which cannot be solved with existing web technologies. Generally, an ontology is used to retrieve information about objects. Ontology is a structured and formal set of concepts that makes sense to the information. It is written in OWL language; one of the languages developed and standardized by the W3C. Ontology-based search engines like Swoogle [8], OntoSelect [9], and OntoKhoj [10] indexed ontologies and capture concepts, their properties, and their relationships for specific domain which can be used by computers to process within the data of those domains semantically.

Many architectures and techniques of semantic search engine are proposed. Shekhar et al [11], presented a semantic approach to search web pages. The objective of this approach is to analyze the user's browsing activity and return semantically relevant web pages for the query provided. The ontology created from the browsing history is analyzed following an input search query. Finally, the corresponding results are returned to the user by providing an organized and semantically meaningful output. Shah et al [12] conducted a comparative study of the most famous semantic search engines: Hakia, Swoogle, Lexxe, Factbites, DuckDuckGo, and SenseBot Engine. He studied the approaches and techniques used by these search engines, based on various parameters, to determine their properties and limitations. He concluded that there is not yet a completely semantic search engine that would be able to respond to search queries in the correct context, and return relevant results. Many algorithms and working methods must be developed before creating a relevant semantic information retrieval system. Also, the specificity of the Internet involves many problems; the obstacle of using search engine on a large scale is more difficult to cross that simultaneously affects the number of documents (volume of data to be processed), sources (numerous and widely distributed) and requests (extremely high). Reliability and performance is more challenging to apply that justify heavy investments by search engines companies. For these reasons, it is wise to think about a distributed search engine at a reduced cost.

4 Distributed Search Engines

A distributed search engine is a search engine where there is not only one single central server. Unlike traditional centralized search engines, the processes such as crawling, indexing and query processing is divided between several decentralized peer. There are several models and architectures in the literature related to distributed search engines. We can classify it in two categories. First of all, there is a group of techniques based on big data that use Hadoop, MapReduce model, Lucene/Solr, HBase and Cassandra.

MapReduce Hadoop is a clustering package for the parallel processing of data offline, allowing applications to work with thousands of nodes and petabytes of data. Apache Lucene / Solr is a search engine based on the Lucene Java library, which provides indexing and searching text in the index database, to find documents, while Solr provides a search user interface above Lucene with additional options such as caching and replication management. Apache Cassandra and Hbase are NoSql databases for managing extremely large data sets.

Chen et al [13] and Hong et al [14] have presented architectures for distributed search engines, where crawling, indexing, and searching are in a distributed manner using MapReduce/Hadoop model. Rui [15] presented a distributed search engine based on Hadoop. Indexes are created according to the MapReduce computation model. These indexes are stored in HBase clusters. Wan et al [16] presented a design and implementation of a distributed search engine based on Apache Nutch, Solr and Hadoop, and he has also proposed a new classification algorithm based on click log and user profiling information.

The second category of techniques used by researchers for creating a distributed search engine is the peer-to-peer (p2p) technology. The p2p architecture connects distributed peers together, where each peer can act as both server and client. This architecture is mostly common in file sharing. Faroo [17] is a search engine where users install a p2p software on their computer. Exploring the web, constructing index and storing, are distributed among engine users. Once activated, and each time they visit a website, it stores website documents in an index database. The index is then used to automatically adjust - in real time - search rankings through the Faroo network. It is based on user behavior to determine the relevancy of websites. For example, a website we regularly back and spend a lot of time will be ranked higher than a site visited only one time.

Seeks [18] is another p2p search engine that can be installed on a personal computer. The same as Faroo, he builds a social search around "the online activity of users." Based on search queries and selected results, in other search engines, Seeks builds a personal profile stored on user's computer. Personal profiles are placed in groups based on "common interests", which allows results filtering.

YaCy [19] is a p2p search engine. Access to the search engine is via a local web server that provides a field for entering the keywords and provides the results as a regular web page. On each peer, YaCy runs either in crawler mode or in proxy mode, and each YaCy peer launches indexing robots that analyze the pages found. The results are recorded in a distributed database (indexed). Once activated in "private" mode, the user can use a robot to explore the sites of their choice. Once collected, the indexes are stored on his personal computer. It can then search in its local index, which is composed only of documents that he analyzed.

