Abstract—From the times when humans know how to store the information on stones, metal sheets, papers etc. there is requirement to access information from all these information storages. In today’s era information retrieval becomes more complex as the mediums are available to store huge amounts of information as on computer’s hard disk, secondary storages etc. Thus as information stored is in large amounts some system is also required to access the relevant information from these mediums. Such systems are known as information retrieval systems. Many such systems are available which provides relevant information to the users, dependent on their needs. As some systems provides the quality information on the basis of the number of times the words of query are appeared in the documents, those documents are given greater priority which having higher frequency of words appeared in the query. Some systems provide relevant information with the help of index. Index is generated by taking keywords (words) from the stored information. By using index searching the relevant documents becomes most accurate, fast and relevant. There are many types of indexes are created.

Index Terms—Index, Information Retrieval.

I. INTRODUCTION

When there is a large collection of documents exists then also some method is required to get the best result from the collection for the fired query, such a method is known as information retrieval and software system which supports this process is known as Information Retrieval System. In IR system data is stored in the form of documents known as corpus, these documents are interrelated to each other or may be having different information in them. All these documents are the collection of keywords. Query fired by the user is the composition of the keywords. When a query is accepted by the IR system then the search for a query is based on the keywords provided by the user, those documents are made available to the user having these keywords. If higher frequency words match with topic keyword, then the document is considered to be relevant. This is method which is used by many of the IR systems.

II. RELATED WORK

ONTOLEGY BASED DOCUMENT INDEXING

Working of knowledge intensive organizations like consultancy services, consultancy and supply services, data processing services etc. are dependent on the information which has been gathered from various resources of information which may be inside the organization and/or external resources like the internet. This is very important to the organization to make the best use of gathered information from these sources. Knowledge intense organizations can do the knowledge management, which deals in the field of various phases of knowledge life cycle: identification, acquisition, development, sharing, use and preservation of organization knowledge. Thus a system is designed for gathering the information for such tasks and the system developed is web based Webocrat system. Webocrat can interact at the knowledge level with the help of language. This language has been provided to the system by ontology with syntax and semantic rules. Use of ontology enables to define concepts and relations representing knowledge about a particular document in domain specific terms.

Scheme of Document Retrieval

Here developed package with three different approaches to document retrieval: vector representation, latent semantic indexing method (LSI), and ontology-based method used in the Webocrat system.
Vector Representation Approach

In this approach all documents are represented as vectors. In first step all the documents are preprocessed using preprocessing tools. In this step stop words are removed and stemming is done. Then as the internal representation of document vector of index term weights are calculated. Weights are calculated using tf-idf scheme.

\[ w = \text{tf}_{ij} \cdot \text{idf}_{ij} \]

where \( \text{tf}_{ij} = \frac{\text{freq}_{ij}}{\max_i \text{freq}_{ij}} \) and \( \log \left( \frac{N}{n_i} \right) \)

\( n_i \) is the number of documents are given by \( N \), and is the frequency document for term \( t_i \) in the whole document collection. Such a vector is then normalized to unit length and stored into the term-document matrix \( A \), which is the internal representation of the whole document collection. In term-document matrix \( A \) those vectors are stored which are normalized to unit length, and this represents the collection of whole documents internally. For query \( Q \) if it is required to search some documents which are relevant to the query \( Q \) it is necessary to represent the query \( Q \) in the same way as a vector \( D_i \) (i.e. a vector of index term weights). Similarity between a query \( Q \) and a documents \( D_i \) is computed as cosine of those two normalized vectors (document and query vectors).

\[ \text{Sim}_{\text{TF-IDF}}(Q, D_i) = \frac{D_i \cdot Q}{||D_i|| \cdot ||Q||} \]

