A Ranking Algorithm integrating Vector Space Model with Semantic Metadata

Saurabh Wadwekar
WiDiCoReL Research Lab
Maharashtra Institute of Technology
S. No. 124, Paud Road, Kothrud
Pune 411038, India
+91-99601-79733
sw.saurabhw@gmail.com

Debajyoti Mukhopadhyay
WiDiCoReL Research Lab
Maharashtra Institute of Technology
S. No. 124, Paud Road, Kothrud
Pune 411038, India
+91-77091-52655
debajyoti.mukhopadhyay@gmail.com

ABSTRACT

Semantic Web is defined as a structured collection of information, which specifies the rules of inferences required to derive a specific piece of information from it. Data in semantic web is organized in a structured manner, and the relationships clearly defined amongst them. The potency of semantic web lies in is its capacity to comprehend the meaning of the given data, and connect it with other available pieces of information, to produce a new set of tangible knowledge. The algorithm developed in the paper takes advantage of this property of semantic web, and generates clusters of data based on the semantic metadata present in them. A system has been proposed to analyze the observations of algorithm implementation. The system is studied alongside an existing semantic web search engine, to assess the quality of its search results.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – Search process, Retrieval Models

General Terms

Algorithms, Performance, Design, Experimentation, Theory, Verification.

Keywords

Clustering, Semantic web, mining, vector space, information retrieval, document vector, ontological domains

1. INTRODUCTION

Conventional retrieval models place greater emphasis on the keywords present in the document than the semantic meaning of entire data [13]. They read the documents, index the data present in them, and cluster them according to the keywords present in them. Standard clustering algorithms include hierarchical clustering, centroid-based clustering and density-based clustering. In most cases, clustering calculates the proximities between the documents by taking into account the common keywords in them. This often leads to ambiguity since the keywords translate to different meanings corresponding to the context in which they are used. In the algorithm proposed in this paper, clustering is performed according to similarities in meaning of the content. Initially document vectors are constructed using the standard vector space model. Vector space model was originally proposed by Gerard Salton, A Wong and C.S. Yang [2].It is a simple as well as an efficient model to index data. In vector space model, the documents are represented as vectors. The relevance weight of the keywords present in them is calculated using the tf-idf (term-frequency inverse document frequency) method.
The weight of any keyword with respect to a document is given by:

$$W(t,d) = t\text{-}f * \log(n/t(n))$$

Here, t-f is the term frequency, n is total number of documents and t (n) is number of documents in which the term appears.

The vector space model is effective in offering search results on keyword basis, but fails to take into account the semantic nature of the data. This often leads to 'false positives' or 'false negatives'. This limitation can be handled by creation of ontological domain vectors, which is explained in the subsequent section. The ontological domain vector comprises of semantic information related to a single document. Clusters of documents can be prepared by determining vectors with least distances. In the IR process developed here, user is offered a choice to drill down on his preferred search result. This drill down evaluates the closest documents (semantically) to the selected result, and offers next set of results accordingly. One advantage of such drill down is user can narrow down his search to his topic of interest without specifying the topic of interest beforehand.

This paper initially outlines research advances in the domain of ontological based clustering and ranking algorithms. It then proceeds to describe the design rationale behind the new system and its specifications. Finally, sIRch, the developed system is tested against user queries and its performance is evaluated alongside an existing semantic web search engine. The paper concludes by noting the advantages of the approach used and stating its scope of improvement.

2. LITERATURE REVIEW

Researchers have carried out substantial amount of work in proving that clustering is greatly improved by taking ontology into consideration. Hotho et al [3] described the improvements in text based clustering method by mapping the terms in the document to a set of concepts. William De Luca et al [4] used the WordNet ontology to classify the documents into different 'Sense Folders', proving that ontological classification provided better results. The method provides disambiguation information of a query with respect to documents of a result set. Aussenac-Gilles [5]

CUBE 2012, 3-5 September, 2012, Pune, India.

also developed an algorithm which performed search within a context. The search results were from a hierarchical set of concepts extracted from ontology. Controlled vocabulary was used here, and documents were clustered according to the information space defined by the users. This type of clustering would be effective for specialized needs, as the information space needs to be defined before clustering. Zhanget al[6]showed that 're-weighting' the words in a vector to find semantically similar core words yields better results on PubMed document sets. Exploiting the semantic metadata has always shown to generate better clusters, with documents of similar meaning grouped together. A need was felt by us to develop an algorithm which would work on any data set and offer search results through semantic analysis. Also, the search results should be accurate and fast enough for practical purpose. The work carried out by us is primarily focused on creation on ontological vectors for each document, which represent the concepts and social contexts extracted from the document. Clustering is as a result of calculations upon these vectors. A system has been proposed which implements the developed algorithm to generate efficient search results. The principles of the system can be applied to any information retrieval systems without modifications.

