http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 13 (2): 217-230, 2014
ISSN 1812-5638 / DOI: 10.3923/itj.2014.217.230

Document Based Semantic CMS in Cloud

R. Priyadarshini and Latha Tamilselvan
Department of Information Technology, B.S. Abdur Rahman University,
Seethakathi Estate, Vandalur, Chennai, India

Abstract: Cloud computing allows accessing resources across Internet, without intervention of an expertise and without a control over the underlying infrastructure. Content Management System (CMS) is one of the domains identified in Software as a Service (SaaS) in cloud computing. The amount of digital content is increasing at accelerating speed and CMS is a way of organizing such content. The content organized is reproduced in desired pattern or format by providing procedures to manage work flow. Document databases are used to manage and organize chunks of data, build indexes and cross link the documents and comments etc. The content referred in the net through any website will tend to change over time. In order to manage these contents online document based CMS with automatic URL indexing is very much essential. It's highly possible that the content in CMS will be redundant over the web as most of the time the content will be gathered from already existing websites. Back tracking the source of such content will become obsolete and also changes to the source are difficult to be tracked. In this study it is proposed to reinvent the document based CMS in cloud. The proposed document based CMS varies from these traditional CMS in architecture in storage and control flow. Source URLs and content markings are indexed and mirrored. In this study the URLs and content marked in the source websites are stored in the database. The source content and URLs are stored separately, the marked content in the user blog is highlighted and semantically processed with a NLP tool. The marked content is semantically matched with concept matching and named entity recognition technique. The semantically annotated content are located in the stored websites and matched with the original source websites. This study comprises of initial work of implementing annotation technique and location of the marked content in the document. The results are evaluated with the metrics like recall and precision techniques and appropriate graphs were plotted.

Key words: CMS, NOSQL, cloud, SAAS, application of SAAS, semantic web

INTRODUCTION

Cloud computing is an emerging technology which has following characteristics like scalability, reliability and availability of resources across the globe. There are three service models based on the cloud, among which SAAS is an on demand software delivery model. Moreover, Cloud computing is good solution for processing content in cloud environment. Based on the above facts, the content management system is identified as one of the applications of SAAS in Cloud (Tyagi *et al*., 2012; Diaz-Sanchez *et al*., 2011). The arrival of cloud computing and deployment of software as service is creating different methods for handling data. The cloud based databases are ever increasing size of data sets. Since the conventional database services are too much costly for execution, numerous cloud database services run in clusters in several computers across internet. These cloud databases are able to serve data which are scalable from gigabytes to hundreds of terabytes and petabytes. The conventional relational database servers expand horizontally whereas Cloud Databases give less execution time and it is elastic in nature along with better performance and latency time.

The cloud databases are required only for the data which expands enormously and increases its size to great extent. Scalable NoSQL databases are suitable for the cloud. It includes MongoDb, Couch DB and Riak. NoSQL based CMS not only manages huge repository of data, also has high availability and scalability (Padhy *et al*., 2011). CMS is used in most of the professional industries in order to organize and manage type of digital information like pictures, audio and video documents etc. These type of systems are featured by the need for effective storage and reading the huge volume of content. Strict rules are incorporated for controlling and sharing of information's among the users and organisations. Nowadays huge number of web based

Corresponding Author: R. Priyadarshini, Department of Information Technology, B.S. Abdur Rahman University, Seethakathi Estate, Vandalur, Chennai, India

Content Management Systems are in use. This kind of CMS compensates the explosion of web content in blogs, social networks etc. The problem in web is that it is very difficult to find relevant information in World Wide Web. To address this problem "semantic web" has been proposed as the extension of current web (Diaz-Sanchez *et al*., 2011).

The aim of the proposed study is to back track the location of referred content in the internet dynamically in a cloud environment. The study also clearly indicates the problems in cloud regarding location of resources and scalability issue with semantic solutions.

SEMANTIC WEB

The semantic web plays important role in emerging web which helps the users to explore, interact and collaborate with each other easily. Linked data is one of the key manifestations of the semantic web; it collects the URI linked data resources on the web. The connection between the resources has potential of identifying and extracting relevant information in different documents (Aksac *et al*., 2012). The unstructured information like the terms in the text document is connected to linked data resources which enhance information retrieval and knowledge discovery (Auer *et al*., 2007; Mendes *et al*., 2011). Semantic annotation is one of the key areas in the semantic web along with this there are two important tasks which can be performed such as (1) Extracting and hyper linking named entities and (2) Finding relevant documents in accordance with the entities (Kiryakov *et al*., 2004). Extracting entities in a web document and finding relevant information through linked data domains is one of the main concerns in semantic annotation.

All the existing browsers has the option of bookmarking the sites which can be used for future reference. Whenever these websites are used for the creation of a new web document, the referred sites can be bookmarked but the content in the document is not exactly marked for the corresponding sites. The proposed system automates this work semantically for backtracking the originality of the content and to ease the work of creating newer web documents in cloud.

This study presents the proposed solution for automatic backtracking of the content available in the web. As the face of the web has changed to great extent over the decade and the contents in web tend to change from time to time, the content might either exist, vanish or modified. The web has also become more semantic in terms of searching the content. The proposed system handles the problem of identifying the modifications in the existing content and also to monitor whether the source content exist or vanish after certain period of time. The source content is linked along with content in new web document and stored in the database which is scalable; this will help the user in great extent for their future references and also for updating of the existing content from the source web pages. This system will alert the user about the modifications, existence and transition. The content from corresponding source is exactly matched with the part of new content in the web document. In short the proposed system automates semantically the bookmarked sites which are used for creation of new documents online.

PREVIOUS RELATED WORK

Zhang (2011) in his study he systematically introduces main features, principles, properties in MVC (Model, View, Controller) in a CMS which is exclusively designed for the native people in china to use. It provides an efficient tool to the webpage-makers. This CMS separates form from the Web page content, so it becomes easy for the search engine to index. The CMS is applicable only for the native china mainland the advanced application is left for research. The database used is not scalable; it's only the conventional Relational Database management system (Fazzinga *et al*., 2011). Dobecki and Zabierowski (2010) in their study describes how to design content management system using the newest web-based techniques, Model, View and Control architecture of the web application was analyzed and Five layer model for web CMS have been proposed. Existing frameworks like Spring and Hibernate were used for all the five layers in the web based model and their advantages are stated clearly which makes the development process much easier. All the frameworks and plugins being used to improve the modularity slow down the performance of the application when moderate load applied (Dobecki and Zabierowski, 2010).

