STUDY OF KNOWLEDGE DISCOVERY ON THE WEB USING FUZZY APPROACH

SWATI KUMAR, ASHOK SHAKY

Abstract— **The World Wide Web contains large number of information. It is very difficult for a user to find the high quality information which he wants to need. When we search any information on the web, the number of URL's has been opened. User wants to show the relevant on the top of the list. So that Page Ranking algorithm is needed which provide the higher ranking to the important pages. In this paper, we discuss the Page Ranking algorithm to provide the higher ranking to important pages.**

***Index Terms*— Web Mining, Web Usage Mining, Web Structure Mining, Web Control Mining, HITS algorithm, Page Rank.**

I. INTRODUCTION

In this paper, we illustrate a method of query processing, which reclaim the documents containing not only the query terms but also documents having their synonyms. The scheme performs the query processing by retrieving and scanning the inverted index document list. We demonstrate that query reaction time for conjunctive Boolean queries can be noticeably condensed, at cost in terms of disk storage, by applying range partition feature of Oracle to reduce the primary memory storage space requirement for looking the inverted list. The proposed scheme is based on fuzzy relations and fuzzy reasoning to retrieve only top ranking documents from the database. To group similar documents Suffix tree clustering is used.

Text databases are widely used as information repositories and can contain vast volume of data. To search a word, word searching technique require that some or all of the words be indexed with a pointer in the word database i.e., lexicon.

Bratley and Choueka have proposed a permuted dictionary mechanism for processing partially specified terms in database queries. This technique uses a permuted lexicon that consists of all possible rotations of each word in the lexicon. Prefix-omission compression mechanism is used to reduce the space requirements of the permuted. Despite the use of compression mechanism, method desires more space, since a word of n characters contributes $n+1$ entries to the permuted lexicon.

Manuscript received December 23, 2014.

SWATI KUMAR, Research Scholar, Saroj Institute Of Technology And Management, Lucknow, India.

ASHOK SHAKY, Assistant Professor, Computer Science Department, Saroj Institute Of Technology And Management, Lucknow, India.

The simplest method to search a word in the lexicon that match a given pattern is to use brute force, that is, search the lexicon from the beginning to end. To use this method, lexicon doesn't have to be sorted, and no supplementary structures are required. But it is a slow process. To influence the pattern matching itself, the UNIX regex package can be utilized. This package provides reasonably rapid pattern matching over a rich pattern language. To match a pattern against a series of words, a function is utilized to compile the pattern into an interior form, and then, for each word, an additional function is utilized to check whether the pattern and word match. This method represents one extreme in the space-time tradeoff, at which the space overheads are small and independent of the size of the lexicon. But searching is slow. For reasonable response, more structured access is needed.

In 1994, Moffat and Zobel introduced the concept of inverted file, a standard mechanism for supporting conjunctive queries. An inverted file holds entry for each term that appears in the database along with the list of number of documents containing that term. This self-indexing inverted file reduces the time required to process ranked queries by a factor of between two and four. Moffat and Zobel consider compressed lexicon model to be placed in main memory, eradicating the disk access required to search for query terms and improving query performance. But it is effective if given the large main memories on computers. It furthermore requires compression and decompression techniques to access the lexicon.

Many sophisticated methods are used to organize the index file, such as B-trees, SB-tree , TRIEs, hashing or variations and combinations of these. STAIRS uses two levels for the index files. Words that begin with the same pair of letters are stored jointly in the second level, while the first level restrains pointers to the second level, one pointer for each letter pair; Lesk uses an over-loaded hash table with separate chaining, to attain fast retrieval in a database of bibliographic entries. Although the above method is simple to implement, quick and supports synonyms, but the main drawback are the storage overhead (which can reach up to 30% of the original file size), the cost of updating and reorganizing the index, if the environment is dynamic, and the cost of merging the lists, if they are too long or too many.