3. PROBLEM DESCRIPTION

A system is to be designed which clusters data based on semantic analysis and ranks them accordingly. The algorithm developed should work with any kind of data set and provide swift search results. Primary aim in developing the system is to offer semantically relevant results. In this paper, an effort is made to develop a system which would offer high quality semantically relevant search results. The developed algorithm is implemented on top of vector space model.

4. PROPOSED SYSTEM

4.1 Requirements

A system which would offer semantically relevant results by clustering the documents on the basis of ontological information present in them. The response time of the system should be small enough for practical use and it should not restrict itself to result set of any particular domain. Foremost step in satisfying this requirement is creation of ontological domain vectors of each document.

4.1.1 Generic domain vector equation

Ontological domain vector constitutes the main part of the clustering process in our algorithm. For each document, a vector describing its metadata is constructed. Semantic metadata extraction from a piece of data yields the following information:

1. Ontological Domain Information
2. Social Contexts/Topics present in the document

The ontological domain information highlights the broad domains in which the document falls. Social contexts or topics are the numerous subjects which are covered in the document.

A Web-service (OpenCalais) [7] is used to extract the semantic metadata from a document. Document ontology domain vector is constructed on the basis of this data.

For example, a document containing an article of networking device 'router' contains the following semantic metadata, as shown in Table 1:

Table 1. Semantic information extracted from a document containing data about 'router'

Type	Term
Domain	Technology Internet
Topic	Computing
Topic	Network architecture
Topic	Computer networking
Topic	Open Shortest Path First
Topic	Routing table
Topic	Server appliance
Topic	Router
Topic	Networking hardware
Topic	Internet protocols

The domain 'Technology Internet' summarizes the entire meaning of the document under its domain. Different topics which are covered in the document can be condensed into terms such as 'Networking hardware', 'Internet Protocol'.

The vector of any document could be generically specified as:

$$\overrightarrow{D1} = (a.\overrightarrow{d1} + b.\overrightarrow{d2} + \ldots + n.\overrightarrow{dn}) + (p.\overrightarrow{t1} + q.\overrightarrow{t2} + \ldots + z.\overrightarrow{t3})$$
$$= \vec{D} + \vec{T}$$

Where $\overrightarrow{d1}, \overrightarrow{d2}, \overrightarrow{dn}$ are ontological domains and a, b, c are the weights associated with them. $\overrightarrow{t1}$, $\overrightarrow{t2}$, $\overrightarrow{tn}$ are the topics, and p, q, t their respective weights. $\vec{D}$and$\vec{T}$ are generic vectors representing the ontological domain and topic of the document. For example, according to the semantic information extracted, the ontological domain vector for a file containing content about 'Cisco' would be –

$$\overrightarrow{D1} = (a.\overrightarrow{d1} + b.\overrightarrow{d2}) + (p.\overrightarrow{t1} + q.\overrightarrow{t2} + r.\overrightarrow{t3} + s.\overrightarrow{t4} + u.\overrightarrow{t5} + v.\overrightarrow{t6})$$

Where $\overrightarrow{d1}$ is domain 'Technology Internet', $\overrightarrow{d2}$ is domain 'Business Finance',$\overrightarrow{t1}$ is topic 'Computer Networking' and so on.

The proximity or nearness between any two documents can be calculated by taking the dot product of two documents' vectors:

$$\overrightarrow{D1}.\overrightarrow{D2}$$

Two different types of vector constitute any single document's vector (domain and topic); the dot product would also yield two types of numeric values:

$$a.\vec{D} + b.\vec{T}$$

An ontological domain would be a highly generalized term which would have many topics under it. Since the ontological domain is superset of the topics in the above vectors, the $\vec{D}$ in the result is given precedence over the $\vec{T}$. A dot product with high value of the

domain coefficient denotes a greater proximity than a dot product with lower domain coefficient and higher topic coefficient.

4.1.2 Vector weight calculation

The value of weights associated with any domain/topic can be calculated using different methods. One method of calculating weights is described below. For example,

= (a. + b. +…. + n.) + (p. + q. + … + z.)

