International Conference on Computational Modeling and Security (CMS 2016)

DSPAA: A Data Sharing Platform with Automated Annotation

Keerthana.I.P\textsuperscript{a,*}, Aby Abahai.T\textsuperscript{b}

\textsuperscript{a}PG Scholar, Dept. of CSE, Mar Athanasius College of Engineering, Kothamangalam-686666, Kerala, India
\textsuperscript{b}Assistant Professor, Dept. of CSE, Mar Athanasius College of Engineering, Kothamangalam-686666, Kerala, India

Abstract

Document annotation and search are two important factors that need to be considered in data sharing platforms. A large unstructured text document contains substantial amount of structured attribute information. Important information is very difficult to find in these documents. Current ad-hoc or predefined annotation of the shared data causes inadequate search, retrieval and analysis capabilities. In this paper we propose a new approach that supports the generation of the structured annotation in the form of attribute name and attribute value pairs from unstructured document. A new data sharing platform DSPAA (Data Sharing Platform with Automated Annotation) is proposed, where the document annotation occurs when the author uploads a document and it is based on a probabilistic framework that considers the attributes in the document content and the query collection. The new system also performs semantic annotation of document using WordNet database. When a user submits a search query, then the system will search for documents in the annotation database and rank the selected documents by using Vector Space model. From experiment results it is clear that the system generates superior results at a rate faster than traditional document retrieval strategies.

© 2016 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Peer-review under responsibility of the Organizing Committee of CMS 2016

Keywords: Content Score; Document Annotation; Document Ranking; DSPAA; Query Score; Semantic Annotation; Vector Space Model; WordNet

1. Introduction

The metadata or annotation is such data, which describes other data or gives information about the data. For example, letters or characters in a text are data, but the number of letters in a text is the metadata. If a narrower sense is implied and the term is used in connection with file types, the metadata means, for example, such information as
the name and the title of a file, its author, keywords to the contents of a file or the date of the saving. An obvious positive effect of metadata in files is: they allow cataloguing and browsing of data according to certain general criteria in a simpler and more precise way.

Data management tools like Microsoft's SharePoint permit users to share documents and tag them for some specific case. SAP NetWeaver permit users to annotate documents, share and do simple keyword based queries. Similarly in Google Base, users can specify their own <attribute name, attribute value> pairs in addition to the ones proposed by system. But, suggested attributes in Google Base are fixed for each category. This fixed or predefined annotation of the shared data causes problems like schema explosion or inefficient data annotation, which in turn guide to unsatisfactory analysis and search performances. A scenario is complicated where the author has to fill a number of fields at time of uploading a particular document. Hence users often avoid such annotations. Such problems results in very simple annotations that is often limited to small keywords. Users are often limited to plain keyword searches, or have access to very basic annotation fields. Such annotations cause the analysis and querying of the data complicated. Annotations which use <attribute name, attribute value> pair requires users to be more principled in their annotation work. Here users must have clear idea in applying and using the attributes.

Data Sharing Platform with Automated Annotation (DSPAA) is a new document sharing platform which performs annotation of document during document upload phase and supports fielded data annotation. This system uses the query collection to direct the annotation process, in addition to inspecting the content of the document. The goal of DSPAA is to create efficiently annotated documents that can be useful for usually issued semi-structured queries of user. This paper is structured as follows: in section 2, the works related to document annotation is discussed. Section 3 presents the implementation details of the proposed system. In section 4 performance analysis of the proposed system is discussed. Finally the section 5 gives the conclusion for the work.

2. Related Works

The information management challenges today stem from organizations based on a large number of heterogeneous, related data sources, but having no way to manage their dataspaces [1] in a convenient, integrated, or principled fashion. They propose dataspaces and their support systems are used for data management. A common problem of database systems is that they are hard to query for users discomfort with a formal query language. To handle this problem in [2],[3] form-based interfaces and keyword search have been proposed, combining the two for creating an approach that provides best result. At query time, a user issues standard keyword search queries, instead of returning tuples, the system returns forms relevant to the search query. The user then creates a structured query by using one of these forms and submits it to the system. With large number of data sources available over the web integration of them is an important problem. Integration of the hidden sources is integration of their query interfaces. An interactive, clustering-based approach to matching query in interfaces is discussed in [4].