II. BACKGROUND

Text databases are widely used as information repositories and can contain vast volume of data. To search a word,

word searching technique require that some or all of the words be indexed with a pointer in the word database i.e., lexicon.

Bratley and Choueka have proposed a permuted dictionary mechanism for processing partially specified terms in database queries. This technique uses a permuted lexicon that consists of all possible rotations of each word in the lexicon. Prefix-omission compression mechanism is used to reduce the space requirements of the permuted. Despite the use of compression mechanism, method desires more space, since a word of *n* characters contributes *n+1*entries to the permuted lexicon.

The simplest method to search a word in the lexicon that match a given pattern is to use brute force, that is, search the lexicon from the beginning to end. To use this method, lexicon doesn‘t have to be sorted, and no supplementary structures are required. But it is a slow process. To influence the pattern matching itself, the UNIX regex package can be utilized. This package provides reasonably rapid pattern matching over a rich pattern language. To match a pattern against a series of words, a function is utilized to compile the pattern into an interior form, and then, for each word, an additional function is utilized to check whether the pattern and word match. This method represents one extreme in the space-time tradeoff, at which the space overheads are small and independent of the size of the lexicon. But searching is slow. For reasonable response, more structured access is needed.

In 1994, Moffat and Zobel introduced the concept of inverted file, a standard mechanism for supporting conjunctive queries. An inverted file holds entry for each term that appears in the database along with the list of number of documents containing that term. This self-indexing inverted file reduces the time required to process ranked queries by a factor of between two and four. Moffat and Zobel consider compressed lexicon model to be placed in main memory, eradicating the disk access required to search for query terms and improving query performance. But it is effective if given the large main memories on computers. It furthermore requires compression and decompression techniques to access the lexicon.

Many sophisticated methods are used to organize the index file, such as B-trees, SB-tree , TRIEs, hashing or variations and combinations of these. STAIRS uses two levels for the index files. Words that begin with the same pair of letters are stored jointly in the second level, while the first level restrains pointers to the second level, one pointer for each letter pair; Lesk uses an over-loaded hash table with separate chaining, to attain fast retrieval in a database of bibliographic entries. Although the above method is simple to implement, quick and supports synonyms, but the main drawback are the storage overhead (which can reach up to 30% of the original file size), the cost of updating and reorganizing the index, if the environment is dynamic, and the cost of merging the lists, if they are too long or too many.

Hyperlink structures are one of the features of Web pages. Users can navigate the huge Web space easily through this hyperlink structures. In this section, we appraise related work of Information Retrieval systems using the web‘s hyperlink information. Carrier and Kazman proposed a method of web pages, acquired from search results. The rank of a page in their model is computed as equal to the sum of its in-degree and its out-degree, thus, based on the concept of “directionless” version of WWW link structure.

In 1999, Chakrabarti et al. studied the hyperlink structure of the WWW for information discovery and categorization. Links within a ordinary domain are treated as navigational links and are assigned low weight. Also large numbers of pages from a single domain are assigned low weights to prevent a single site from becoming dominant. Yang proposed a Fusion method of combining various content-based (VSM) and link–based (HITS) systems to remove the inadequacies of singular web IR approaches.

The analysis of the investigational results obtained suggested that index source, query length, and host definition were the most influential system parameters for retrieval performance and the optimum performance level of one method can be raised by combining it with a reasonably effective method of a different kind.

Westerveld et al. suggested an empty page finding task to retrieve the web pages based on information about the document‘s content along with its in-links, URLs and anchors. The page URL is classified in four categories-Root, SubRoot, Path and file. It was found that the proper combination of URL and in-link information (i.e., without the independence assumption) performs better than the two separate priors. Also it is observed that URL information gives the best prior information. Adding in-links yields marginal improvement.