Latent Semantic Indexing Approach

LSI approach is based on singular value decomposition of tf-idf matrix \( A \). By this decomposition three matrixes are computed:

\[ A = USV^T \]

where \( S \) is the diagonal matrix of singular values and \( U, V \) are matrices of left and right singular vectors. If the singular values in \( S \) are ordered by size, the first \( k \) largest values may be kept and the remaining smaller ones are set to zero. The product of the resulting matrices is a matrix approximately equal to \( A \), and is closest to \( A \) in the least squares sense.

\[ A \approx A_{\text{SVD}} \] where \( A_{\text{SVD}} = U_k S_k V_k^T \]

Ontology Based Approach

Here for document retrieval ontology is used by Webocrat system. In Figure 2 the whole process of query processing by this approach is shown. When a user throw a query then first appropriate concepts are retrieved (here this work can be done manually) Then the set of concepts associated with each document is retrieved from database. Then using simple metric both the sets are compared which expresses the similarity between a document \( D_i \) and given query \( Q \).

\[ \text{Sim}_{\text{onto}}(Q, D_i) = \frac{Q_{\text{con}} \cup D_{i,\text{con}} \neq \emptyset}{k} \]

Where \( Q_{\text{con}} \) is a set of concepts assigned to query \( Q \) and \( D_{i,\text{con}} \) is a set of concepts assigned to document \( D_i \). and \( k \) is small constant, e.g. 0.1. Resulted number represents ontology-based similarity measure.

Figure 2 An ontology based document retrieval

KEYWORD BASED RETREIVAL SYSTEM

Here introduced a distributed information retrieval framework which is based on the Okapi probabilistic model, this is the framework which is able to achieve the same level of effectiveness as those achieved by a single centralized index system. Okapi model is proposed by Robertson and Spark Jones. This is the enhanced probabilistic retrieval model based on binary independence model. Here used a simplified Okapi weighting function, in which \( w \) was assigned to a given term \( t_i \) in a document \( d \) and was computed according to the following formula:

\[ w_i = (k_1 + 1) \cdot \frac{\text{tf}_{ij}}{K + \text{tf}_{ij}} \]

where

\[ K = k \cdot (1 - b) + b \cdot \frac{1}{\text{avdl}} \]

\( l \) is the document length,
\( \text{avdl} \) is the average of document length.
\( b, k, k_1 \) are constants.
\( \text{tf}_{ij} \) is the occurrence frequency of term \( t \) in document \( d \).

The following formula shows the weight given to the term \( t_i \) in document \( d \).

\[ w_{qi} = \frac{q_{tf_{i}}}{k_3 + q_{tf_{i}} \cdot d_{i}} \cdot \log \left( n - d_{i} \right) \]

www.alliedjournals.com
INDEX BASED INFORMATION RETRIEVAL SYSTEM

This is a three step process:

**Indexing:** In this step preprocessing of documents has been done. Here first a file is created for each document containing the words other than stop words (at, the, is, an etc.) also the stemming of the words is done, so that to get the words in their root forms (like: use is the root form of using, used, usable). In the next step frequency of each word has been counted and the words having frequency more than threshold value (based on a formula) is placed as a index term. In the collected form all such terms create index table for that document.

**Formulation:** In this step first of all query has to be expanded based on the domain knowledge which stored as a tree in the form of ontological structure.

**Comparison:** In comparison step, when user give a query to the system than a comparison between the query and stored document representatives is started and then it is decided which document has to be displayed to the user and in what order. According to the relevancy whole or part of the document is provided. This comparison can be done in the form of matrix multiplication. For this purpose all the documents are given a id and similarly ids are given to the queries and matrix of both are generated separately. After this matrix multiplication can be done and the necessary result has been generated. Then with the help of this result it can be found out which document is more relevant to the query. Mathematically it can be shown as:

Consider there are 2 documents \( i \) and \( j \) represented as:

\[
\text{Doc } (i) = (\text{Term}(g), \text{Term}(j)) \ldots \text{Term} (k)
\]

\[
\text{Doc } (j) = (\text{Term}(g), \text{Term}(j)) \ldots \text{Term} (j)
\]

Where \( k \) and \( i \) are no. of terms in respective documents.