The value of coefficient 'p' can be calculated as:

p=∑ (Keywords which can grouped under)

This weight assignment would have to be based on an evolutionary algorithm [8], since the keyword categorization under an appropriate topic is a subjective process. A keyword, for example 'bill' can be grouped under 'person' or 'price' depending upon its relative usage in the document.

The value of coefficient 'a' of ontological domain can be calculated as:

a= ∑ (Topics of)

Each of the topics (or social contexts) which are extracted from the piece of information can be grouped under a broader ontological domain. The weight of any domain found in a document can simply be calculated by adding the number of that specific domain's topics present in the document. One more way to determine the domain weights can be:

a= ∑ (Topics coefficients)/ Number of Topics

In above formula, the topics considered are the only ones which fall in the domain, and the topic coefficients are the weights calculated in the previous section. In the simulation described below, the value of coefficients is set as 1 by default.

The weight of topic coefficients can be derived by interpreting the significance of the topic in context to the particular document. This can be determined while semantic metadata extraction is performed. Topic segmentation, in natural language processing can be used in deriving topic coefficients.

4.2 Design Rationale

A data set is first selected on which the clustering and ranking is performed. Wikipedia database dump[9] has been selected for this purpose due to diverse nature of information available on it. Back end of simulation is developed using Java, and the front end is designed using JSP Servlets. REST APIs are used in Java to send the document data to the web service. The semantic data in the form of vectors is stored in the Oracle 11g database. Servlets offers the powerful feature of rendering Java functionality onto the front end. Detailed description of the system is explained in the next section.

4.3 Theoretical foundation of sIRch

sIRch system is designed to index the documents and provide a search engine to effectively query it. The back end of the system is developed in Java, with Oracle 11g as the database. The sample document dataset used for indexing is the Wikipedia database dump. The database dump consists of a single XML file of around 27GB, which renders it unsuitable for loading it into memory by the Java DOM parser. SAX parser API offers enhanced functionality while dealing with large XML files, but the huge size of the file also made this method unreliable. As a result, the single XML file was subsequently divided into smaller text files, which could be easily read by the Java APIs. For carrying this task, a software written in Ruby called WP2TXT [10]is used. IDE used for development is NetBeans 7.0.1, and the version of JDK is 1.6.Indexing steps in creating the vector space model are –

a. read_file(f) – Read file f.
b. Separate_articles(f) – Divide the file according to different articles present in it.
c. ∑ Create_forward_index[a(i)], i=0 to n – For each article, create a forward index file containing the term information.
d. ∑ Create_reverse_index[a(i)], i=0 to n - mergethe indices into the common index pool – resulting in reverse indices.
e. Calculate_weight() - calculate tf-idf weights of all the words. The weight calculation can also be performed separately at the end, and only once, since the weights change when any new file is added. Determining the weights after all the files have been indexed appears to be a more practical option.

The indexing procedure in the proposed system is accompanied by the domain clustering of data. After one article is read completely from the file, it is sent to the web service (OpenCalais) for domain and topic extraction. The web service creates rich semantic metadata for the submitted content using methods like natural language processing and machine learning. This step is succeeded by vector creation – storing the received information in the form of a vector. Clustering is done by calculating the dot product of two vectors. The documents with high dot product values can be said to be similar and thus clustered together. An average running instance of back-end looks like this:

```
Reading - Acoustic metamaterials
SENDING IC 289 TO WEB SERVICE FOR DOMAIN CLASSIFICATION...
Calculating proximities of - file319_vector
DONE CLASSIFICATION. CREATING WORD TABLES FOR - IC 289
NOT CALCULATING WEIGHTS...
```

Figure 1. A running instance of indexing algorithm

The above depiction shows the working of developed algorithm in a condensed manner. The article on 'Acoustic metamaterials' is the last article in the file being currently read. After all the articles in a particular file are read, the first piece of data, an article about 'IC 289' is sent to the web service for ontological domain classification. This step ends in creation of the vector, storing the IDs of the domains/topics returned. Clustering is performed after this step. Dot product of the newly created vector is taken with all the other vectors stored in the database. The closest documents are then stored in a separate table for swift retrieval during the search process. The line 'Calculating Proximities of – file319_vector' denotes the product calculation of the file named as file319. After proximity calculation, document vectors according to vector space model are created. The weight is not calculated in every individual file's operation; rather it is calculated once in the end.