Wang and Kitsuregawa in 2002 proposed an improved clustering algorithm to cluster the web search based on the similarity between the documents computed using the keywords appearing in the anchor text of a hyperlinked web page. The link information of the hyperlinked web page i.e., the quantity of out-links and the in-links is also taken into consideration for analyzing the similarity between the two documents taken under consideration.

Glover et al. proposed two methods: a combination method and uncertainty sampling for improving the classification accuracy of the web pages. They utilized document‘s full-text, inbound anchor-texts and extended anchor-texts (words and phrases occurring near a link to a target page) to classify web pages and to name clusters of web pages.

In 2003, Eiron and McCurley studied the relation of an anchor text to a document and showed that the documents retrieved by anchor text techniques improve the quality of web text search than documents retrieved by content indexing. Their study revealed that anchor text is less ambiguous than other types of texts like title of document which are typically longer than individual anchor text and

thus they resembles real-world queries in terms of its term distribution and length. Also anchor text offers improved indication of the summarization of the page in dissimilar contexts, by different people, than that afforded by a single title which is authored by one author.

Kurland and Lee proposed a structural re-ranking approach using standard language- model based ranking principle to reorder the documents in an initially retrieved set by exploiting asymmetric relationships between them based on generation links, which indicate that the language model induced from one document assigned high probability to the text of another.

From the above study, it is observed that ranking of a web page is highly influenced by the following factors

(i) Text content of a web page

(ii) Anchor text of the hyperlinks in the page

(iii) In-links to a page and out-links from a page

(iv) Web page URL

But Content-based IR approaches have difficulty dealing with the diversity in vocabulary and quality of web documents, while link-based approaches undergo from incomplete or noisy link topology. This insufficiency of singular web IR approaches focuses the researchers to modify the existing conventional IR methods and to propose new approaches which use both page content and link information of web pages to solve IR problems . Pirolli et al. exploited new sources of information including web page meta-information and usage statistics to develop new techniques to arrange web pages based upon functional categories, node types as well as relevancy. Szymanski and Chung analyzed the similarity of the word usage at the different link distance from the page of interest and demonstrated that a structure of words used by the linked pages enables more efficient indexing and search.

III. FUZZY INFORMATION RECUPERATION

The term fuzzy Information Recuperation refers to methods of Information Recuperation that are based upon the theory of fuzzy sets. These techniques are increasingly recognized as more realistic than the various classical methods of Information Recuperation.

The problem of Information Recuperation involves two finite crisp sets, a set of recognized index terms,

$X = \{x_1, x_2, \ldots\ldots, x_n\}$, and a set of relevant documents , $Y = \{y_1, y_2, \ldots\ldots, y_n\}$. (1)

Although these sets vary whenever novel documents are added to systems or new index terms are recognized (or, possibly, when some documents or index terms are discarded), they are fixed for each particular inquiry.

In fuzzy Information Recuperation, a fuzzy relation expresses the relevance of index terms to individual documents, $R: X \times Y \rightarrow [0,1]$, such that the membership value $R(x_i, y_i)$ specifies for each $x_i \in X$ and each $y_j \in Y$ the grade of relevance of index term x_i to document y_j. One way of determining the grades objectively is to define them in an appropriate way in terms of the frequency of occurrences of individual index term in the document involved relative to the size of the document.

Another important relation in fuzzy Information Recuperation is called a *fuzzy thesaurus T*. It is composed of a number of semantic relations (equivalence *E*, inclusion *I*, association *A*) that cover every possible semantic entity. *T* is principally a reflexive fuzzy relation, defined on X^2. For each pair of index terms $(x_i, x_k) \in X^2$, $T(x_i, x_k)$ expresses the degree of association of x_i with x_k; that is, the degree to which the sense of index term x_k is compatible with the meaning of the given index term x_i. By using the thesaurus, the user query can be expanded to contain all the associated semantic entities. The expanded query is expected to retrieve more pertinent documents, because of the superior probability that the annotator has included one of the associated entities in the description. Use of the equivalence *E* relation, which is analogous to the MPEG-7 *equivalentTo* and *identifiedWith* relation, expand the query to contain terms that are, in one sense or another, synonyms.