So, all terms for all documents together can be represented as:

\[
[\text{Term}(g), \text{Term}(j)] \ldots \text{Term} (k) \cup \text{Term} (g), \text{Term}(j) \ldots \text{Term} (j)
\]

\[
= [\text{Term}(g), \text{Term}(j)] \ldots \text{Term} (k) \cap \text{Term}(g), \text{Term}(j) \ldots \text{Term} (j)
\]

\[
\text{i.e. Term}(1) \ldots \text{Term}(n) = \text{all distinct terms of both documents } i \text{ and } j
\]

Here comparison is based on weighted values and implication of inverted document frequency (IDF):

Weight defines how many times a term of the query is appeared in the document searched. Thus weight implies for the particular document the relevancy of the term.

IDF is inverse of document frequency calculated for incorporating measure that favors terms which occur in fewer documents. In lesser number of documents query terms occur, value of weight is increases along with.

If terms are relevant to the document result than the greater value has been shown by the weight * IDF factor. By using this searching time is also reduces effectively.

ATTENTION BASED INFORMATION RETRIEVAL

As we know IR systems provide the user relevant information which generates on some criteria. In some systems the relevancy of the information is collected from implicit feedback sources. This feedback is based on clicking of the mouse, input from keyboard, scrolling etc. In Attention based IR, here an eye tracker is used as the feedback observer. It observes about the user that which part of the document is read out or skimmed by the user. Information generated by eye tracker is stored in an attention annotated documents (ex. Passages which are commented, or highlighted) with the help of these annotated documents, estimations on user’s current thematic context can be implicitly or explicitly generated. Challenges with this system are to know the reaction of the user about the information provided. This can be found out by taking explicit feedback from the users. By taking into account such a feedback relevancy of the document increases. This requires high efforts from users side thus can be avoided. Thus there is a focus on implicit feedback. Here eye tracker can observe the movement of eyes very precisely and thus provides a feedback through user’s eye movement as it can directly reflects the user’s attention. Thus eye tracker becomes a very useful tool for IR systems in providing contextual information. But due to their expensiveness they are not widely used. In near future if it becomes less expensive than can be used by workers in middle or large – sized enterprises.

![Figure 3 Shows generation of attention based index.](image)

**Attention-Based Index**

Documents reflects the mentality of the authors who write it, thus its relevancy to the users is may varied, as users regards different parts of the document with different intensities. Therefore, a document index for local desktop search, which supports retrieval of already used documents, should consider the user’s degree of attention on the different parts of used documents. Secondly when search for the query has been made than not only single document is made available, there are many documents which are related to the query. Thus some connection is maintained between these documents in the index so as to find them for later use.

**[III] PROPOSED WORK**

The architecture of the proposed work based on context based indexing in IR system is shown in the following figure 4.
Design And Implementation Of Context Based Information Retrieval System

Figure 4 Architecture of context based indexing
Here in the above architecture web crawler is used as it can copy the web pages accessed and later provided for fast search. These web pages are stored in the page repository in documents form. Than these documents are passed through context manager and indexer where the pre-processing of the documents are done i.e. documents are broken into tokens (words) form than the documents are sent to the next stage where using thesaurus, words which are having authenticated meanings are taken and then context of each word using dictionary is generated. Now the main part presented work is started, i.e. creating index using the tokens (words) and contexts. The index is generated using the binary search tree and a pointer is also theirs which points to the documents which having the searched keywords. Now when user enter the query using search interface, than query has been sent to the context manager and indexer which first provides the user contexts of the word searched by the user, when particular context has been selected than system generates the related documents containing words which are presented by the query. At the next level these documents are ranked i.e. given priority by ranking module dependent on the frequency of the words contained by the document. Higher the frequency of words in the document higher will be the rank.