Sung Ho Jang, Chang Hyeon Noh, Jong Sik Lee, proposed semantic web based digital content management system . In the system, a semantic web agent analyses user requirements and infers a proper digital content from the digital content ontology. They have

dealt with the flaw in grid computing based digital content management systems for searching digital contents from relational database systems by few keywords is overcome by ontology based query language. Although they proposed the idea of semantic web based content management system, the performance of latter with automated reasoning and semi-automated annotation is left for future research (Schaffert *et al*., 2008; Bolettieri *et al*., 2007). Ismail and Joy (2011) have proposed an application to identify the similarities in a domain using semantic approach. Their work focused on the existing approaches for describing semantic relationships in Content Management Systems and how metadata capture the pedagogic information which can be applied to the semantic information stored in CMS. Novel approach to capture relationship between tagged metadata for learning objects stored from repository has been described clearly. But Back tracking of Source URL or the Origin of the content in a domain is not addressed.

Padhy *et al*. (2011) discussed the advantages of NOSQL over RDBMS by elaborating the features of NOSQL and compared the features of various NOSQL Databases. This huge amount of data storage does not require schema structure. There is no join operation and it's horizontally scalable. NOSQL databases generally process data faster than relational databases. NOSQL databases like Mongo DB, Couch DB, Cassandra and Big Table are compared. Scaling up using Relational Database Management Systems (RDBMS) is quite expensive as it has to be done across multiple servers. Performance of RDBMS will be poor and slow to handle massive amount of data. In RDBMS when data does not fit in to table; the database structure will become complex. Leavitt (2010) elaborated the pros and cons NOSQL. The storage of data does not require a schema structure and there are no join operations among the tables. NOSQL databases generally execute the query faster than relational databases. According to O'Grady users will make use of NOSQL when there is a need of storing huge amount of data or when storage is scalable. Scaling up using Relational Database Management Systems (RDBMS) is quite expensive as it has to be done across multiple servers. Performance of RDBMS will be poor and slow to handle massive amount of data. In RDBMS when data does not fit in to table, the database structure will become complex. Aksac *et al*. (2012) defined and presented a novel semantic web browser called person which is short for personal semantic extension. The semantic layer provided by person not only works on web pages, but also on RSS data of user choice. In this study, they proposed a new solution for semantic web browsing. The working of the new approach is clearly indicated in their study. They surveyed the related systems and architectures and compared them with person's features. They also conducted a usability analysis on the developed tool to show that the users are mostly satisfied with the performance of the semantic web browser.

From the above literature survey it is inferred that backtracking of source URL and monitoring the source content semantically was not concentrated. So it is proposed to back track the source content of the marked paragraphs in the newly created weblogs. In the proposed system the content in the newly created web document is semantically annotated with the source document. Also content in the source website and the corresponding URL is accordingly stored in the database repository. The marked content in the original source is compared with the live source content in the web dynamically. If there is a significant change then it is indicated to the user by a trigger or alert. The main advantage of the proposed system is semantic linkage of new content with the source which can be easily used for future reference. It also preserves the originality of the references marked that will be used for updation of existing content and for future reference.

SYSTEM DESIGN

The proposed system is classified in to two main phases. The first phase of the framework is the document based Semantic CMS and second phase will be the deployment of designed CMS in cloud which will be optimized with scalability. Figure 1 shows the first phase of the proposed system. The framework consists of four main modules such as (1) Content Marker, (2) Rendering Engine, (3) Change Tracker (Monitoring System) and (4) User Interface.

In the proposed system framework a web based user interface is designed to create and store the new web document by referring different existing websites. The user creates a new web document by exploring the relevant websites in a particular domain or field say for example ten different source websites in the news sites like CNN, BBC, etc. The content in the new web document is partitioned corresponding to those websites with its URL. In order to track and locate these contents from the source, content marker and change tracker are designed. The content marker marks the content in the new web document using semantic annotation. It will also locate the content by matching with the source websites. The rendering engine is designed to maintain the stack of virtual documents with its corresponding source websites and URLS in a repository. Rendering engine will also render the meta data of the marked new web document to

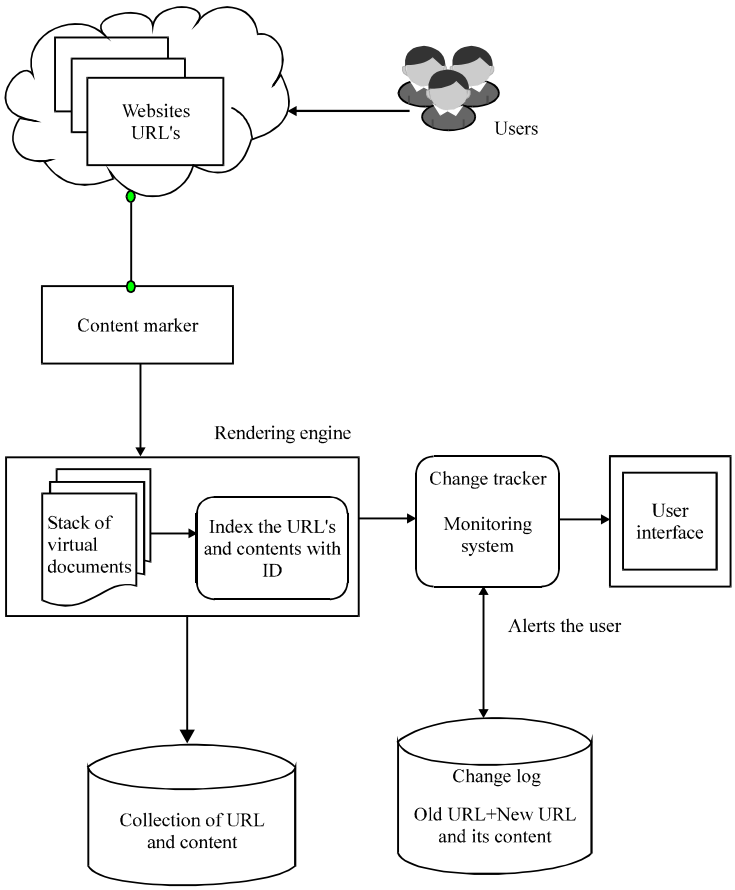


Fig. 1: Illustration of identifying source URL using content marker and change tracker

the change tracker. The change tracker will track the source website of the specified marked content. It also checks whether content is modified, updated, deleted or transited to some other URL. If there is more than one source matched with the marked content then the change tracker locates the source web document based on the timestamp. The user interface is designed in such a way that user should be able to create a new web document and mark it semantically. The semantic marking is done by annotating the entities so that matching and location of content will be more meaningful and accurate. The annotated content will be located in the interface which is redundantly stored in the repository. In the second phase of the proposed design the application as whole is deployed in cloud based platform. If the number of users increases then proportionally the number of virtual documents stored will increase. In order to manage the increasing demand, scalable database will be deployed which will very well suit the cloud environment. The optimization and performance will be evaluated with scalability as main crteria. Figure 2 shows the detailed design of the proposed system with the rendering engine. Virtual documents are stored in form of chunks. The chunks consist of ID, URL, content and indexed content. The virtual documents are rendered in the form of metadata to the change tracker. It will track the source website and compare the meta data with the source website and monitor the changes. It will also store the modified new content in the change log. The user will be given an alert if there is any significant change. The following sub phases are identified in the second phase of the design.