Despite their diversity, none of distributed search engines is able to reach a sufficient level of quality and efficiency, to really compete with traditional search engines.

5 GENAUM

A. ENNAINAI proposed in his thesis [20], architecture of distributed and collaborative search engine GENAUM.

GENAUM is a project of search engine started in 2006 in our research laboratory. In its first age, the objective was to extract the relevant parts of documents using the GENE/CLONE method [21]. Its goal now is to improve the relevance of results and the response time by migrating to a semantic, distributed and collaborative search engine.

5.1 *Genaum Architecture*

GENAUM is a collaborative search engine whose principle role is to contribute the maximum of users, to improve the calculation of relevance. So search results presented are the fruit of collaboration between several human users. Figure 2 shows the basic architecture of GENAUM.

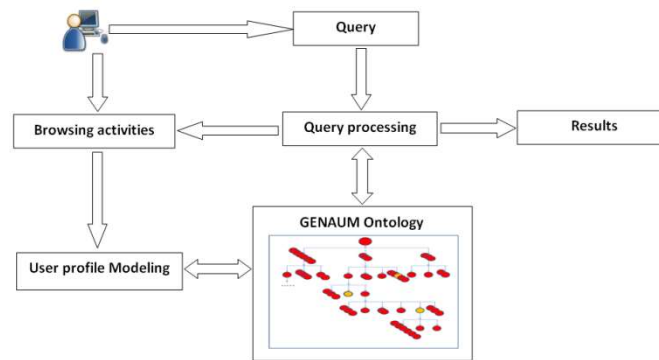


Figure 2 GENAUM architecture

5.2 *GENAUM Ontology*

GENAUM ontology (GENAUM core) is a global ontology, which consists of a growing network of computers called “sons” or “agents”, affiliated to at least one node of the ontology, and supervised by a super agent called the “father”. This father collects all necessary information about his agents, and attributes them to a sub-ontology according to their behavior (navigation mode, interests, context, content ...). He is also responsible of agent’s affiliation and request’s categorization. Each agent is continuously scored based on the relevance it offers on a given sub-ontology (Figure 3).

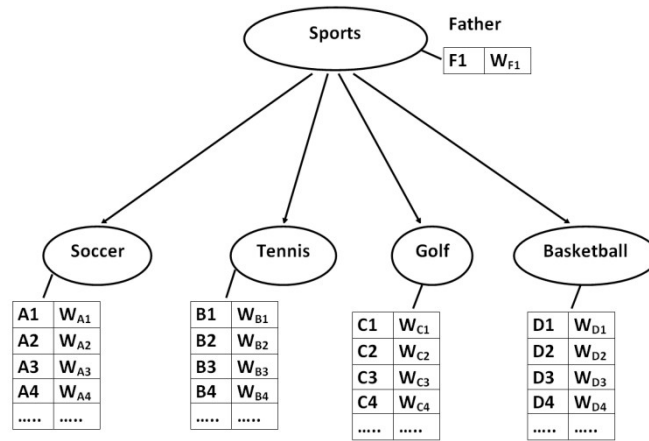


Figure 3 example of a sub-ontology and related agents

5.3 Collaborative search

GENAUM is a collaborative search engine. Therefore, its results are the fruit of collaboration between several human users, and are modified by users or influenced by different systems such as votes, favourite’s management, etc. The collaborative search engine or «the social search" exists through the will of Internet users to give and share their opinion with the others, by leaving comments, bookmarks, making the Page Ranking or noting documents (figure 4).