The algorithm for the back end can be summarized as:

a. Read the directory in which the files are present
b. Read the file in which the data is present
c. If new article found, send the article to web service

d. Create fileno_vector consisting of domains and topics found in the document

e. Calculate the proximities of the new article and store it in file_proximity. This vector clusters the files according to their semantic metadata.

f. Calculate the document vectors according to vector space model and assign tf-idf weights to the words

g. If more data present in the directory, go to step b

The file_vector mentioned in step d stores the domain information. It consists of an ID field, and a 'type' field for identifying whether the entry is a domain or a topic. A typical file vector looks like (Figure 2) –

ID	TYPE
441	s
577	s
578	s
579	s
580	s
247	s
13	t
4	t
7	t

Figure 2. A file vector

Some identifier details, which are stored in the above file vector, are as follow -

441 – Games, Topic

578 - Tile based board games, Topic

13 – Sports, Domain

7 – Business Finance, Domain

file_vector captures the multi-dimensional semantic nature of a piece of data into a single vector. This type of representation greatly helps in capturing the nature of entire content into a single compact equation.

Centroid based clustering is used to create clusters. Documents having overlapping domains or topics are clustered together. Centroid is decided by the most common domains or topics present in the considered data set.

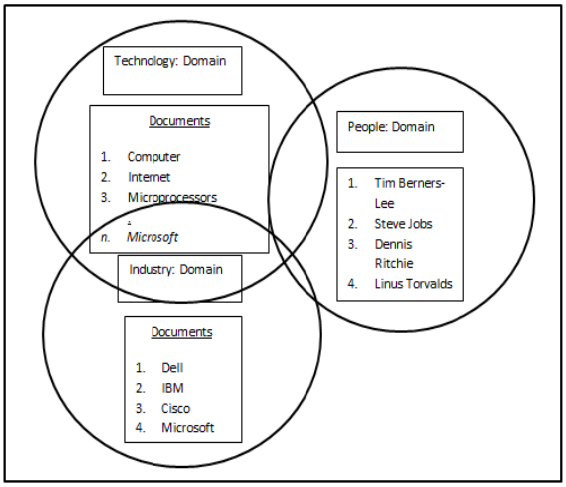

Figure 3.. Possible cluster formation by ontological domain clustering

Figure 3 depicts that one document can be part of more than one cluster according to the nature of content present in it. If three broad clusters are created according to domains, then a document about 'IBM' will fall into 2 major clusters – Technology and Industry.

4.4 sIRch – Prototype implementation

sIRch is an IR (information retrieval) system which is designed to observe the results of the algorithm presented in the paper. sIRch is a web portal developed in JSP (Java Server Pages), Servlets and HTML. The system is deployed on Apache Tomcat 7.0.14 server. The basic interface of the system looks like (Figure 4):

Figure 4.sIRch interface

5. EXPERIMENTAL RESULTS

For observing the results of ontological domain clustering implemented in sIRch, query term 'Texas' is chosen. The results for the particular query are shown in Figure 5:

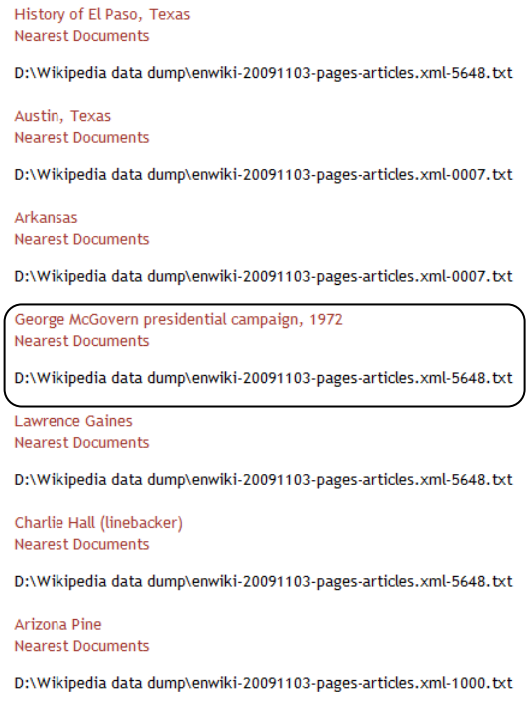

Figure 5. Search results for 'Texas'

All the search results are associated to Texas in some or other manner. For observing how the algorithm works out, we select the 'Nearest Document' link of 'George McGovern presidential campaign, 1972.' The results of selecting the closest document are as shown (Figure 6):

Figure 6.Documents closest to 'George McGovern presidential campaign, 1972.'