Data Partitioning-Clustering the virtual documents semantically:

- Identifying replicas using Low Latency
- Fault Tolerant (LLFT) algorithm for cloud environment
- Load balancing is done by automatic migration in database level

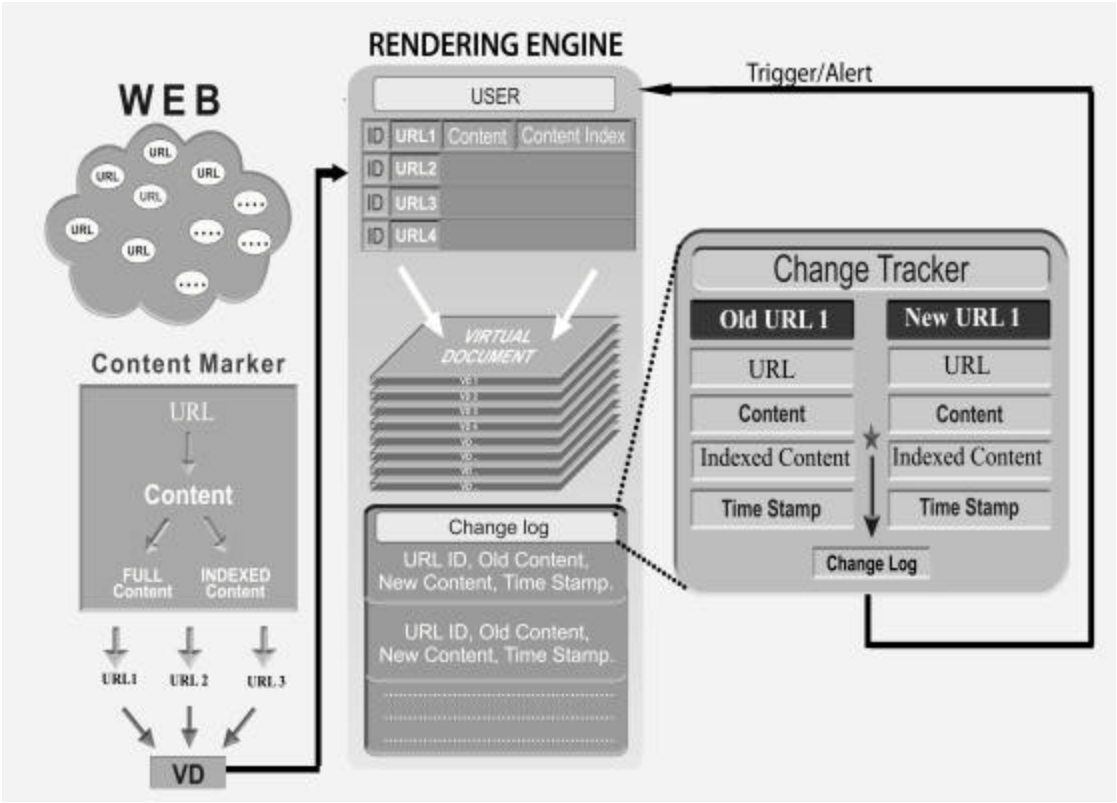


Fig. 2: Methodological working mechanism for triggering alert by rendering engine

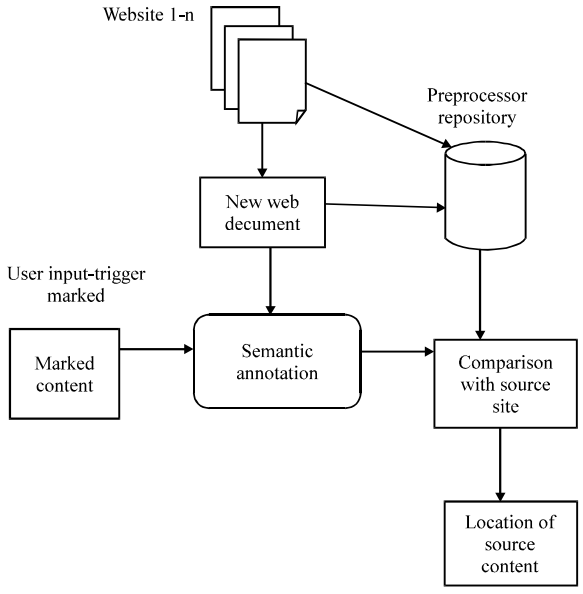


Fig. 3: Location of source content using semantic annotation technique

METHODOLOGY

Content marker: Content marking is done with the help of the semantic annotation; the text in the newly created web document is highlighted and then annotated semantically along with the URL. Figure 3 depicts that creation of a new webpage from multiple source websites by giving their content and, respective URL's as input. All the source websites (URL with the content) are stored in a database. The source websites are pre-processed to clean the additional information (styles, advertisements, images, etc.,) and also be stored in the database. The Entities present in the newly created webpage and the relationship between those entities are identified for making the webpage semantic using an annotation tool.

When a user searches the source for a highlighted portion of text in the newly created web page, the URL with the marked content is identified using a pattern/string matching algorithm. The algorithm performs the comparison of the new web page with the cleaned source web pages in database. The source content in the returned URL are located using the proposed "Content Location" algorithm. The identified location of the

\content in the source web page is finally displayed to the user. The Steps involved in the content marker are as follows: (1) Creation of new web document (2) Pre-processing and storing (3) Entity relationship identification (4) Comparison using pattern matching and (5) Location of the source content.

The marked content and corresponding source website of the content is stored along with ID, whenever it is necessary to store multiple documents with multiple users the database should be expandable in nature. The steps to proceed with the content marker are given in algorithm.