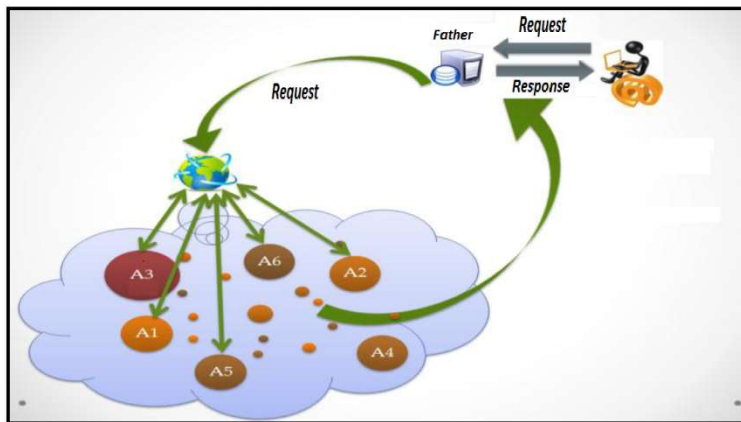


Figure 4 Collaborative search in GENAUM

In GENAUM, The super agent (or the father) is responsible of dispatching tasks, and selects the appropriate agents that can contain the relevant information. Also, it is responsible of grouping agents of the same family in a set of contextual clusters, based on their concept’s weights.

5.4 User Profile Modeling

User modeling allows representing data that characterize the user. Without user profile, a web search engine behaves in the same way with all users, who have different knowledge, preferences, goals and

interests. If we take into account user's preferences about his topics of interest the search engine will provide more relevant information [22]. In GENAUM the user profile is a collection of several behavioral data collected on user's interests and preferences to tailor search results. It's a semantic representation in the form of a hierarchy of concepts.

A plug-in called "Web profiler" will be installed in each agent, and captures submitted queries and other information, like URL, page title, page description and frequency, about all the pages and URLs that user consults, and saves it into an XML file. The idea is when a user clicks on one of the documents in the result list; he has a good idea about what can be found in the document. This provides us with a solid basis for using clicks and browsing activities as an indication of user interests and goals.

Agent categorization and scoring are calculated and updated through an algorithm based on the quantity and quality of information it provides to the network, a sub-ontology or a given concept. Categorization and relevance are not calculated entirely on the basis of syntactic and lexical similarities, but a large part of the calculation is a semantic distance that we call "ontological distance".

5.5 Query Processing

A special processing is applied to the query to find which nodes of the ontology of GENAUM are closest to it semantically. These nodes are called "answering agents" (relevant agents). Then these agents will be selected by the father to respond to the user request. A relevant agent is an agent who has a high interest towards a specific concepts represented by the node. It plays the same role as single agent. The only difference is that it can respond to requests sent by the others. Note that an agent may be affiliated with several groups, so it can become a relevant agent in multiple groups, if the number and manner with which it consults the pages is preferred compared to other agents of group.

A request submitted to GENAUM is resubmitted to all the relevant agents having the highest ontological similarity with it. To decide which agent is most relevant, a calculation of the ontological distance is realized. This is an assessment of the similarity between the request and agent concepts. The father performs selection and ordering of answers, to give them to the user that issued the request.

5.6 Document and Agent Relevance

Information stored in each user computer about his interest and concepts, is used to give a general idea on the importance of the current page. Document title and description, are rich elements describing the content of the document that we can use to index and categorize web pages. Frequency helps us to compute the document relevance. As a user views a page several times, and for a long time, it becomes relevant to him. On the other hand, the same page can be viewed by another user, so it will have a different value of relevance which is usually different. Finally, the average of all dynamic pertinences gives the opinion of agent's community against a page. So the popularity and the relevance of a document are inferred by taking into account behaviors and implicit advice of agents who have previously visited this page (figure 5).

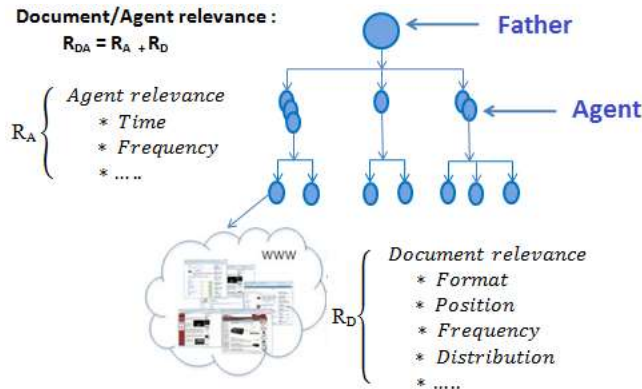


Figure 5 Agent and document relevance

6 Experiments

The main purpose of the experiment is to evaluate the efficiency of user modeling in GENAUM. In our search engine the user profile is generated automatically, using behavioral data collected on user’s interests and preferences. This information is provided by submitted queries and clicked links; once a user submits a query, or chooses a document from result search, we use an algorithm to extract the most relevant concepts and classify user’s queries and web pages automatically into concepts on the reference ontology. The Mapping requires an aggregate representation of the reference ontology by computing a term vector for each of its concepts.