There are huge amounts of text on the internet that are neither grammatical nor formally structured. These sources of data, called posts are full of useful information for agents searching the Semantic Web, but they miss the semantic annotation to make them searchable. By leveraging their common attributes, called reference sets, it can annotate these posts despite their lack of grammar and structure [5]. In [6], a method for semantic annotation of web pages is introduced and performed semantic annotation by using web patterns. This method is based on extraction of patterns, which are characteristic for a particular domain. They have annotated pages in a database with regard to patterns so there is information about which patterns are contained on each page.

In [7], highlights the challenges in two scenarios – the Deep Web and Google Base. Traditional data integration techniques are not valid in the case of such heterogeneity and scale. They propose new data integration architecture called PAYGO, which is based on the concept of dataspaces and emphasizes pay-as-you-go data management for achieving web-scale data integration. In [8] describe three recent extraction systems that can be operated on the entire Web. In [9], a tool called KMAD which tells the quality of document or its usefulness based on annotations is presented. Collective sentiments of annotators are classified as negative, positive and objectivity. In [10], introduces the use of Wikipedia for automatic keyword extraction and word sense disambiguation. Given an input document, the system identifies the important concepts in the text and automatically links these concepts to the corresponding Wikipedia pages. In [11],[12] CADS, a Collaborative Adaptive Data Sharing platform, where the information demand of the community is exploited to annotate the data at insertion-time. In [13], describes GoNTogle, a framework for document annotation and retrieval, built on top of Semantic Web and IR technologies. GoNTogle supports ontology-based annotation [14] for documents of several formats, in a fully collaborative environment.
3. Proposed System

A new system is proposed for advanced document annotation and search facilities. The proposed system is Data Sharing Platform with Automated Annotation (DSPAA), a new platform that supports fielded document annotation. The goal is to support the annotation of the documents during the document upload phase. The DSPAA system has two types of actors: producers (authors) and consumers (users). Producers will upload data into DSPAA system by using annotation suggestion forms and consumers search for relevant information by using query search forms. Fig.1 presents the DSPAA architecture. Separate databases are used for storing document collections, query collection and annotations. DSPAA has two phases: insertion phase and query phase. The insertion phase begins with the submission of a new document to be included in the document collection database. After that DSPAA analyzes the text and creates an annotation suggestion form (See Fig.2) with the set of the most probable <attribute name, attribute value> pairs to annotate the new uploaded document. The author can add new attributes, which are not proposed by the annotation suggestion form and submits the filled annotation form. After that the system will perform the semantic annotation of document based on meaning of attributes. The last stage consists of the storage of the metadata in the DSPAA repository. The query phase deals with document search and ranking.

DSPAA has mainly three modules. First module deals with training set formulation using automated techniques. Second module annotates the documents using content score, query score and semantics. Final module deals with querying and ranking documents of several formats. We have created this application using NetBeans IDE using Java for front-end designing and MySQL Server as back end.
3.1 Training set formulation

**Annotation Training.** We used Reuters-21578 Text Categorization Collection for implementing the proposed system. We formed the training set using automatic keyword extraction and categorization from the documents. Formed training set is used for training and judgment of the new system. The automatic annotation can be achieved in two steps. Wikipedia is used for automatic keyword extraction and categorization of the extracted keywords. As the first step, upload an input document into the system. The words after stemming are then added to the selected list and check whether these words appear as a title in a Wikipedia page. The word that satisfies this criterion is selected as an important word in the document. The second step is to categorize or to find the categorization label for the selected words based on the context it appears. The process of resolving the conflicts that arise when a single term is ambiguous, when it refers to more than one topic covered by Wikipedia is known as word sense disambiguation (WSD). Category label for the words are found by using Wikipedia disambiguation API. Assigned the correct category for the word based on the context. This is the procedure used for the annotation training in the form of attribute name (category label), attribute value (word) pair and to form the training set.