Algorithm 1:

Input: Web pages with the URL
Output: Location of source content for the highlighted text.
Variables: **"Ccontent"** is marked indexed content in the new web document.
Begin
for each "new_webdocument" **do**
if "sourceURL!=NULL" **then**
Get the URLs
Parse the URLs
for each "URL" **do**
Parse the webpage into HTML document
// use of jsoup html parser
for each "HTML document" **do**
// apply pre-processing steps
Remove all navigation links
Remove all cascaded style sheets
Remove all images
end for
Return "text document"
// pre-processed metadata from the html document
end for
Store "text document" in database
//Relational database is used
end if
Var Ccontent = user marked content in new_webdocument.
if "Ccontent " in "new_webdocument" is marked **then**
while "Ccontent != NULL" do
Annotate the content and mark semantically
// Alchemy API is used.
Compare "Ccontent" with all the "text documents" stored in the database
end while
Return "corresponding URL of the text document";
end if;
if annotated content marked **then**
Display marked "Ccontent" with URL
for each "returned URL" **do**
while "returned URL !=NULL" do
Match the "Ccontent" with the corresponding "text document"
Apply pattern matching and concept matching logic // semantic matching
end while
Return "location of "Ccontent" in text document";
end for
for each "returned location in text document" **do**
while "returned location !=NULL" **do**
Get corresponding URL of "text document" ;
Open the corresponding webpage for URL;
Identify the "returned location" in the URL;
end while
Return "identified location in the URL";
end for
end if
end for
End

The results of the identification of entities, its storage in database semantic annotation are shown in the results section (A) using named entity recognition.

Rendering engine: The rendering engine composes of the stack of documents and the corresponding URL'S and source contents with the corresponding identification numbers. Figure 4 shows the number of virtual documents rendered by the custom mark-up language file. When the number of document grows enormously the rendering engine maintains the stack accordingly with the contents by following steps:

Algorithm 2:

Input: The content chunks from the content marker with the index and URL.
Output: Virtual document
Begin
for each content chunks **do**
Assign ID with the corresponding URL
// Output from the content marker is indexed.
end for
if content in the web document is marked **then**
Render the virtual document
Display the highlighted portion to the client
else
Check the virtual document format
Check the attributes and content references
Mark the contents
Render the mark-up file as output.
end if
End

As the marked content in the new web page is semantically annotated and compared and the content in the virtual document will be rendered by the custom mark-up language. This content in turn will be directed to the source website for tracking and monitoring purpose.

Change tracker/Monitoring system: The change tracker in the proposed system will track the source websites of the content stored in the new web document. It also detects the changes by comparing the meta data of the located content in new web document and source website. The updation of the changes in the websites will be done by Really Simple Syndication (RSS) feeds which are used to publish frequently modified documents in a standard format. It also serves the users with the updated information.

Basically an RSS document includes metadata about a newly created web document and source document that has a changed content. RSS feeds can be read by tools called aggregators. With aggregators, users can subscribe news channels and create their own newspapers that include articles form different sources (Zhong *et al*., 2007). There are many RSS readers available on the web with the advanced configuration capabilities. Some of them are also developed as extensions of web browsers. The

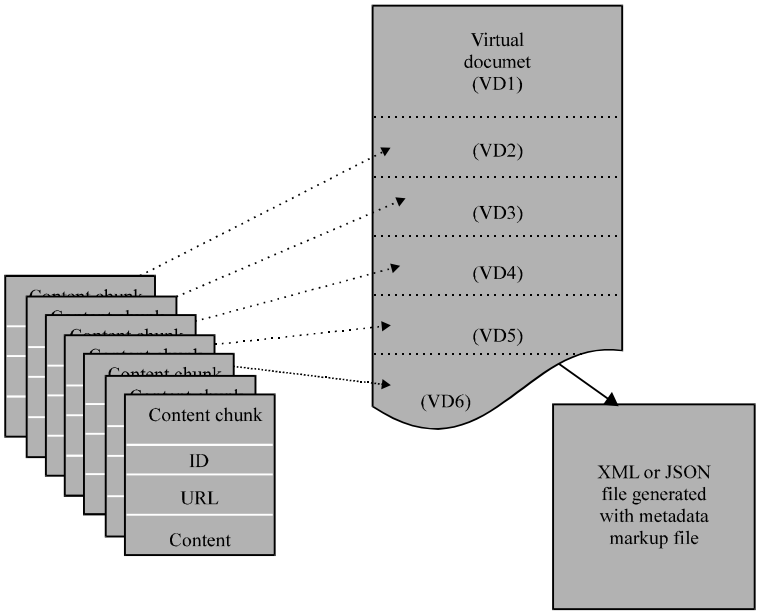


Fig. 4: Stack of virtual documents in the rendering engine

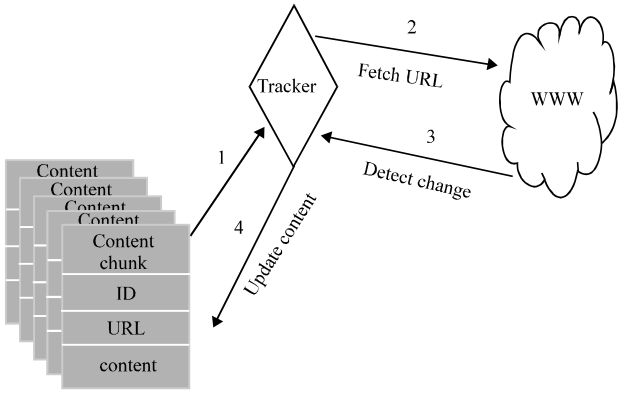


Fig. 5: Work flow of change tracker for fetching the URL

change tracker will first identify whether the source website has RSS feed reader or not, then accordingly it has to fetch the data through the script written manually or through the aggregators. Figure 5 shows the step by step procedure for the monitoring or tracking the website online either through aggregators or by algorithm based manual scripts. The change tracker or the monitoring system involves source sites in web, the content in the specified URL's would have been modified, vanished or moved to some other URL. By detecting these changes the user will be updating himself with referred content for future use.

The Steps involved in Change tracker are as given below in algorithm 3.