All the retrieved results are very similar to the original document of George McGovern. For example, the topmost search result is of Glen Casada, who was also a politician of same era. The fact that the document is the topmost search results suggests that the two documents have semantically common domain/topics. Closer inspection tells us about the common topics (Table 2):

Table 2. Common domains/topics between 'George McGovern presidential campaign, 1972' and 'Glen Casada'

Type	Term
Domain	Politics
Topic	United States

Both the domains fall under the ontological domain of 'Politics' and the common topic between the two is 'United States'. Algorithm for arriving at search results through the sIRch can be summarized as:

a. Read the query.

b. Remove the stop words

c. Stem the query

d. Returndocuments in order of [w(k1)+w(k2)+…+w(kn)], where k1, k2 are the stemmed keywords in the query, and w(k1) is the weight of the keyword with respect to a particular document.

e. Offer suggestions of nearest documents along with every search result.

6. RESULT OBERVATIONALONGSIDE AN EXISTING SEMANTIC WEB SEARCH ENGINE

In this section, search results from the newly designed system sIRch are analyzed alongside an existing semantic web search engine. The web search engine used for comparison provides an option of retrieving similar results based on the content of the given URL. The query selected for performing the test is 'constellations'. The query yields the following results in the sIRch (Figure 7):

Figure 7. Search results for 'constellations'

All the search results are either constellations in themselves or are related to star systems in some manner. The search result 'Argo Navis' is selected for drilling down on nearest documents. The results of nearest documents of Argo Navis are (Figure 8):

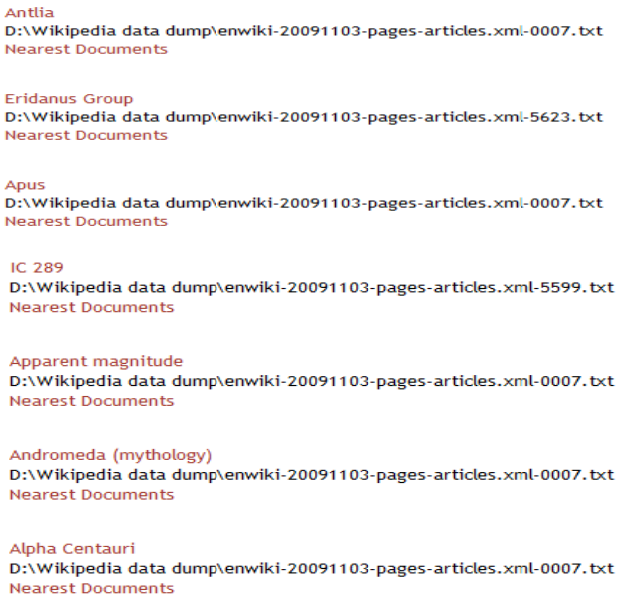

Figure 8. Nearest Documents of Argo Navis constellationas per sIRch

A current operating semantic web search engine which offers results based on semantic analysis is selected for observations.

The query entered in the search engine was – 'constellations like argo navis'-

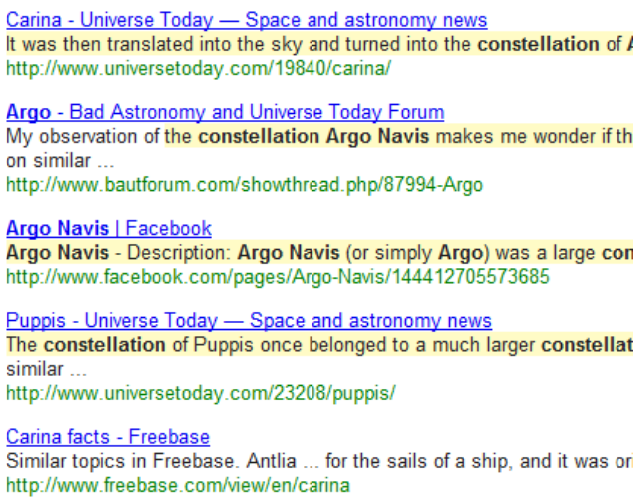

Figure 9. Results obtained from a web search engine against similar site search for 'Argo Navis' constellation

This is a screenshot (Figure 9) showing the first page of search results. It can be observed that the search engine is not able to semantically understand the meaning of the query.The query is unambiguous about the fact that the 'argo navis' refers to a constellation and the user intends to search for similar constellations. But the results generated are not accurate. Apart from the result of 'Carina', none of the results are constellations similar to 'argo navis'. Whereas in the proposed system's result, almost all the 'nearest documents' are close to the actual intended results. Nearly all the results are related to constellation, and the type of content is explanatory in nature. This type of relation is possible when the data is clustered according to the semantic metadata. Precision in the results of sIRch can be calculated as:

Precision = | {Relevant Documents}∩ {Retrieved Documents}| / |{Retrieved Documents}|

= 6/7 (the document titled 'apparent magnitude' can be termed as an irrelevant result)

~ 86%section.