Now, Let *A* denote the fuzzy set representing a particular inquiry. Then, by composing *A* with the fuzzy thesaurus *T*, a new fuzzy set on *X* (say, set *B*) is obtained, which represents an augmented inquiry. That is,

$A \circ T = B$, (2)

where $\circ$ is typically considered to be the max-min composition, so

$B(x_j) = \max \min[A(x_i), t(x_i, x_j)]$, $x_i \in X$ for all $x_j \in X$.

The retrieved list of documents, articulated by a fuzzy set *D* defined on *Y*, are then obtained by composing the augmented inquiry, articulated by fuzzy set *B*, with the significance relation *R*. That is,

$B \circ R = D$ (3)

where $\circ$ is usually understood to be the max-min composition, so

$D(x_j) = \max \min[B(x_i), r(x_i, y_j)]$, $x_i \in X$, $y_j \in Y$.

The above document recuperation systems return long lists of ranked documents that users are forced to surf through to find relevant documents. The majority of today's Web search engines follow this paradigm. Web search engines are also characterized by extremely low precision. The short precision of the Web search engines coupled with the ranked list presentation make it hard for users to find the information they are looking for. Consequently clustering has to be applied to group the set of ranked documents

returned in response to a query.

IV. ALGORITHM OF SUFFIX TREE CLUSTERING (STC)

A suffix tree is a data structure that admits efficient string matching and querying. Suffix trees have been premeditated and used broadly, and have been applied to elementary string problems such as finding the longest repeated substring , approximate string matches, string comparisons, and text compression.

Suffix Tree Clustering is a fresh, incremental, *O(n)* time algorithm designed to meet the requirements of post-recuperation clustering of web search results. STC does not treat a document as a set of words but rather as a string, making use of proximity information among words. STC relies on a suffix tree to efficiently identify sets of documents that share common phrases and uses this information to create clusters and to succinctly summarize their contents for users.

The STC algorithm has three logical steps, which will be detailed in the following sections:

1.**Document Parsing**, in which each document is transformed into a sequence of words and phrase boundaries are identified.

2.**Phrase Cluster Identification**, in which a suffix tree is used to find all maximal phrase clusters. These phrase clusters are scored and the highest scoring ones are selected for further consideration.

3. **Phrase Cluster Merging**, in which the selected phrase clusters are merged into clusters, based on the overlap of their document sets.

A suffix tree of a string S is a compact tree containing all the suffixes of S. Documents are treated as strings of words, not characters, thus suffixes hold one or more complete words. In more precise terms:

i. A suffix tree is a rooted, directed tree.

ii. Each internal node has at least 2 children.

iii. Each edge is labeled with a non-empty sub-string of *S* (hence it is a tree). The label of a node in delineated to be the concatenation of the edge-labels on the path from the root to that node.

iv. No two edges out of the same node can have edge-labels that begin with the same word.

v. For each suffix *s* of *S*, there exists a suffix-node whose label equals *s*.

Each suffix-node is marked to designate from which string (or strings) it originated from (*i.e.*, the label of that suffix node is a suffix of that string). In the application, suffix tree is constructed including all the sentences of all the documents from the collection.

Each node represents a group of documents and their common phrase. Each base cluster is assigned a score, that is a function of the number of documents it holds, and the words that make up its phrase.

The score *s(B)* of base cluster *B* with phrase *P* is given by: $s(B) = |B|.f(|p|)$,

Node	Phrase	Documents
a	cat ate	1, 3
b	Ate	1, 2, 3
c	Cheese	1, 2
d	Mouse	2, 3
e	To	2, 3
f	ate cheese	1, 2

Table 1: Six nodes and their corresponding base clusters

Where, $|B|$ is the number of documents in base cluster *B*, and $|P|$ is the number of words in *P* that have a non-zero score (i.e., the effective length of the phrase). Words appearing in the stoplist, or that appear in too few (3 or less) or too many (more than 40% of the collection) documents receive a score of zero. The function *f* penalizes single word phrases, is linear for phrases that are two to six words extended, and becomes invariable for longer phrases.