Algorithm 3:

Input: The content chunks from the content marker with the index and URL.
Output: Virtual document
Begin
for each virtual documents (VD) **do**
Fetch the content and URL from appropriate websites
if RSS (Really Simple Syndication) feed is enabled **then**
Get the metadata from the website
Compare the metadata of the content along with the metadata of the source site
else if
Construct script to get the metadata from the site
Detect the significant changes
end if
Update the Content in the virtual documents
Apply the changes
Alert a trigger to the user
End

Cloud deployment as SaaS: In the proposed system, the application will be deployed as SaaS in the private cloud environment. The practical example for Saas is Customer relationship management "salesforce.com". In the proposed study the data stored in the repository will be in form of node which comprises of URL, Index, Time Stamp and Content. In this case the link and content are bound together so the content will be a legitimate and genuine. For example if a new web document is created based on five or more sites, the index and data of all the sites will be stored sequentially. Index will be relevant to the URL and the data will be the content grabbed from site along with the images and pictures in the source website. When the data starts growing in the application, say there are 100 users who has 1000 MB of documents each and accordingly if the data increases to 1 Terabytes it is highly impossible for a Relational database systems to handle the data which is growing rapidly, the solution for this problem is the NOSQL databases which will be easily scalable and deployable in cloud.

In NOSQL databases the data can be stored in the form of chunks and data can grow horizontally. It is proposed to use Mongo DB for storing the scalable data in the framework. The repository comprises of number of users like U_1, U_2, U_3, …,U_n, Let the total number of virtual documents be VD_1, VD_2, VD_3, ...,Vd_n for one user U_1 and then Virtual documents VD will comprise of individual nodes such as $node_1$, $node_2$, $node_3$, ..., $node_n$. The Virtual documents are the new documents created by the user by referring number of URL's like URL_1, URL_2, etc. Therefore for each user there will be many virtual documents and in turn individual nodes and corresponding timestamps. The node comprises of the elements like Index, timestamp which varies according to the source websites referred. The performance evaluation will be done for the proposed system by introducing more number of users and multiple documents with n number of nodes. In brief the storage pattern can be easily represented as shown in the following equations:

$$U_1 = VD_1 + VD_2 + VD_3 + \ldots\ldots VD_n \quad (1)$$

$$VD_1 = \sum_{i=1}^{n} node_1 \quad (2)$$

The storage pattern of the URL and content for multiple documents is illustrated in Fig. 6. Scalability is one of the key features in the cloud environment, as there is rapid increase in data, a test bed will be created for the proposed s ystem which comprises of scalable database like Mongo DB and experiments will be conducted and optimized by the proposed algorithm 4.

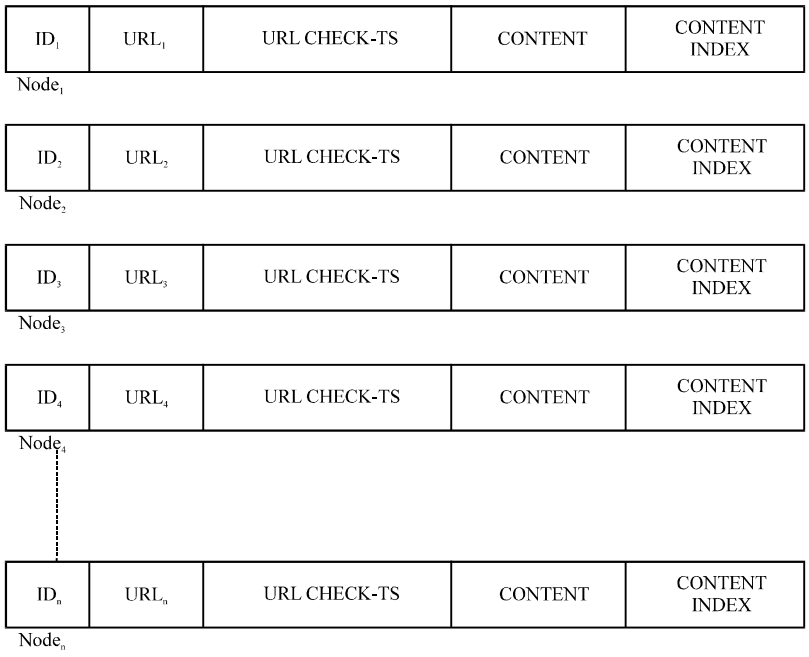


Fig. 6: Storage pattern of URL and content

Algorithm 4:

Input: The nodes from the content chunks in document.
There are three steps in the algorithm:

Step 1: Partitioning of Virtual Documents (VD)
- Storage of chunks in form of nodes
- Each node consists of the ID, URL, Timestamp, content and Indexed content
- Each and every part of the content in the document is partitioned and indexed with ID and corresponding URL according to the domain

Step 2: Clustering of Nodes to a particular domain
- Select nodes according to a particular domain
- Cluster the nodes of same domain based on ID
- Nodes of different clusters are stored with different index

Step 3: Semantic Data Partitioning
- The documents in the clusters are semantically analysed and processed
- Based on the meaningful relationship between the contents in the document portioning is improvised
- The efficiency of the scalable database is tested with the time of processing

Load balancing in Mongo DB will also be handled in the evaluation. The migration of data from one sever to another can be showed using Mongo DB when there is more load on the first server. Mongo DB supports an automated sharding/partitioning architecture, enabling horizontal scaling across multiple nodes. For applications that outgrow the resources of a single database server, Mongo DB can be configured to automatically managing failover and balancing of nodes. Private cloud environment can also be set up with the help of Eucalyptus in which the application could be easily deployed.

In order to handle the content from multiple sources the time stamp of the indexed content will be concerned. Based on the time stamp the primary virtual document will become old VD and then it is also referred as one of the sources. The communication between the primary and New primary is initiated to check whether the content matches. Both the documents are ranked accordingly as it enters a cluster based on domain. To realise this concept as the VD grows, an algorithm 5 is proposed based on Low Latency Fault Tolerance Protocol (LLFT, Michael-Melliar Smith, 2012). Figure 7 shows the detailed implication of the LLFT for the proposed system.

Algorithm 5:

Input: The nodes from the content chunks in document
The steps involved in the algorithm are as follows:

- Virtual documents from the source websites are termed as primary virtual documents
- Compare the new document with old document and if the domain matches then latest VD then it becomes new primary
- Checking the timestamp ts_1 and ts_2 of the VD, the new primary document with the latest ts_2 will be the replica of original VD

User interface of the system: The user interface is nothing but the panel which is created for copying the content and the URL's of the content. The panel can also be of dashboard kind which will grab the URL and the content simultaneously, the grabbed content+URL is saved in the database. The users are given separate login to access the documents in the proposed design, so that every time the user login there will be a trigger for the change tracker to analyse the data from the source websites. To handle this scenario the server should be powerful and scalable for faster and accurate tracking of the data.

