Relevant Page Retrieval and Query Recommendation using Semantic Analysis of Queries

Usha Yadav, Neelam Duhan, Bhawna Kaushik

Abstract—Due to the massive size of the Web and low precision of user queries, search results returned from present web search engines can reach to even hundreds of thousands of documents. Therefore, finding the right information can be a tedious task if not impossible. This paper represents an approach that tries to solve this problem by finding the semantic similarity and similarity metrics to group similar terms using clustering techniques. These similar terms are bind to each other using links in proposed index repository in order to facilitate presentation of results in more compact form and enable the semantic browsing of the results set.

Index Terms: Index Repository, Search Result Clustering, Fuzzy similarity, Semantic similarity

1 INTRODUCTION

In traditional Search engine, when a user submits a query on its interface, it gets processed and a large list of documents is shown to the user. The users have to go through all these documents to find out the required ones. This is very tedious and time consuming process. One approach to manage large results set is by clustering. The concept arises from document clustering in Information Retrieval (IR) domain: find groupings for a set of documents so that documents belonging to the same cluster are similar and documents belonging to different cluster are dissimilar. Search results clustering thus can be defined as a process of automated grouping search results. However, in contrast to traditional document clustering, clustering of search results is done as per user query request. Clustering of search results can help user navigate through large set of documents more efficiently. By providing concise and accurate description of clusters, it lets user localizes interesting document faster. The growth of WWW is likely to explode and thus for efficient access and exploration of useful information, appropriate interfaces to search and navigation through this enormous collection are of critical need.

Web search engines allow user to formulate a query, to which it responds using its index to return set of references to relevant web documents. User can search for information by navigating through categories to identify the needed reference. Although the performance of search engines is improving every day, searching on the Web can be a tedious and time-consuming task due to the following facts:

- Search engines can only index part of the "indexable" Web, due to the huge size and highly dynamic nature of the Web.
- The user's "intention behind the search" is not clearly expressed which resulted in too general, short queries.
- As the effects of the above two, results returned by search engine can count from hundreds to hundred thousands of documents
- Over half of users did not access results beyond the first page and more than three in four users did not go beyond viewing two pages

The most important challenge with information retrieval revolves around two basic problems:

1 Getting a good query from search experiences with the aim of helping them craft better questions.
2 Presenting "easy-to-judge" results to minimize what the user has to read through.

For these reasons in this paper, improved data structure for indexing is proposed so as to optimize the Search Engine's result, keeping in mind its storage and access efficiency. Also a new search result interface is proposed in which the internet users are presented with the few coherent groups of queries similar to user's own query. Unlike scanning a long list of documents satisfying a user's query, it is often easier to scan a few coherent groups than many Individual documents. The paper has been organized as follows: Section 2 describes the basis terminologies forming the basis of the proposed work and literature work done under them. Section 3 explains the proposed optimization system. Section 4 explains the Term similarity Analysis module. Section 5 explains the Terms Clustering module. Section 6 explains the Proposed Index Repository along with example illustration. Section 7 describes Query

- *Usha Yadav is currently pursuing Ph.D in computer engineering in YMCA University of Science & Technology, India,. E-mail: usha.yadav.912@gmail.com*
- *Neelam Duhan is currently working as an Associate Professor in YMCA University of Science & Technology. E-mail: neelam_duhan@rediffmail.com*

Recommendation module and section 8 concludes the paper with some discussion on future research.