**Query Collection Formulation.** We have formed the query collection using popular search queries suggested by Google Trends and Google Auto complete. Google Trends allows us to search into Google's database of searches, to find which keywords are most popular. Google Auto complete tool helps to round out the research by providing keywords as seen through the searchers experience.

3.2 Document annotation

Given a new document D, for which only know its text content, document collection and query collection. We have to find important attributes in D, so that it can be used to help future search queries into the system. For each attribute two important information have to find for identifying and suggesting attributes for a document D: First information is Query Score (AttributeScorequery). The attributes must have high Query Score with respect to the query collection Qcollection. That is, they must appear in many queries in Qcollection. Second, the attributes must have high Content Score (AttributeScorecontent) with respect to document collection Dcollection. That is, they must appear in many documents in Dcollection. Otherwise document D will not be rightly annotated. We combined both scores, using a probabilistic approach i.e., Bayes theorem. Using Bayes theorem attribute score total is the product of AttributeScorequery and AttributeScorecontent. Attributes are displayed to the user in the decreasing order of their scores. Attributes with high scores will be on the top of the DSPAA annotation suggestion form. The score of an attribute can be found using the following equation.
\[
\text{AttributeScore}_{\text{total}} = \text{AttributeScore}_{\text{query}} \times \text{AttributeScore}_{\text{content}} \tag{1}
\]

Query Score can be defined as follows: Query Score of an attribute can be obtained by dividing the probability that the attribute present in the \(Q_{\text{collection}}\) by the probability that the attribute not present in the \(Q_{\text{collection}}\).

\[
\text{AttributeScore}_{\text{query}} = \frac{p(\text{Attribute in } Q_{\text{collection}})}{1 - p(\text{Attribute not in } Q_{\text{collection}})} \tag{2}
\]

where

\[
p(\text{Attribute in } Q_{\text{collection}}) = \frac{\text{COUNT}(Q \text{ Attribute}) + 1}{\text{COUNT}(Q \text{ in } Q_{\text{collection}}) + 1} \tag{3}
\]

\(\text{COUNT}(Q \text{ Attribute})\) is the count of queries in which the attribute present and \(\text{COUNT}(Q \text{ in } Q_{\text{collection}})\) is the total number of queries in the query collection.

Content Score can be defined as follows: Content Score of an attribute can be obtained by dividing the probability that the attribute present in the \(D_{\text{collection}}\) by the probability that the attribute not present in the \(D_{\text{collection}}\).

\[
\text{AttributeScore}_{\text{content}} = \frac{p(\text{Attribute in } D_{\text{collection}})}{p(\text{Attribute not in } D_{\text{collection}})} \tag{4}
\]

where

\[
p(\text{Attribute in } D_{\text{collection}}) = p(\text{Term}_1 | \text{Attribute}) \times p(\text{Term}_2 | \text{Attribute}) \times \ldots \times p(\text{Term}_n | \text{Attribute}) \tag{5}
\]

After uploading a new document the system will perform stemming on its content. \(\text{Term}_i\) is the \(i^{th}\) word obtained after stemming. Here \(p(\text{Term}_i | \text{Attribute})\) is the probability of dependency between word \(\text{Term}_i\) and the attribute.

\[
p(\text{Term}_i | \text{Attribute}) = \frac{\text{COUNT}(D \text{ ANNOTATE(\text{Attribute}) AND CONTENT(\text{Term}_i)}) + 1}{\text{COUNT}(D \text{ ANNOTATE(\text{Attribute}))} + \text{COUNT}(D \text{ in } D_{\text{collection}}) + 1} \tag{6}
\]