Indexed structure for binary search tree
Structure used to store the documents is shown in the figure 5.

Figure 5 Data Structure used to store document keywords. Structure used for the data stored is given in figure 5. Here all the alphabets (A-Z) are stored in the index for each alphabet there is a BST on whose node those words are stored which are started from the corresponding alphabet. Again for each word their store corresponding contextual meaning words and a pointer which indicates to the documents having related words in them. Here the focus is providing fast response to the user. For searching the particular documents, user fire a query and then search begins as starting with searching with the first letter of the keyword given and then starts matching the words over the BST when the match is found, list of contextual meaning words are provided to the user, when particular choice has been made by the user than only those documents are made available to the user which are having the searched keywords. Thus the search becomes more relevant.

Detailed structure of a node.
This section provides details of the context based indexing structure to serve as a reference for implementation purpose. The indexed information is stored in the tree nodes themselves in the form of meaning/context tables. Figure 6 shown the information stored in one of the nodes given in example.

Figure 6 Structure of a node.
Here,
Token- This is the keyword entered in the query and is appeared in the documents stored in the corpus.
Context List- This is the list of the words which having similar meaning to the searched keywords.
Con1-This stands for contextual sense 1. In context list with each word there exists a pointer which points to those documents having these words.
PD1-This stands for the pointer to document number 1. Here PD1 helps to point out the particular documents having those words which are searched by the user, so as to make search less time consuming.

Algorithm for indexing-
The algorithm shows the various steps to construct the context based searching of keywords.

Step 1 The preprocessing of the documents which includes stemming and removing stop words is performed by indexer.
Step 2 Parse the document using parser and thus generates the list of tokens (words).
Step 3 Create a BST.
Step 4 Store Tokens in the BST.
Step 5 Convert BST into inorder BST.
Step 6 Query entered by user, from these query keywords are extracted.

Step 7 Context are made available to the user.

Step 8 Search begins for the query as according to the context has been chosen.

Step 9 Result has been generated in the form of document names.

Step 10 Frequency of the keyword, found in the documents is also given along with document names.

[IV] RESULT

Using the above algorithm IR system is implemented as follows

The interface includes.

1. One button to display all files names
2. Second button to parse all files and add all words generated in a BST.
3. Third button to display BST in inorder.
4. There is one text box to enter the query.
5. Fourth button to show the available contexts.
6. There is another text box to enter the value of the context, so that only relevant documents are made available to the user.
7. Fifth button is used to show the query entered and context selected.
8. Finally there is a button to show the result for the query.

Figure 7 Final result snapshot.

[V] CONCLUSION

In this paper keywords are indexed along with their context which is extracted from the documents. Here Binary Search Tree is used to store the keywords for improving the performance of the retrieval system in terms of accuracy and efficiency for retrieving more, relevant documents as per the requirements of the user as context of the keywords are also provided. Thus using index for searching purposes provides fast access to document context structure along with an optional searching.

REFERENCES


[2]. “Term Proximity Scoring for Keyword-Based Retrieval Systems” Yves Rasolofo and Jacques SavoyUniversity de Neuchâtel, Neuchâtel, Switzerland Published in Lecture Notes in Computer Science 2633, 1611-3349, 2003

[3]. “Ontology-based Information Retrieval” Jan ParalicDepartment of Cybernetics and AI, Technical University of Kosice, Letna 9, 04011 Kosice, Slovakia Ivan KostialDepartment of Cybernetics and AI, Technical University of Kosice, Letna 9, 04011 Kosice, Slovakia

[4]. “Attention-Based Information Retrieval” Georg BuscherGerman Research Center for Artificial Intelligence (DFKI)

[5]. “Retrieval Effectiveness Of An Ontology Based Model For Conceptual Indexing” Sofia Stamou Computer Engineering and Informatics Department, Patras University, 26500 GREECE