2 PRILIMINARIES AND RELATED WORK

2.1 Web Crawler

Search engine relies on a crawler module to provide the grist for its operation. [1] describes that Crawlers are small programs that `browse' the Web on the search engine's behalf, similarly to how a human user would follow links to reach different pages. The programs are given a starting set of URLs, whose pages they retrieve from the Web. The crawlers extract URLs appearing in the retrieved pages, and give this information to the crawler control module. This module determines what links to visit next, and feeds the links to visit back to the crawlers. (Some of the functionality of the crawler control module may be implemented by the crawlers themselves.) The crawlers also pass the retrieved pages into a page repository. Crawlers continue visiting the Web, until local resources, such as storage, are exhausted. In most cases, the crawler cannot download all pages on the Web. With so much research in this area, comprehensive search engine currently indexes a small fraction of the entire Web [2],[3]. Considering this fact, it is important for the crawler to carefully crawl the pages and to visit important pages by prioritizing the URLs in the queue properly, so that the fraction of the Web that is visited (and kept up-to-date) is more meaningful. In this paper we have utilize the terms present in Page Repository to find the similarity between these terms in order to further recommend user with more useful query.

2.2. Indexing

In [4],[5] described that the existing search engine rely on the concept of Inverted Index for purposes of information storage. Basically an inverted index contains two parts dictionary and postings. Dictionary contains terms (vocabulary or lexicon) and for each term, there is a list that records the documents in which term occurs. Each item in the list which records that a term appeared in a document is conventionally called a posting. The list is then called a postings list (or inverted list), and all the postings lists taken together are referred to as the postings. In another paper [6], the author presented an improved schema for Index Repository for the purpose of better interpretability and efficiency of crawling and indexing process in web search engine and also to provide User Feedback Mechanism. In modified Index Repository, the schema was described as: Dictionary, Document Posting, Position Postings
where
Dictionary: <term, doc_freq>
Document Postings: <Doc_ID, depth, inlinks, outlinks, rank, click_count, term_freq>
Position Postings: <pos1, pos2, … posn>

Dictionary consists of the terms and the document frequency i.e. the number of documents containing the term, while Postings provide the related document information. The last field in the document postings is the term frequency which gives the number of occurrences of the term in the document. The position postings give the positions in a document at which term appears. We had further customized the schema of Index Repository so as to map the similar terms in a cluster together by using new filed "pre-link" and "next-link".

2.3 Similarity Metrics for Terms Clustering

Perfect clustering requires a precise definition of the proximity between a pair of objects, in terms of either the pair wised similarity or distance. A variety of similarity or distance measures have been proposed and widely applied, such as edit distance, cosine similarity and the Jaccard correlation coefficient. Meanwhile, similarity is often conceived in terms of dissimilarity or distance as well [7]. Given the diversity of similarity and distance measures available, their effectiveness in text document clustering is still not clear. Although Strehl et al. compared the effectiveness of a number of measures [8].

Semantic similarity measures play important roles in information retrieval and Natural Language Processing. Lexical relations are very difficult concepts to define formally; a detailed account is given by [9]. Rather than struggle with a operational definition of synonymy and similarity, It will be good to rely on lexicographers for 'correct' similarity judgments by accepting words that co-occur in thesaurus entries (synsets) as synonymous.. The English thesaurus has been a popular arbiter of similarity for 150 years [10], and is strongly associated with the work of Peter Mark Roget [11]. Researches done on lexical resources are, the most influential computational lexical resource is WORDNET [12]. WORDNET, developed by Miller, Fellbaum and others at Princeton University, is an electronic resource, combining features of dictionaries and thesauri, inspired by current psycholinguistic theories of human lexical memory

In this paper, we focused on two measures of similarity with some example illustration: Fuzzy Similarity metrics with edit distance and semantic similarity using WORDNET. Edit distance measures the minimum number of edit operations (insertion, deletion, and substitution) to transform one string to another. And the lexical resource WORDNET used to compute the semantic similarity between words like synonym, hypernym, meronymy.

2.4 Search Results Clustering

One approach to manage large results set is by clustering. The concept arises from document clustering in Information Retrieval domain: find groupings for a set of documents so that documents belonging to the same cluster

are similar and documents belonging to different cluster are dissimilar. Search results clustering thus can be defined as a process of automatically grouping search results into to thematic groups. However, in contrast to traditional document clustering, clustering of search results are done as per user query request and locally on a limited set of results return from the search engine. Clustering of search results can help user navigate through large set of documents more efficiently. By providing concise, accurate description of clusters, it lets user localizes interesting document faster.