RESULTS AND DISCUSSION

The results given in this study are only the preliminary results of the proposed framework. The results

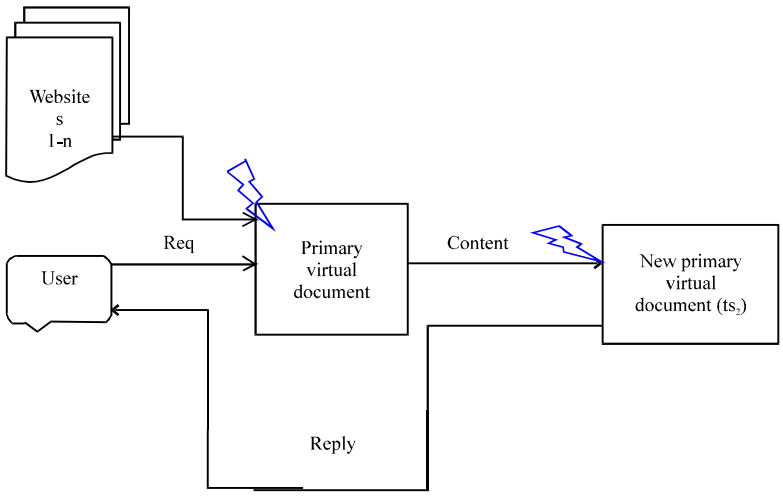
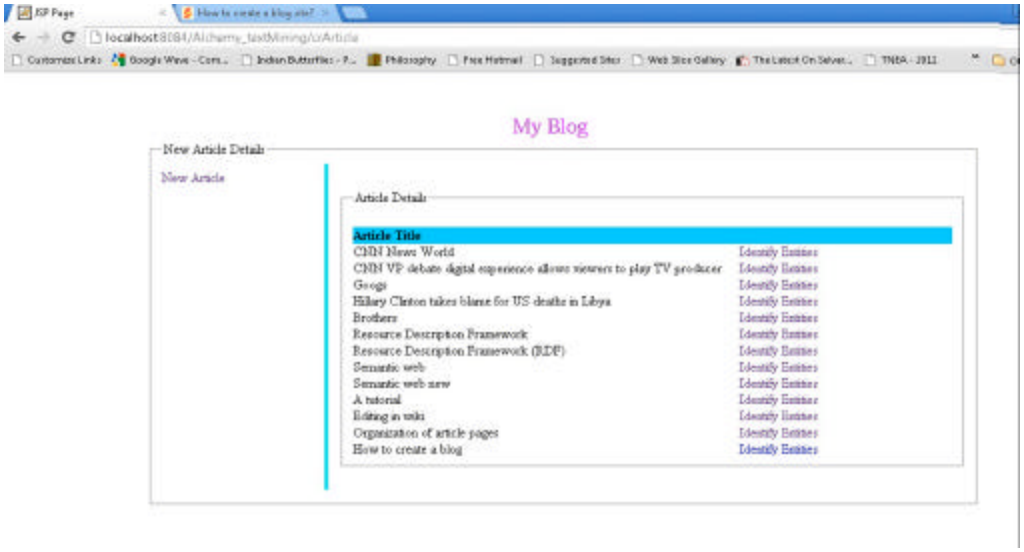


Fig. 7: Creation of virtual document replica

only depict the initial work of the Content Marker module given in the Section IV. To experiment the proposed concept the Blogs are created as new web documents, the articles in the blog are created from the existing websites and an interface is created to copy and paste URL and content from the exiting source websites. Figure 8a shows the screen shot of the panel for creation and implementation of named entity recognition in new articles of the Blog. Initially the content marking and URL copying is done for one user and one article. The details of the created articles are stored in the form of entities, the entity identification and annotation of the content in the blogs is done using the Semantic annotation tool called Alchemy API which is configured and customized accordingly. Figure 8b shows the storage of the blogs in BLOB format. The XML formatted output file from AlchemyAPI tool is stored in the database. By linking an XSL style sheet with the output xml file, the identified entities are finally marked. The entities identified are annotated semantically for the future location and matching of the source content of virtual documents.

(a)



(b)

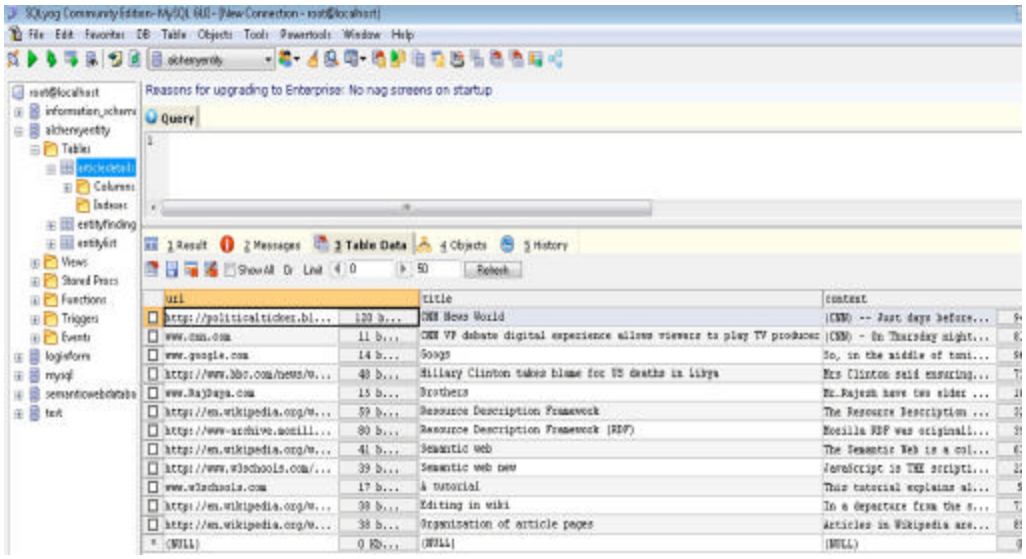
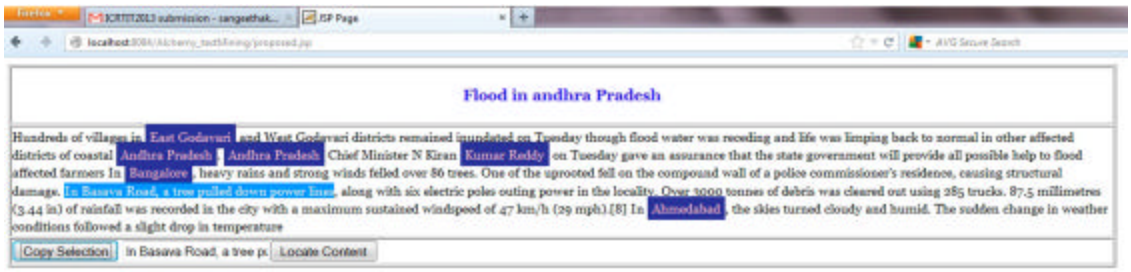


Fig. 8(a-d): Continue

(c)



(d)

Fig. 8(a-d): (a) Implementation of named entity recognition in the newly created blog, (b) Storage of weblogs in database in blob format (c) Identification of marked content in the blog and (d) Location of marked content in the URLs metadata

The content in the document is initially highlighted and then exactly located in the document in the future.