\(\text{COUNT}(D \text{ ANNOTATE(\text{Attribute}) AND CONTENT(\text{Term}_i)})\) is the number of documents which contains the word \(\text{Term}_i\) and also annotated with the attribute. \(\text{COUNT}(D \text{ ANNOTATE(\text{Attribute}))}\) is the count of documents that annotated with the attribute. \(\text{COUNT}(D \text{ in } D_{\text{collection}})\) is the number of documents in the document collection \(D_{\text{collection}}\). We applied Laplace smoothing to ignore zero probabilities of attributes that do not appear in query collection and document collection.

\[
p(\text{Term}_i | \text{NOT(\text{Attribute})}) = \frac{\text{COUNT}(D \text{ NOT ANNOTATE(\text{Attribute}) AND CONTENT(\text{Term}_i)}) + 1}{\text{COUNT}(D \text{ NOT ANNOTATE(\text{Attribute}))} + \text{COUNT}(D \text{ in } D_{\text{collection}}) + 1} \tag{8}
\]

\(\text{COUNT}(D \text{ NOT ANNOTATE(\text{Attribute}) AND CONTENT(\text{Term}_i)})\) is the number of documents which contains the word \(\text{Term}_i\) and not annotated with the attribute. \(\text{COUNT}(D \text{ NOT ANNOTATE(\text{Attribute}))}\) is the number of documents that not annotated with the attribute.

For combining Query Score and Content Score a 2-Phase Algorithm is used. In first phase it extracts the important words as attributes. Second phase will calculate the score of selected attributes using Bayes Equation and return them in an annotation suggestion form in the decreasing order of their scores. Consider a sample document collection, query collection and annotations shown in Table 1 and Table 2 respectively. The document collection contains 5 documents D1, D2, D3, D4, and D5. Query collection contains 7 queries. The new document that we are going to upload to the system is D6. We have to find the important annotations for document D6.

\{D6: coconut agency at Philippine\}

After stemming the words we get from the document D6 are:

\(\text{Term}_1: \text{coconut, Term}_2: \text{agency, Term}_3: \text{Philippine}\)

The system will find the total score for each of these words using equation (1). We are going to see how to find the score for the word coconut. Here coconut is attribute value and the corresponding attribute name is Product which is obtained from document annotations.

\[
\text{Attribute}_{\text{ProductScore}}_{\text{total}} = \text{Attribute}_{\text{ProductScore}}_{\text{query}} \times \text{Attribute}_{\text{ProductScore}}_{\text{content}} \tag{9}
\]

\[
\text{Attribute}_{\text{ProductScore}}_{\text{query}} = \frac{0.875}{1-0.875} = 7 \tag{10}
\]
\[
\text{Attribute}_{\text{Product}}\text{Score}_{\text{content}} = \frac{3 \cdot 1 \cdot 2}{1 + 1 + 3 + 2 + 1 + 6} = 0.973
\] 
\[
\text{Attribute}_{\text{Product}}\text{Score}_{\text{total}} = 7 \times 0.973 = 6.81
\]

Table 1: Document collection.

| Document Name | Document Content | Document Annotations |
|---------------|------------------|----------------------|
| D1            | Philippine coconut product exports fall in January | Place: Philippine, Product: coconut, Month: January |
| D2            | SriLanka to upgrade quality of coconut products | Place: SriLanka, Product: coconut |
| D3            | U.K potato futures volume down in February | Place: U.K, Product: potato, Month: February |
| D4            | Pakistan cotton crop seen record 7.6 mln | Place: Pakistan, Product: cotton, Crop: 7.6 mln |
| D5            | Singapore petroleum revises petrol pump prices | Place: Singapore, Product: petroleum, revises: petrol pump prices |

Table 2: Query collection.

| Query | Query Terms |
|-------|-------------|
| Q1    | Place: U.K, Product: potato |
| Q2    | Place: SriLanka, Product: coconut |
| Q3    | Place: Philippine, Month: January |
| Q4    | Product: coconut, Month: January |
| Q5    | Place: Brazil, Product: coconut |
| Q6    | Place: Singapore, Product: petroleum |
| Q7    | Product: cotton, Month: March |

Finally the system will suggest the annotations for document D6 in an annotation suggestion form, where the form includes these words as attribute value and their labels as the corresponding attribute names. These attributes will be represented in the form in the decreasing order of their scores.