The earliest work on search results clustering were done by [13], the author proposed a new approach to this problem by constructing a graph of concepts (which represents text) rather than collection of words. [14] worked on Scather/Gather system, followed with application to web documents and search results by [15],[16] to create Grouper based on novel algorithm Suffix Tree Clustering. Inspired by their work, a Carrot framework was created by [17] to facilitate research on clustering search results. This has encouraged others to contribute new clustering algorithms under the Carrot framework like LINGO [18], AHC [19]. Other clustering algorithms were proposed by Zhang [20], Semantic Hierarchical Online Clustering using Latent Semantic Indexing to cluster Chinese search results or Class Hierarchy Construction Algorithm by [21]. An in-depth survey of Search Result Clustering algorithms is available in [22].

2.5 Limitation of Existing Work

A critical look at the available literature indicates the following issues, which need to be addressed while building efficient recommendation system for search engines:

1. Lack of user's knowledge in formatting queries on Search engine.
2. Improve precision of user query by suggesting or recommending queries.
3. Providing the user with the search result interface which is compact and concise.
4. Instead of clustering the documents based on the common phrases only, is the main concern. Semantic relationships among the documents should also be considered.
5. Data Structure used for indexing need to be examined so as to optimize the storage and access efficiency.
6. Processing time and memory used in Clustering and indexing need to be examined properly.

In the next chapter, proposed work is discussed. Various components and the data structures used in the work of the proposed system are described in detail. Example illustration and a sample interface for representation of results are also presented.

3 PROPOSED OPTIMIZATION SYSTEM

The proposed system tries to improve the efficiency of the existing search engines. In search result clustering, search results mean the documents that were returned in response to a query. The default presentation of search results in information retrieval is a simple list. Users scan the list from top to bottom until they have found the information they are looking for. In the proposed system, the approach of compacting the search results is developed, so that similar documents appear together. By this way, it is often easier to scan a few coherent groups than many individual documents.

In the proposed system, a new data structure of index repository is proposed to optimize and enhance the efficiency of the Search results. The large collection of terms present in the Page Repository of the Crawler is generally collected after parsing the downloaded web pages. This collection of terms can prove very useful to optimize the User Search Result. So, Similarity between all these terms is calculated so as to form number of clusters, containing terms similar to each other based upon their semantic. This cluster information is used by the indexer to enhance its Data Structure so as to provide user with more recommended queries based upon the query submitted by the user on Search engine Interface.

The proposed system works in four steps which briefly are given below to have an overview of how the system works:

Step 1: The motive of the system is to recommend the user with more semantically related query. For this, the similarity between the huge collections of terms in the page repository is found. This similarity is calculated based upon the fuzzy based similarity function which uses Levenshtein edit distance and thesaurus based similarity function which finds the relationship between terms using the lexical database.

Step 2: Term cluster database containing the clusters of similar terms is formed using the Term Clustering Module which depends upon the two similarity functions discussed in Step 1.

Step 3: These cluster information are embedded in proposed index repository so as to link all the similar terms together in the dictionary.

Step 4: When the user puts its query, then it receives the documents fulfilling its query along with the recommended set of query (coherent groups) which are semantically related to the user query.

3.1 Proposed System Architecture

The architecture for the proposed system is shown in Fig 3.1, which consists of the following functional components:

1. Term Similarity Analyzer
2. Term Clustering Module

3. Query Recommender

Along with the functional components, two databases are also been used. These are given below:

1. Term Cluster Database
2. Proposed Index Repository

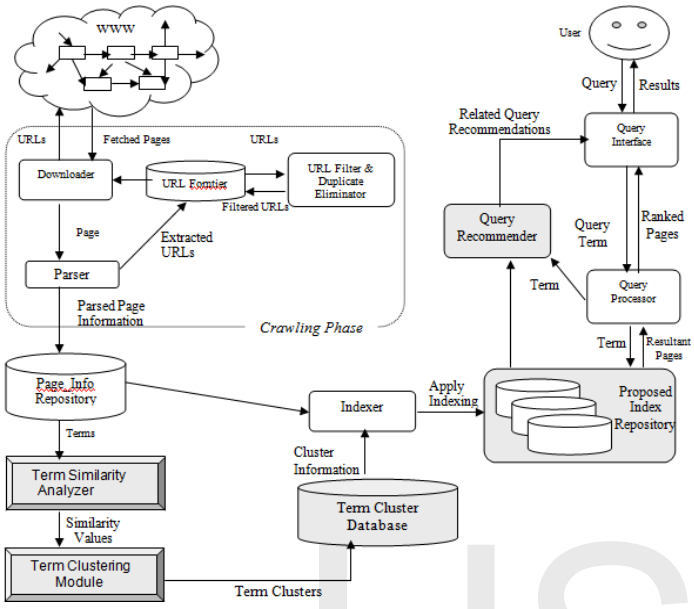

Fig. 3.1 High Level Architecture of the Proposed Optimization System

4 TERM SIMILARITY ANALYZER

The component Term Similarity Analyzer computes the Semantic similarity of the terms, which are collected in the database known as Page Repository of the Crawler. After the crawler downloads the web pages, parsing is done and all the terms in the web pages is collected in the Page Repository. This repository contains the complete knowledge about all the fetched pages In the proposed architecture, Page repository is mined by applying various similarity functions on terms(token) to measure the semantic similarity between the them. Similarity of each term is calculated with all the other terms in the Page Repository. This component uses two similarity functions, one is Fuzzy based Matching which uses Leveinshtein Edit Distance and other is Thesaurus Based Similarity which uses any lexical Database or resources such as WordNet.

With each iteration, two terms from the repository are taken by the Fuzzy based similarity function and Thesaurus based similarity function to calculate the similarity value till the similarity of each terms with all the other terms are not calculated. The similarity value is calculated in the range [0,1]. The average value of both the similarity functions has been calculated. It means equal weightage is given to both the similarity functions used. But as per the requirement, it can be changed. Any of the similarity functions can be given more weightage than the other. Instead of calculating average value, α and β values can be assigned to both the functions. The combined similarity function is given in (4.1).

$$Simcombined(T1,T2)=\alpha Simfuzzy(T1,T2)+\beta Simthesaurus(T1,T2) \quad (4.1)$$

where α, β can be any value between 0 and 1.

Now, α and β can be given by the experts depending upon the importance given to any of the Similarity function i.e Fuzzy Similarity function or Thesaurus Based Similarity function.

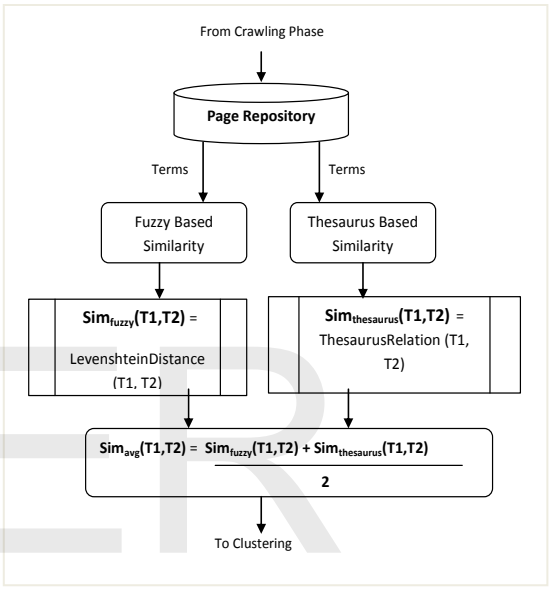

Fig. 4.2 Graphical Representation of Term Similarity module

4.1 Fuzzy Based Similarity

Fuzzy Based Similarity function calculates the similarity values in the range between 0 and 1. It uses Levenshtein Edit Distance, which is widely applied in tasks including spell checking and plagiarism detection. The distance between two text strings is defined to be the sum of the costs of the number of edit operations required for transforming one term into the other. Having insertion, deletion, and substitution as operations, each at the cost of 1, yields the Levenshtein distance.