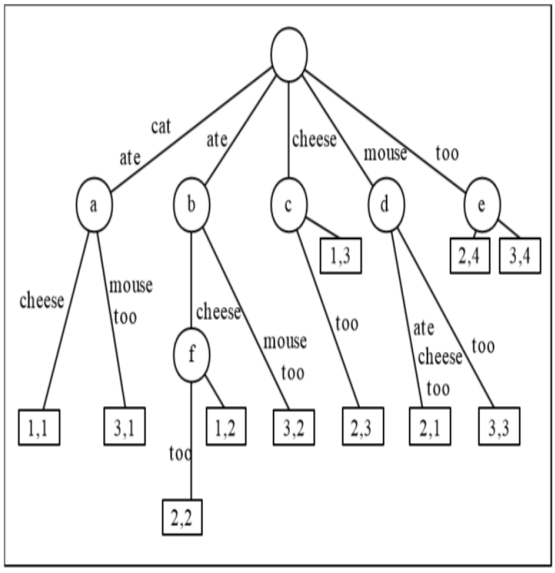

Figure 1: The suffix tree of the strings "cat ate cheese", "mouse ate cheese too" and "cat ate mouse too"

c) Combining Base Clusters Step

Documents may share more than one phrase. As effect, the document sets of dissimilar base clusters may overlie and

may even be alike. To evade the proliferation of nearly identical clusters, the third footstep of the algorithm combines base clusters with a high overlap in their document sets (phrases are not considered in this step). A binary similarity measure is defined between base clusters based on the overlap of their document sets. For known two base clusters B_m and B_n, with sizes $|B_m|$ and $|B_n|$ correspondingly, and $|B_m \cap B_n|$ representing the number of documents common to both base clusters, we delineate the resemblance of B_m and B_n to be 1 iff:

$$|B_m \cap B_n|/|B_m| > \alpha \text{ and}$$

$$|B_m \cap B_n|/|B_n| > \alpha$$

Otherwise, their resemblance is defined to be 0.

Next, consider the base cluster graph, where nodes are base clusters, and two nodes are linked if the two base clusters have a resemblance of 1. A cluster is delineated as being a linked component in the base cluster graph. Each cluster holds the union of the documents of all its base clusters.

e.g.	Node	Phrase	Documents
B_m:	a	cat ate	1,3
B_n:	b	ate	1,2,3

Table 2: Node a,b from the Table 1 with their corresponding base clusters to find similarity between them

The Suffix Tree Clustering algorithm is incremental. As each document arrives from the Web, it is "clean" and is added to the suffix tree. Each node that is updated (or created) as a result of this is tagged. Then update the pertinent base clusters and recalculates the similarity of these base clusters to the rest of the base clusters. Finally, check if the changes in the base cluster graph result in any changes to the final clusters.