**ALGORITHM 1.** 2-Phase Algorithm

**Input:** Document D to upload.  
**Output:** Suggested attributes \( A_i \) in the document D.

**Phase-1**
1. **Upload** the document D;
2. **Extract** text in document and perform stemming;
3. **for** each word extracted after stemming **do**
   4. **Check** whether the word appears as title in a Wikipedia page;
   5. **if** word appears as a title in a Wikipedia page
   6. **Mark** word as important keyword and add it to selected keywords list \( S \);
   7. **else**
   8. **Mark** word as unimportant and discard keyword;
9. **end for**

**Phase-2**
1. **for** each keyword in \( S \) **do**
2. **Calculate** score of keyword using Bayes Equation;
3. **end for**
4. **Arrange** words in \( S \) in decreasing order of their scores;
5. **return** list of keywords as important attributes in annotation suggestion form;

**3.3 Semantic annotation**

It is a method which helps to add semantics that is meaning and relationship information to unstructured and semi-structured document text. This meaning of the words is stored in the annotation database. Semantic annotations deal with attaching synonyms and meanings with data. It will help in providing better search functionalities and
helps users to search for information not only based on the keyword based search, but also using general concepts that describe the needed information. For mapping each important attributes with its synonyms and related words we have to perform semantic annotation. We have implemented semantic annotation using WordNet 3.0 [15] and Java API for WordNet Searching (JAWS). A word can have more than one synonym. But assigning correct synonyms for a word is a challenge. Correct synonym for a word in a text is obtained by using the algorithm Semantic Annotation.

**ALGORITHM 2. Semantic Annotation**

**Input:** Document D, Suggested attributes A<sub>i</sub> in document D  
**Output:** Selected synonyms for suggested attributes A<sub>i</sub> in document D 

1. \( N \text{words} = \text{number of words in document text;} \)
2. for each word in annotation do 
3. for each synonym \( \text{synonym}(\text{word}) \) do 
4. initialize \( \text{score}[\text{synm}] \) to 0; 
5. end for 
6. end for 
7. for each word in annotation do 
8. \( L = N \text{words} - 1 \text{ other words in document text;} \)
9. for each term in \( L \) do 
10. for each synonym \( \text{synonym}(\text{term}) \) then 
11. if synonym contains \( \text{synonym}(\text{term}) \) then \( \text{score}[\text{synm}] ++; \)
12. end if 
13. end for 
14. end for 
15. end for 
16. end for 
17. for each word in annotation do 
18. find average score of \( \text{synonym}(\text{word}); \)
19. result = set of \( \text{synonym}(\text{word}) \) with \( \text{score}[\text{synm}] \geq \) average score; 
20. end for 
21. return result; 

3.4 Document search

This phase is dedicated for users who want to search the relevant documents within the document collection and then retrieve them in the decreasing order of their rank relative to the query. Query can submit in two ways. First one is the normal searching method similar to we use in search engines which uses a text box to type the query and this method is called as search by content. Second method is called as search by value. Here the user can specify the search query in \( \langle \text{attribute name, attribute value} \rangle \) pairs. Query submission using search by value method is shown in Fig.3. After submitting the search query the system will split the query terms and then search them within the annotation database. Documents that contain the search words are selected from the document collection in this manner. After that the system will rank the selected documents based on Vector Space model.