There is an assumption in Levenshtein distance is that, more is the distance between the two terms, less similar is the terms.

So, to calculate the similarity value, the Levenshtein has been divided by the maximum number of characters contained in any of the two terms and then subtracted from 1 to get fuzzy similarity value between them. More is the fuzzy similarity value; more similar are the two terms. The algorithm to calculate the Fuzzy similarity value is given below in Fig. 4.3. The two terms from the page repository has been given as an input and the fuzzy similarity value is

calculated and returned as output. The output value lies between only 0 and 1 i.e. [0,1].

4.1.1 Example Illustration

Similarity Values between some sample terms are calculated and shown in Table 4.1.

Algorithm: Fuzzy Similarity Calculator ()
I/P : Two terms T1 and T2 from page repository
O/P : Similarity value of two given terms in the range [0, 1]

```
int fuzzy_similarity (char T1[1..m], char T2[1..n])
 {
  // for all i and j, d[i,j] will hold the Levenshtein distance
between
  // the first i characters of T1 and the first j characters ofT2t;
  // note that d has (m+1)*(n+1) values
declare int d[0..m, 0..n]
for i from 0 to m
   d[i, 0] := i // the distance of any first term to an empty second
term
for j from 0 to n
   d[0, j] := j // the distance of any second term to an empty first
term
 for j from 1 to n
  {
   for i from 1 to m
   {
    if T1[i] = T2[j] then
      d[i, j] := d[i-1, j-1]       // no operation required
    else
      d[i, j] := minimum
                (
                  d[i-1, j] + 1,  // a deletion
                  d[i, j-1] + 1,  // an insertion
                  d[i-1, j-1] + 1 // a substitution
                )
    } //end inner for
  } // end outer for

  Leh.distance = d[m,n]

  Return (1- Leh.distance / MAX (m.n) )   // MAX (m,n) calculates which terms
                has maximum  number of characters in it
}
```

Fig. 4.3 Algorithm for Fuzzy Based Similarity Function

Table 4.1 Calculation of Fuzzy Similarity Values

TERM (T1)	TERM (T2)	Fuzzy Similarity Simfuzzy(T1,T2)
human	humanbeing	0.5
people	person	0.33
animal	mammal	0.5
plant	flower	0.16
name	authorname	0.4
elbow	eye	0.2
bark	tree	0
elpoep	nosrep	0.33

The values in the given above table are calculated by taking two terms from the Page Repository and then applying the algorithm as shown in Fig. 4.3. This has been done by applying two most important steps:

1. By making a matrix, whose dimensions depends upon the number of characters in both the terms and then the algorithm is followed to find out the Levenshtein Distance between the two terms.
2. After having Levenshtein Distance, Fuzzy Based Similarity between the two terms has been calculated by applying the formula shown in the algorithm.

After applying all these steps, Fuzzy based similarity of some sample terms are presented in the Table 4.1.

4.2 Thesaurus Based Similarity

In Text Similarity Analyzer, Thesaurus Based Similarity module finds out the semantic relationship between terms such as synonymy, hypernymy and Meronymy relations. All the terms in the page repository of the crawler are analyzed to find out the thesaurus relation between them. Semantic similarity or semantic relatedness is a concept whereby a set of documents or terms within term lists are assigned a metric based on the likeness of their meaning / semantic content. For a machine to be able to decide the semantic similarity, intelligence is needed. It should be able to understand the semantics or meaning of the words. But a computer being a syntactic machine, semantics associated with the words or terms is to be represented as syntax. There are various methods proposed to find the semantic similarity between words. Some of these methods have used the precompiled databases like WordNet, and Brown Corpus. In essence, semantic similarity, semantic distance, and semantic relatedness all mean, "How much does term A have to do with term B?" The answer to this question is usually a number between -1 and 1, or between 0 and 1, where 1 signifies extremely high similarity/relatedness, and 0 signifies little-to-none. For example, “car” and “driver” are related, but not much similar, but “car” and “vehicle” are similar in some degree. Relatedness is thus more general than similarity. Furthermore, semantic distance is the inverse of semantic similarity that is the less distance of the two concepts, the more they are similar.