6.1 Data Collections

In our experiments we used DMOZ as reference ontology to classify user queries and clicked documents.

The DMOZ directory is organized as a tree, where the topic categories are inner nodes and web pages are leaf nodes. Figure 6 shows an example of the sub-ontology Top/Computers in DMOZ.

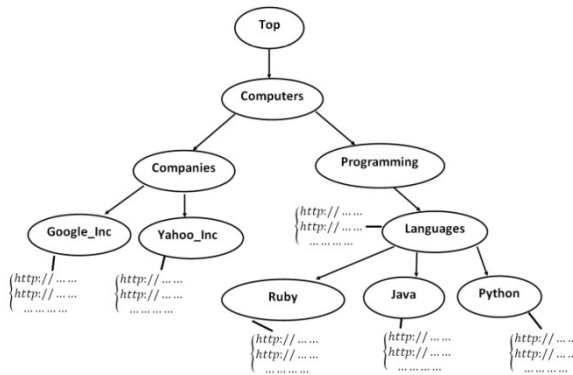


Figure 6 Example of the sub-ontology Top/Computers in DMOZ

We represented each concept (category) of DMOZ, by a vector of weighed terms selected from the title and the description of the associated web pages, using the technique presented in [23]. The vector is used to extract concepts of the user queries and clicked documents.

For this, we proceeded as follows:

- (i) We parsed the complete DMOZ directory into Mysql Database. In our experiments we exclude categories under the “World” category, because it contains categories in languages other than English.
- (ii) For each category, we concatenated the associated titles and descriptions of URL links, into a super-document. Therefore each category C_i has an associated super-document D_i .
- (iii) We removed stop words and applied porter stemming algorithm, to stem super-documents terms.
- (iv) We represented each super-document D_i by a vector of weighted terms, where the relevance value of the term t in D_i is computed using the $tf *idf$ weighting scheme as follows:

$$w_{tD_i} = tf(t, D_i) * \log\left(\frac{n}{n_t}\right). \quad (1)$$

Where

n is the number of super-documents in collection,

n_t is the number of super-documents containing the term t ,

$tf(t, D_i)$: is the total frequency of the term t , in the super-document D_i , and also in each of the super-documents D_k , where C_i has n related sub-concepts C_k each one is represented by D_k

$$tf(t, D_i) = \frac{[tf(t, D_i) + \sum_{k=1}^n tf(t, D_k)]}{n+1}. \quad (2)$$

- (v) We stored all the vectors of weighted terms calculated, and their categories, in a new table in our Database.

The collection that we created contains 356 480 super-documents, 356 480 vectors and 14 616 529 associated terms.

6.2 Query Categorization technique

In this section, we discuss the method used to generate a list of suggested categories for a query given by the user.

To decide candidate concepts for a query, the similarity between concepts vectors and the vector representing the query is calculated, using the cosine similarity technique. Cosine similarity is the best metric which is used frequently when trying to determine similarity between two texts [24]. If query keywords appears in a super-document related to a category, its importance and the similarity will be high, otherwise it will be low. For cosine similarities resulting in a value of 0, the query do not share any attributes (or words) because the angle between the objects is 90 degrees.

The cosine similarity function is determined by the following formula:

$$\text{sim}(D, Q) = \frac{D \cdot Q}{\|D\| \|Q\|} = \frac{\sum_{i=1}^m w_i y_i}{\sqrt{\sum_{i=1}^m w_i^2} \sqrt{\sum_{i=1}^m y_i^2}} \tag{3}$$

Where

D is a super-document; $D = (w_1, \dots, w_m)$

Q is a query; $Q = (y_1, \dots, y_m)$

w_i and y_i are terms weights in D and Q.