Vector Space model helps to represent the documents as a vector of terms. Term frequency \( \text{freq}(d,t) \) is the number of occurrences of term \( t \) in the document \( d \). The term-frequency matrix \( T F(d,t) \) is a matrix which calculates the connection of a term \( t \) with respect to the document \( d \). It is defined as 0 if the document does not contain the term, and nonzero value otherwise. Set \( T F(d,t) = 1 \) if the term \( t \) appears in the document \( d \). Inverse document frequency (IDF) represents the importance of a term \( t \). It is the number of documents in which the term appears. TF and IDF are combined together and form the TF-IDF measure:

\[
\text{TF-IDF}(d,t) = \text{TF}(d,t) \times \text{IDF}(t)
\]

Similarity of a document vector to a query vector = cosine of the angle between them

We know that \( x, y = |x| \times |y| \cos \theta \). From this similarity score can be written as

\[
\text{Sim}(D_1, q) = \cos \theta = \frac{D_{p\times q}}{p|q|} = \frac{\sum_{i=1}^{n} W_{1i} \times W_{iq}}{\sqrt{\sum_{i=1}^{n} W_{1i}^2} \times \sqrt{\sum_{i=1}^{n} W_{iq}^2}}
\]

where \( w_{ij} \) is a weight for term \( i \) in document \( j \) and \( w_{ij} \) is a weight for the term \( i \) in query \( q \). The system will rank documents by using the decreasing order of cosine value. Fig.4 shows search results after the ranking process. Ranking process can be summarized as given in the ranking algorithm.
Algorithm 3. Document Ranking

**Input:** Search query.

**Output:** Selected documents that match with the query.

1. Split query into component terms and search them within the annotation database;
2. Find related documents and add them to the selected documents list \( L \);
3. Find all distinct words in selected documents and add them into list \( T \);
4. for each document \( d \) in \( L \) do
5. for each term \( t \) in \( T \) do
6. find freq\((d,t)\);
7. end for
8. end for
9. Form term-frequency matrix TF\((d,t)\);
10. Find IDTF\((t)\) for all terms in \( T \);
11. Calculate TF-IDF\((d,t) = \text{TF}(d,t) \times \text{IDF}(t)\);
12. Calculate tf-idf vector for the query;
13. Compute score of each document relative to query using cosine similarity;
14. Rank documents in the decreasing order of the similarity value;

### 4 Results

We have evaluated the performance of DSPAA based on its precision, recall and query execution time. The total number of relevant documents retrieved by executing ten different search queries on the proposed DSPAA which uses annotation and those in the old systems without annotation is recorded. Based on the observation precision, recall and execution time values obtained are given in Table 3.

| Query Number | Search using Annotation | Search without using Annotation |
|--------------|-------------------------|--------------------------------|
|              | Precision | Recall | Execution Time (ms) | Precision | Recall | Execution Time (ms) |
| 1            | 0.90      | 1.0    | 152               | 0.75      | 0.82   | 266               |
| 2            | 0.87      | 1.0    | 92                | 0.60      | 0.84   | 237               |
| 3            | 0.88      | 0.93   | 315               | 0.83      | 0.68   | 329               |
| 4            | 0.93      | 0.97   | 116               | 0.70      | 0.80   | 187               |
| 5            | 0.92      | 0.95   | 146               | 0.66      | 0.78   | 186               |
| 6            | 0.95      | 0.94   | 133               | 0.73      | 0.84   | 256               |
| 7            | 0.86      | 1.0    | 168               | 0.84      | 0.75   | 245               |
| 8            | 0.89      | 0.98   | 164               | 0.86      | 0.79   | 190               |
| 9            | 0.93      | 0.92   | 243               | 0.80      | 0.89   | 296               |
| 10           | 0.91      | 1.0    | 79                | 0.63      | 0.70   | 237               |
| Average      | 0.904     | 0.969  | 160.8             | 0.740     | 0.789  | 242.9             |

Table 3: Document retrieval.
Fig. 5: Precision.
Fig. 6: Recall.
Fig. 7: Execution Time.

The Fig. 5, Fig. 6 and Fig. 7 show the comparison between the precision, recall and execution time of two methods respectively. From the results we can see that search using annotation has a higher precision and recall value when compared to search without using annotation. We can also conclude that DSPAA effectively reduces the execution time of search queries and thus generates results at a faster rate. So we conclude that the proposed DSPAA is more efficient in relevant document retrieval and its performance is better than the performance of the current systems.