In this thesis, Semantic Similarity between the terms like synonym, hypernym, meronymy has been calculated by using any of the lexical databases such as Word Net.

4.2.1 Example Illustration

The Thesaurus based Similarity between the terms in the Page Repository is shown in Table 4.2 with the help of WordNet and the similarity value returned is 0 or 1.

Table 4.2 Examples of Thesaurus Based Similarity

Terms (T1)	Terms (T2)	Relationship	Simthesaurus
sick	Ill	Synonymy(T1,T2)	1
name	Authorname	Synonymy(T1,T2)	1
animal	Mammal	Hypernymy(T1,T2)	1
computer	Glacier	No relationship	0
plant	Flower	Hypernymy(T1,T2)	1
elbow	Arm	Meronymy(T1,T2)	1
bark	Tree	Meronymy(T1,T2)	1

While using, Thesaurus Based Similarity, the matcher returns either 0 or 1 as the similarity value. The matcher returns 0 if there exists no relationship between terms T1 and T2 . If any relationship (synonymy, hypernymy or meronymy) exists between two terms, the matcher returns 1. For example, for attributes "author name" and "name" two different terms (see row 2 of Table 4.2) respectively, the matcher returns 1 as the similarity value.

4.3 Average Similarity Measure

The Average Similarity Value has been computed by taking the average of the similarity values obtained from the Term Similarity Analyzer which uses two strategies: Fuzzy Based Similarity which compute the Levenshtein edit distance between the two terms and gives the results in the index range of [0,1] and other is Thesaurus Based Similarity finds out the semantic relationship between terms. Both the measures are equally important, so the average of both the two Similarity measures is taken which is shown in (4.1):

$$\mathrm{Sim}_{avg}(T1,T2) = \frac{\mathrm{Sim}_{fuzzy}(T1,T2) + \mathrm{Sim}_{thesaurus}(T1,T2)}{2} \qquad (4.1)$$

Where T1,T2 are any two terms from the Page Repository, T1 not equal to T2.
As an example illustration, Fuzzy similarity between two terms i.e. name and authorname is 0.4 and Thesaurus Based Similarity between these two terms are 1. So, the average similarity between name and authorname is

Simavg (name, authorname) = (0.4 + 1)/2 = 0.70
Now using the Average Similarity value, the clusters of similar terms can be formed. In this paper both the strategies is given equal importance, but it can be altered by the expert analysts depending on the importance being given to two similarity measures.

The next component Term Clustering Module is explained in the next section.

5 TERM CLUSTERING MODULE

The Term Clustering Module forms a number of different clusters of terms by using the Similarity functions as explained in the previous section. The similar terms are placed together in the same clusters. When user submits some query on the search interface, then all the similar terms along with their documents, which are related to the user query are recommended on the interface in the form of coherent groups to increase the efficiency of User Search results. For obtaining these clusters, various methods and techniques can be used. The standard web search engines have low precision, since typically a large number of irrelevant web pages is returned together with a small number of relevant pages. This phenomenon is mainly due to the fact that keywords specified by the user may occur in different contexts.
Consider for example the term "cluster". Consequently, a web search engine typically returns long lists of results, but the user, in his limited amount of time, processes only the first few results. Thus, a lot of truly relevant information hidden in the long result lists will never be discovered. Text clustering methods can be applied to structure the large result set such that they can be interactively browsed by the user.