Alchemy API is configured and JAR files are executed for the required code customization. The entities identified are exactly shown in the left side of the panel. Figure 8c shows the screen shot of the marked content in the blog. Figure 8d shows the screenshot of the metadata which is located for the corresponding marked content in the blog. Alchemy API tool is used for named entity recognition as mentioned above and the output files of the identified entities are stored as an XML file in the database. The XML output in database is shown in the Fig. 9. The XML file as shown comprises of the named entity identified using the alchemy API. The highlighted content in the new article will be exactly located in the source website. Pre-processing was done for segregating the meta data of the document and then the matching is carried out as shown in the Fig. 8d. At present the content is marked only for the identified entities but it will be extended for multiple lines of contents which will also be used for matching exact location of content semantically. This is also applicable for single content having more than one source. There are many tools available for Named Entity Recognition (NER). Some of them are Alchemy API, DBpedia Spotlight, Extractiv, Opencalais and Zemanta. Alchemy API was having best performance with maximum precision of 0.7415 and Maximum recall of 0.8916 (Aksac *et al*., 2012).

The time taken for the exact entity retrieval is calculated for the list of test articles to show the efficiency of Alchemy API embedded in the proposed system. A graph is drawn by increasing the number of entities till 100 counts. The time taken for average retrieval varies from 2000 to 5000 m sec in the tested articles. It does not depend upon number of entities which depicts the efficiency of the AlchemyAPI. Figure 10 shows the

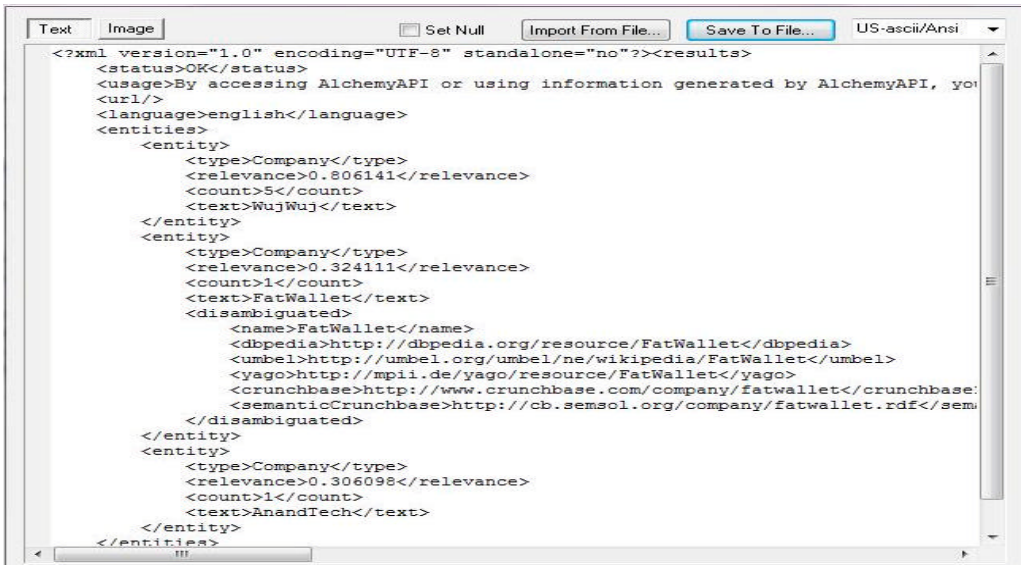


Fig. 9: Stored output format extracted from alchemy API

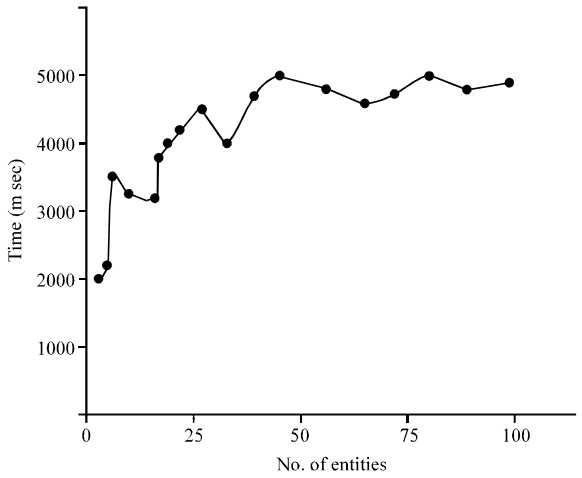


Fig. 10: Retrieval time for stored entities

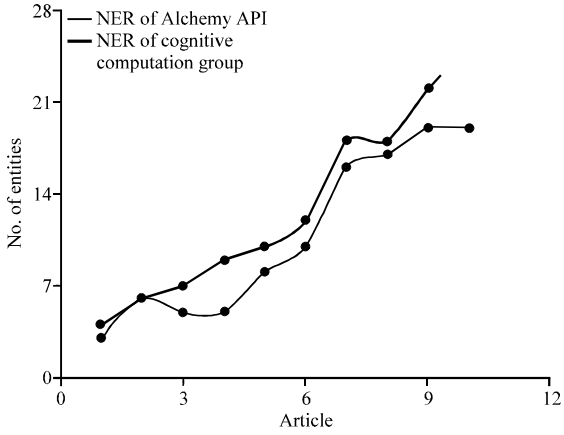


Fig. 11: Named entity recognition-comparison with that of computational group

efficiency in terms of retrieval time for entities in the graph. The Recall and Precision is calculated for different articles by increasing the number of entities in each article.