5 Conclusion

Annotation is one of the best featured techniques to manage documents and get effective search result. <attribute name, attribute value> pairs are generally more meaningful and significant as they can contain more information than untagged approaches. For such maintenance of annotated documents user has to take extra efforts. Data Sharing Platform with Automated Annotation (DSPAA) is used to suggest relevant attributes to annotate a document, for supporting future search queries. DSPAA is a data sharing platform where the annotation of documents is done by using a probabilistic approach. By using this technique it can greatly improves the annotation process and increase the utility of shared data. In order to overcome the drawbacks of keyword-based search this framework supports both keywords and semantic based search. The proposed system reduces user effort, supports both automated and manual annotations, efficient to handle synonymous terms, supports unstructured, structured and semi structured document annotation and improves the precision and recall of document retrieval.

References
1. Michael Franklin, Alon Halevy, David Maier, From Databases To Dataspaces: A New Abstraction For Information Management, ACM Sigmod Record, December 2005.
2. E.Chu, A.Baid, X.Chai, A. Doan, J. Naughton, Combining Keyword Search and Forms for Ad Hoc Querying of Databases, SIGMOD-09.
3. Magesh Jayapandian and H. V. Jagadish , Automated Creation of a Forms-based Database Query Interface, In VLDB 2008.
4. Wensheng Wu, Clement Yu, An interactive clustering-based approach to integrating source query interfaces on the deep web, SIGMOD 2004.
5. Matthew Michelson ,Craig A. Knoblock, Semantic Annotation Of Unstructured And Ungrammatical Text, In IJCAI, 2005.
6. Milo Kudelka, Vaclav Snasel, Ondrej Lehecka,Eyas El-Qawasmeh, Jaroslav Pokorny, Semantic Annotation Of Web Pages Using Web Patterns, Advanced Internet Based Systems and Applications Lecture Notes in Computer Science , Volume 4879, 2009, pp 280-291.
7. JayantMadhavan, Shaw R. Jeffery, Shirley Cohen, Xin (Luna) Dong, David Ko, Cong Yu, Webscale Data Integration: You Can Only Afford To Pay As You Go, Proceedings of the Third Biennial Conference on Innovative Data Systems Research, (2007).
8. M.J. Cafarella, J. Madhavan, and A. Halevy, Web-scale extraction of structured data, SIGMOD Rec., vol. 37, pp. 55–61, March 2008.
9. Archana Shukla, Sentiment Analysis of Document Based On Annotation, International Journal of Web & Semantic Technology (IJWeST) Vol.2, No.4, October 2011.
10. Rada Mihalcea, Andras Csomai, Wikify! Linking Documents to Encyclopedic Knowledge, Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, Pages 233-242, 2007.
11. Vagelis Hristidis, Eduardo Ruiz, CADS: A Collaborative Adaptive Data Sharing Platform, VLDB Workshop on Personalized Access, Profile Management, and Context Awareness in Databases (PersDB 2009).
12. Eduardo J. Ruiz, Vagelis Hristidis, Panagiotis G.Ipeirotis , Facilitating Document Annotation using Content and Querying Value, IEEE Transactions On Knowledge And Data Engineering, Vol. 26, No. 2, February 2014.
13. GiorgosGiannopoulos, NikosBikakis, Theodore Dalamagas,Timos Sellis, GoNTogle: A Tool for Semantic Annotation and Search, In Proc. of ESWC 2010.
14. NikosBikakis, Giorgos Giannopoulos, Theodore Dalamagas, Timos Sellis, Integrating Keywords and Semantics on Document Annotation and Search, Proceedings of the 2010 international conference on On the move to meaningful internet systems: Part II, page 921-938, Berlin, Heidelberg, Springer-Verlag,2010.
15. GeorgeA. Miller, WordNet:A Lexical Database for English, Communications of the ACM November 1995/Vol. 38, No. 11.