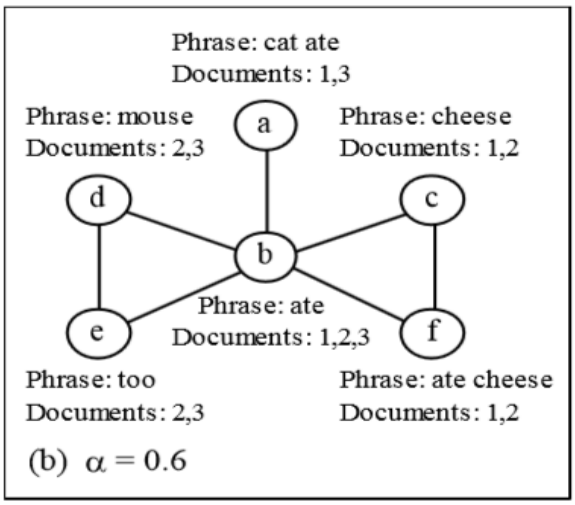

Figure 2 The base cluster graph

The goal of a clustering algorithm in domain is to group each document with others sharing a common topic, but not essentially to partition the assortment. The STC algorithm does not require the user to specify the required number of clusters. It does, on the other hand, require the specification of the threshold used to determine the similarity between base clusters. However, it has been found that the performance of STC is not very sensitive to this threshold, unlike Agglomerative Hierarchical Clustering (AHC) algorithms that showed extreme sensitivity to the number of clusters required.

V. PROPOSED METHOD

Based on the approach of Moffat and Zobel, we proposed a solution to recuperation system that is both quick in operation and economical in disk storage space. The proposed method stored the lexicon using Range partition feature of Oracle that can access the lexicon in blocks from disk, by delineating a range partition on first letter of each lexicon word. To regain words of the lexicon that match the query term, only the partition corresponding to first letter of each query term is fetched into main memory from disk and the words (along with their synonyms) in the partition are identified via the index delineated on the lexicon database. Using this method, the entire lexicon database is not required to be in memory during the query search. Only very few, small size, disk accesses are required (utmost equal to number of query terms, if all query terms have dissimilar first letter) to obtain the relevant information from the stored lexicon database. The proposed method associates the semantic entities (concepts, objects, events) with the aid of a fuzzy thesaurus. It is based on fuzzy relations and fuzzy reasoning.

VI. QUICK TEXT RECUPERATION ALGORITHM

In this section, we will describe the algorithm for grouping the retrieved ranked documents that is based on the structure outlined .

The steps of the algorithm are explained below:

1. Consider a query *Q*. Ignore all the stop words from the search query. Final query is represented by set *A*.

2. For each query term tr,

   a. If target partition of term_table is not in main memory then fetch it to memory from disk

   b. Search the partition term_table to locate tr

   c. Record the posting_list (synonyms with their fuzzy thesaurus association) for term tr, in the final list *X*.

3. Identify the fuzzy thesaurus *T* for each pair of index terms $(x_i,x_k) \varepsilon X$, where $t(x_i,x_k)$ expresses the degree of association of x_i with x_k.

4. Compute

$$B \leftarrow A \; o \; T$$

5. For each term tr ε *X*

   a. if target partition of word_document table is not in main memory then fetch it to memory from disk

   b. Search the partition word_document table to get word_id for term tr

   c. Record the posting_list (for each term tr, get Elias γ codes representing document_id along with their term_frequency value)

   d. Decompress to get the actual document-IDs

   e. Record the document_ids to the final relation *R* along with their term_frequency

6. Compute

$$D \leftarrow B \; o \; R$$

7. Inspect only those document_ids captured by some α-cut of D

8. Bring the document_table in memory

9. For each *d* ε *D*

   a. look up the address of document d in document_table b. retrieve document *d*

The reclaimed documents are ranked but not grouped**.** Therefore clustering has to be applied to the set of documents returned in response to a query. Here, we discuss Suffix tree clustering method to group the retrieved documents .

The steps to be performed to group the retrieved documents are as follows:

1. Perform document "Cleaning" to delete word prefixes and suffixes and to reduce plural to singular

2. Identify Base Clusters. Each base cluster is assigned a score that is a function of the number of documents it holds, and the words that make up its phrase. The score s(B) of foundation cluster B with phrase P is given by:

   $s(B) = |B|.f(|p|)$

   where $|B|$ is the number of documents in base cluster *B*, and $|P|$ is the number of words in *P* that have a non-zero score

3. Combine Base Clusters with common documents. For known two base clusters B_m and B_n, with sizes $|B_m|$ and $|B_n|$ respectively, and $|B_m \cap B_n|$ representing the number of documents common to both base clusters, we define the resemblance of B_m and B_n to be 1 iff:

   $|B_m \cap B_n|/|B_m| > \alpha$ and $|B_m \cap B_n|/|B_n| > \alpha$

   Otherwise, their resemblance is defined to be 0.