Mathematically we consider that two vectors are similar when the cosine is more than 0.8. The concepts with the most similar representing vectors to the query are thus considered as candidates concepts.

6.3 Results

The system has been implemented in python, which provides several packages and APIs for creating and manipulating various types of vectors and computational mathematics.

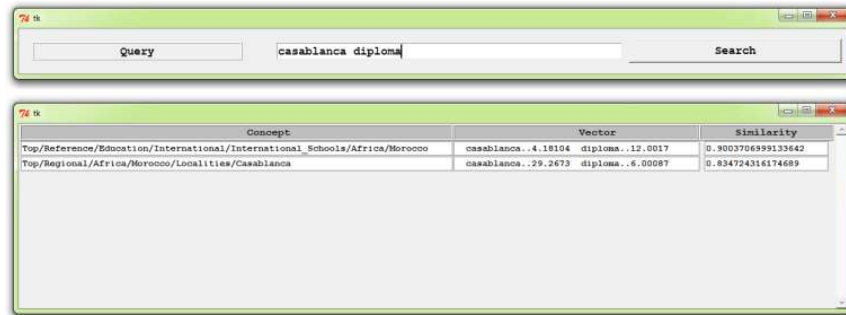
The interface in figure 7 shows an example of categorization for the query “Casablanca casting”. The system interacts with our collection dataset and concepts vectors are analyzed. The cosine similarity between concepts vectors and query vector is calculated. If the similarity is less than the threshold value (0.8), the concept is eliminated. Otherwise the concept is selected as candidate concept.

The results retrieved shows that the category Top/Arts/Movies/Titles/C/Casablanca matches the query submitted with frequency equal to 0.814, and that the context for this query is the movie “Casablanca”.



Figure 7 Result for the query “Casablanca casting”

The second example in figure 8 shows that the candidates concepts for the query “Casablanca diploma” are Top/Regional/Africa/Morocco/Localities/Casablanca And Top/Reference/Education/International/International_Schools/Africa/Morocco. By evaluating, the threshold and the similarity scores, the results shows the two concepts are the most similar as the similarity scores are greater than or equal to the threshold value (0.8). The context for this query is education in Casablanca and Moroccan city.



The image shows two windows from the GENAUM search engine. The top window is the search interface with a query input field containing "casablanca diploma" and a "Search" button. The bottom window displays the search results in a table format.

Concept	Vector	Similarity
Top/Reference/Education/International/International_Schools/Africa/Morocco	casablanca..4.18104 diploma..12.0017	0.3003706999133642
Top/Regional/Africa/Morocco/Localities/Casablanca	casablanca..29.2673 diploma..6.00087	0.834724316174689

Figure 8 Result for the query “Casablanca diploma”

We have conducted several experiments, to find candidate concepts for a given query, and we conclude that user context can be determined automatically, using the dataset generated from DMOZ. This context will be used for user modeling in GENAUM. Also the DMOZ is a suitable for agent’s hierarchy representation in GENAUM core.

7 Conclusion

The architecture proposed in GENAUM is a solution to address the issues of centralized architecture at a lower cost. It accelerates the tasks by reducing the processing time through the direct links between agents (peers) which increase system performance by sharing the workload, and resources aggregation. Another benefit is that information about user preferences and pertinence is received "from source" and regardless of its platform, this being made possible only through the WebProfiler plug-in that observes the entire user’s navigation, while conventional search engines capture only preferences and relevance related to the use of the engine platform. This is a future dimension of user-driven approach. Privacy concerns are also reduced since the user profile is strictly stored and used on the client side. Cosine similarity technique was applied in this paper to extract candidate concepts for a query, and the experiments validate that DMOZ is suitable as reference ontology in GENAUM.