In the Proposed framework NER results of each article with respect to the Alchemy API tool was compared with the NER results of same artical tested in online tool provided by the cognitive computation group, University of Illinois at Urbana-Champaign. Ten to Fifteen articles with varying entities are tested and results are plotted in the graph as shown in Fig. 11. Based on the comparison, it was decided to calculate the recall and precision for the proposed system using Alchemy API by having the online tool of CCG as existing system, the comparison of NER using Alchemy API tool and by Cognitive Computation Group (CCG) is depicted to be more or less same and moreover Alchemy API proves exact and accurate retrieval. Based on the number of entities identified from CCG' NER, the recall and precision percentage for entity recognition was calculated. Equation 3 and 4 are used for the calculation of Recall and Precision. Recall is the fraction of the entities that are relevant to the query that are successfully retrieved.

$$\text{Recall} = \frac{\left|\{\text{relevant entities}\} \cap \{\text{retrieved entities}\}\right|}{\left|\{\text{relevant entities}\}\right|} \qquad (3)$$

Table 1: Table for indicating accuracy and relevancy of entities identified in the article

Articles	1	2	3	4	5	6	7	8	9	10
Recall	1.00	0.94	0.90	0.83	0.83	0.80	0.76	0.71	0.56	0.50
Precision	0.83	0.88	0.89	0.90	0.92	1.00	0.95	1.00	1.00	0.99

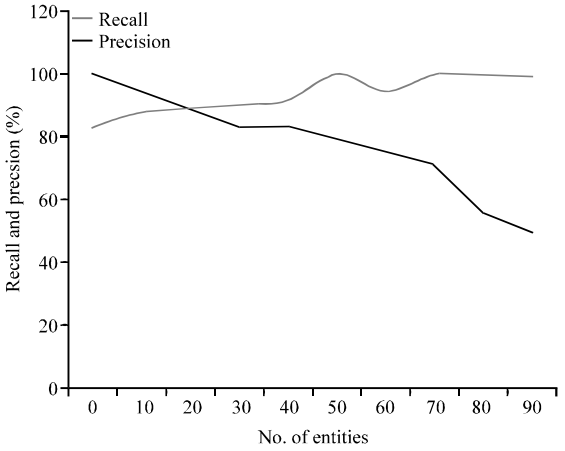


Fig. 12: Comparison chart of recall and precision against number of entities

Precision is the fraction of retrieved entities that are relevant to the search.

$$\text{Precision} = \frac{\left|\{\text{relevant entities}\} \cap \{\text{retrieved entities}\}\right|}{\left|\{\text{retrieved entities}\}\right|} \quad (4)$$

Table 1 clearly shows the accuracy and relevancy of the named entities which can be understood by recall and precision percentage of entity recognition calculated for a set of 10 articles.

A graph which shows the comparison of recall and precision for identifying entities is illustrated in Fig. 12. Recall and precision percentage are inversely proportional to each other. When recall increases, precision will decrease. In the proposed system the embedded Alchemy API shows maximum precision of 1.000 and maximum recall of 0.9431 which is better than the NER of CCG tool.

CONCLUSION

The proposed framework can be used for E-Learning application which manages huge amount of data. It can also be used as an extension for bookmarking in all the browsers. The system manages the content semantically and tracks the source website of annotated content. To handle the growing data storage it will be deployed in cloud environment. The application can also be used as a plugin for the browsers. Once the plugin is installed it will initialize the steps like content marking and change tracking automatically. This can be very well integrated in to the applications based on existing content management systems like Drupal, Joomla and Word Press. In this study the initial static implementation of semantic annotation technique is carried out. The marked content is highlighted and it is located in the source websites. Future work will be enhancing the implementation of the framework using dynamic environment like cloud and hadoop.

REFERENCES

Aksac, A., O. Ozturk and E. Dodgu, 2012. A novel semantic web browser for user centric information retrieval: Person. Expert Syst. Appl., 39: 12001-12013.

Auer, S., C. Bizer, G. Kobilarov, J. Lehmann and Z. Ives, 2007. DBpedia: A nucleus for web of open data. Proceedings of 6th International Conference of the Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference, November 11-15, 2007, Busan, Korea, pp: 722-735.

Bolettieri, P., F. Falchi, C. Gennaro and F. Rabitti, 2007. Automatic metadata extraction and indexing for reusing e-Learning multimedia objects. Proceedings of the International Multimedia Conference, September 28, 2007, Augsburg, Bavaria, German, pp: 21-28.

Diaz-Sanchez, D., F. Almenarez, A. Marin, D. Proserpio and P. Arias Cabarcos, 2011. Media cloud: An open cloud computing middleware for content management. IEEE Trans. Consum. Electron., 57: 970-978.

Dobecki, M. and W. Zabierowski, 2010. Web-based content management system. Computing, 9: 127-130.

Fazzinga, B., G. Gianforme, G. Gottlob and T. Lukasiewicz, 2011. Semantic Web search based on ontological conjunctive queries. Web Semantics Sci. Services Agents World Wide Web, 9: 453-473.

Ismail, A. and M. Joy, 2011. Semantic searches for extracting similarities in a content management system. Proceedings of the IEEE International Conference on Semantic Technology and Information Retrieval, June 28-29, 2011, Putrajaya, pp: 113-118.

Kiryakov, A., B. Popov, I. Terziev, D. Manov and D. Ognyanoff, 2004. Semantic annotation indexing and retrieval. Web Semantics: Sci. Services Agents World Wide Web, 2: 49-79.

Leavitt, N., 2010. Will NoSQL databases live up to their promises. Computer, 43: 12-14.

Mendes, P.N., M. Jakob, A. Garcia-Silva and C. Bizer, 2011. DBpedia spotlight: Shedding light on the web of documents. Proceedings of the 7th International Conference on Semantic Systems, September 7-9, 2011, Graz, Austria, pp: 1-8.

Padhy, R.P., M.R. Patra and S.C. Satapathy, 2011. RDBMS to NoSQL: Reviewing some next-generation non-relational database's. Int. J. Adv. Eng. Sci. Technol., 11: 15-30.

Schaffert, S., J. Baumeister, F. Bry and M. Kiesel, 2008. Semantic wikis. IEEE Software, 25: 8-11.

Tyagi, S., S.D. Sawarkar and P. Lokhande, 2012. Performance and security measure of highly performed enterprise content management system. Int. J. Comput. Appl., 46: 11-17.

Zhang, Y., 2011. WFCMS: An excellent web content management system. Proceedings of the International Conference on Multimedia Technology, July 26-28, 2011, Hangzhou, pp: 3305-3307.

Zhong, L., S. Huang, P. Liu and Y. Yu, 2007. A design for an online RSS reader based on AJAX. Proceedings of the 8th ACIS International Conference on Software Engineering Artificial Intelligence, Networking and Parallel Distributed Computing, July 30-August 1, 2007, Qingdao, China, pp: 794-798.