CONCLUSION

This is the survey paper of PageRank algorithm. PageRank is a better approach for calculating the page value which is a numeric value that represents the importance of a page on the web. There is given a formula for calculating the total number of pages which are linked together and counting these links as a vote. Those page will go on the top of the list which has higher number of numeric value or votes.

REFERENCES

[1] Akrivas G., and Stamou G., "Fuzzy Semantic Association of Multimedia Document Descriptions", 2007.

[2] Aliguliyev R.M., "Clustering of Document Collection – A Weighing Approach", Expert Systems with Applications, vol. 36(4), pp. 7904-7916, 2009.

[3] Aljaber B., Stokes N., Bailey J., Pei J., "Document Clustering of Scientific Texts Using Citation Contexts", Information Retrieval, vol. 13(2), pp. 101-131, 2010.

[4] Arai K., and Barakbah A.R., "Hierarchical K-means: an algorithm for centroids initialization for K-means", Journal of the Faculty of Science and Engineering, Saga University, Japan, vol. 36, no. 1, pp. 25-31, 2007.

[5] Arora R., and Ravindran B., "Latent Dirichlet Allocation and Singular Value Decomposition based Multi-Document Summarization", Proc. Eighth IEEE International Conference on Data Mining (ICDM 2008), IEEE Press, pp. 713-718, 2008.

[6] Basagic R., Krupic D., Suzic B., "Automatic Text Summarization", Information Search and Retrieval, WS 2009, Institute for Information Systems and Computer Media, Graz University of Technology, Graz, 2009.

[7] Bernotas M., Karklius K., Laurutis R., Slotkienė A., "The Peculiarities of the Text Document Representation, Using Ontology and Tagging-Based Clustering Technique", Information Technology And Control, Kaunas, Technologija, vol. 36, no. 2, pp. 217 – 220, 2007.

[8] Chang Y., and Chien J., "Latent Dirichlet Learning for Document Summarization", Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 1689-1692, 2009.

[9] Chatterjee N., and Mohan S., "Extraction-Based Single-Document Summarization Using Random Indexing", Proc. 19th IEEE International Conference on Tools with Artificial Intelligence, vol. 2 (ICTAI 2007), pp. 448-455, 2007.

[10] Dalal M.K., and Zaveri M.A., "Heuristics Based Automatic Text Summarization of Unstructured Text", Proc. International Conference and Workshop on Emerging Trends in Technology (ICWET 2011), pp. 690-693, 2011.

[11] "Data Mining", http://en.wikipedia.org/wiki/Data_mining, 28 May 2011.

[12] Feinerer I., Hornik K., Meyer D., "Text Mining Infrastructure in R", Journal of Statistical Software, vol. 25, no. 5, pp. 1-54, 2008.

[13] Huang A., "Similarity Measures for Text Document Clustering", Proc. Sixth New Zealand Computer Science Research Student Conference, NZCSRSC2008, Christchurch, New Zealand , pp. 49-56, 2008.

[14] Information Extraction, http://en.wikipedia.org/wiki/Information_extraction, 21 May 2011.

[15] Information Retrieval, http://en.wikipedia.org/wiki/Information_retrieval, 20 May 2011.

[16] Lee S., Song J., Kim Y., "An Empirical Comparison of Four Text Mining Methods", Journal of Computer Information Systems, vol. 51, no. 1, pp. 1-10, 2010.

[17] Li Y., Chung S.M., Holt J.D., "Text document clustering based on frequent word meaning sequences", Data & Knowledge Engineering archive , vol. 64, no. 1, pp. 381-404, 2008.