References

1. A. Spink, B.J. Jansen, C. Blakely, and S. Koshman. A study of results overlap and uniqueness among major web search engines. *Information Processing & Management*, 2006. 42(5), 1379-1391.
2. B. B. Cambazoglu, and R. Baeza-Yates. Scalability challenges in web search engines. *Advanced topics in information retrieval*, 2011. 33, 27-50.
3. E. D. Liddy. Enhanced text retrieval using natural language processing. *Bulletin of the American Society for Information Science*, 1998. 24(4), 14-16.
4. J. Gantz, and D. Reinsel. *The Digital Universe In 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East*. Technical report. Internet Data Center(IDC), 2012.
5. J. Beal. Weaknesses of Full text search. *The Journal of Academic Librarianship*, 2008. 34(5), 438-444.
6. A. Ramachandran, and R.Sujatha. Semantic search engine: A survey. *International Journal of Computer Technology and Applications*, 2011. 2(6), 1806-1811.

7. J.Sirisha, B.V.Subbarao, D. Kavitha, and Y. Padma. A Comprehensive Study on Generalized Search Engines versus Semantic Search Engines. *International Journal on Recent and Innovation Trends in Computing and Communication*, 2014. 2(4), 757-761.
8. L. Ding, T. Finin, A. Joshi, R. Pan, R. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs. Swoogle: a search and metadata engine for the semantic Web. *Proceedings of the 13th ACM Conference Information and Knowledge Management*, New York, USA, 2004. 652-659.
9. Y. Zhang, W. Vasconcelos, and D. Sleeman, Ontosearch. An ontology search engine. *Proceedings of the 24th SGAI International Conference on Innovative techniques and Applications of Artificial Intelligence*, Washington DC - USA, 2004. 256-259.
10. C. Patel, K. Supekar, Y. Lee, and E. K. Park. OntoKhoj: a semantic Web portal for ontology searching, ranking and classification. *Proceedings of the 5th ACM international workshop on Web information and data management*, 2003. 58-61.
11. M. Shekhar and RA. K. Saravanaguru. A case study on semantic web search using ontology modelling. *International journal of engineering and technology*, 2013. 5(3), 2342-2348.
12. V. Shah, A. Shah, and K. Deulkar. Comparative study of semantic search engines. *International Journal Of Engineering And Computer Science*, 2015. 4(11), 14969-14972.
13. N. Chen, and C. Xiangyang. Investigation of Distributed Search Engine Based on Hadoop. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, 2014. 12(9), 6954-6957.
14. Z. Hong, M. Yan-hong , M. Wei-jun, B. Zhong-xian. Study of Distributed Personalized Search Engine. *Advanced Materials Research*, 2013.756, 1035-1039.
15. R. GU. Research Distributed Search Engine Based on Hadoop. *Proceeding of the International Conference on Network and Information Systems for Computers*, 2015. 373-375.
16. J. Wan, B. Wang, W. Guo, K. Chen and J. Wang. A Distributed Search Engine Based on a Re-ranking Algorithm Model. *Proceeding of the 10th International Conference on Computer Science & Education*, 2015. 640-644.
17. <http://www.faroo.com/>.
18. <http://www.seeks-project.info/>.
19. M. Herrmann, K. Ning , C.Diaz, and B. Preneel. Description of the YaCy Distributed Web Search Engine. Technical report, KU Leuven ESAT/COSIC, iMinds, 2014.
20. A. ENAANAI. La méta-recherche en langue Arabe : Une approche hybride pour calculer la pertinence documentaire. Thesis at Ecole Nationale d'Informatique et Analyse des Systèmes, Rabat, Morocco, 2014.
21. H. Benlahmer, A. S. Doukkali, and A. Elouerkaoui. A solution for data extraction by a new approach :The method of gene/clone. *Proceeding of the international conference on information and communication technology for the Muslim world*, Kuala Lumpur, Malaysia, 2006. 21-23.
22. M. Bouzeghoub, D. Kostadinov. Personnalisation de l'information : aperçu de l'état de l'art et définition d'un modèle flexible de profils. *Proceeding of Conférence en Recherche d'Informations et Applications CORIA'05*, Grenoble, France, 2005. 201-218.
23. M. Daoud, L. Tamine-Lechani, and M. Boughanem. Using a concept-based user context for search personalization. *Proceedings of the world congress on engineering*, London, UK, 2008. 293-298.
24. A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on Web-page Clustering. *Workshop on Artificial Intelligence for Web Search*, Texas, USA, 2000. 58-64.