Difference in data sets of two algorithms may be a possible reason for variance in result sets.

7. CONCLUSION

The algorithm described in the paper provides an alternate approach to cluster documents according to the meaning of their content. The proposed system demonstrates how the existing search techniques can be improved by incorporating ontological domain clustering techniques in them. Analyzing the nature of data would play a pivotal role in developing search techniques in Semantic web. The proposed system shows how the semantic metadata can be exploited to cluster data and ultimately offer relevant search results. The algorithm can be improved by designing more innovative ways to assign weights to ontological domain coefficients. Extraction of semantic data is a crucial step, and refinement in its methods would consequently lead to better clustering algorithms. The ranking algorithm can also be refined by adding support for multiple ontology supported domains [11]. (i.e. apart from the one which is being used for clustering in the proposed system, i.e., ontology developed by OpenCalais [3]) Work could also be carried out for constructing vectors in shared ontologies [12].

8. ACKNOWLEDGEMENTS

We would like to thank the OpenCalais Web Service for allowing us to extract rich semantic metadata from a document. We would also like to thank the Wikimedia dump service for providing us with sample data on which the proposed system could be experimented.

9. REFERENCES

[1] Tim Berners-Lee and James Hendler; Scientific publishing on the Semantic web- Nature (2001) Volume: 410, Issue: 6832, Publisher: History Today Ltd., Pages: 1023-1024

[2] G. Salton, A. Wong, C.S. Yang ; A vector space model for automatic indexing -Communications of the ACM Volume 18 Issue 11, Nov. 1975 ACMNew York, NY, USA

[3] Andreas Hotho, Alexander Maedche and Steffen Staab: Ontology-based Text Document Clustering - Intelligent Information Processing and Web Mining, Proceedings of the International IIS: IIPWM'03 Conference held in Zakopane (2003) , p. 451-452

[4] Ernesto William De Luca and Andreas Nürnberger - Improving Ontology-Based Sense Folder Classification of Document Collections with Clustering Methods : Proc. of 2nd Int. Workshop on Adaptive Multimedia Retrieval AMR 2004, part of ECAI 2004, (2004)

[5] Nathalie Aussenac-Gilles & Josiane Mothe - Ontologies as Background Knowledge to Explore Document Collections: RIAO, page 129-142. CID, (2004)

[6] Xiaodan Zhang, Liping Jing, Xiaohua Hu, Michael Ng, Xiaohua Zhou- A Comparative Study of Ontology Based Term Similarity Measures on PubMed Document Clustering :DASFAA, volume 4443 of Lecture Notes in Computer Science, page 115-126. Springer, (2007)

[7] Open Calais http://www.opencalais.com.

[8] Evolutionary Algorithms - http://en.wikipedia.org/wiki/Evolutionary_algorithms.

[9] Wikipedia Database dumps - http://dumps.wikimedia.org/backup-index.html

[10] WP2TXT - http://wp2txt.rubyforge.org/.

[11] Debajyoti Mukhopadhyay, Anirban Kundu, Sukanta Sinha; Introducing Dynamic Ranking on Web-pages based on Multiple Ontology supported Domains; Lecture Notes in Computer Science Series, Springer-Verlag, Germany; February 15-17, 2010; LNCS 5966 pp.104-109

[12] Pepijn R. S. Visser and Valentina A. M. Tamma - An Experience with Ontology Clustering for Information Integration: Intelligent Information Integration, volume 23 of CEUR Workshop Proceedings, (1999)

[13] Wu, C., Potdar, V., Chang, E., 2008. Latent Semantic Analysis – The Dynamics of Semantic Web Service Discovery. In: T. Dillon, Chang, E., R. Meersman, K. Sycara eds. 2008. Advances in Web Semantics I. LNCS. Berlin, Heidelberg: Springer. pp. 346-373. 978-3-540-89783-5.