5.1 Algorithm of Term clustering module

The Term Clustering Module uses the algorithm shown in Fig. 5.1, where each run of the algorithm computes k clusters. The algorithm is based on the simple perspective: initially, all the terms from the page repository are considered to be unassigned to any cluster. Each term is examined against all other terms in the page repository, collected after parsing the document downloaded by the crawler. If the similarity value turns out to be above the pre-specified threshold value (τ), then the terms are grouped into the same cluster. The same process is repeated until all terms get classified to any one of the clusters. The algorithm returns overlapped clusters i.e. a single term may span multiple clusters. Each returned cluster containing terms is stored in the Term Cluster Database. The clustering algorithm takes O(n2) worst case time to find all the term clusters, where n is the total number of terms.

5.2 Term Cluster Database

In the Proposed Architecture, a term cluster Database is used which store all the clusters containing semantically related terms. The terms from the Page Repository is given to the TSA (Term Similarity Analyzer) to calculate the similarity between the terms using various functions. Then the TCM (Term Similarity Module) is used to cluster the similar terms based on the threshold given by the experts. And then finally all these clusters are stored in the TCD (Term Cluster Database.), where a cluster contains all the terms similar to each other and the dissimilar terms are contained in other clusters.

Algorithm: Term_Clustering(P, τ)

Given: A set of *n* terms in Page Repository (P) and similarity threshold τ
Output: A set C= {C1, C2,…Ck} of *k* term clusters

// Start of Algorithm
k = 1; *//k is the number of clusters.*
For (each term T1 in *P*)
Set Cluster_Id(T1)= Null; *// initially no term is clustered*
For (each *T1* ε *P*)
{
Cluster_Id(T1)= Ck;
Ck={T1};
For (each *T2 in* P such that *T1=! T2*)
{
//Calculate fuzzy based similarity
Sim_{fuzzy} = fuzzy_similarity(T1,T2);
// Calculate the Thesaurus based Similarity
$Sim_{Thesaurus}$ = ThesaurusRelation(T1,T2)
// Calculate the Average Similarity

$$Sim_{avg}(T1,T2) = \frac{Sim_{fuzzy}(T1,T2) + Sim_{thesaurus}(T1,T2)}{2}$$

If ($Sim_{avg}(T1,T2)$>τ) then
set ClusterId(T2)= Ck;
Ck= Ck ∪ {T2};
else
continue;
} // end for
k= k+1;
} //end outer for
Return Term cluster set C.

Fig. 5.1 Algorithm for Term Clustering Module

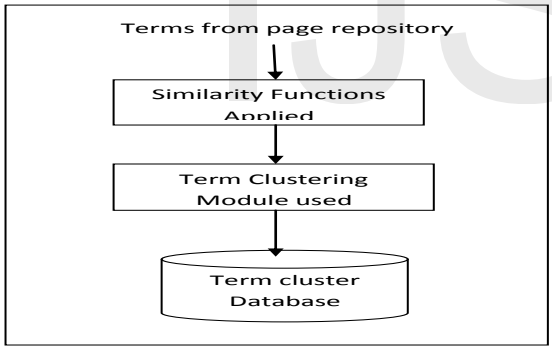

Fig. 5.2 Construction of Term Cluster Database

6 PROPOSED INDEX REPOSITORY

The index is build up by a component called indexer and contains information about all the keywords/terms present in all the downloaded pages by the web crawler. An inverted index constitutes a dictionary of terms (sometimes also referred to as a vocabulary or lexicon). Then, for each term, there is a list that records the documents in which term occurs.

This Index Repository is changed by proposing a new index fields i.e. Pre_link and nxt_link. Pre_link of a term provides link to the previously related term and the nxt_link provides pointer to the next term in the document similar to that term. When a user submit a query term, then all the pre_link and the nxt_link pointed by that term is traversed and list of all the semantically similar terms along with the top ranked documents satisfying that term are presented on to the search interface This will optimize both the user search and the indexing.

6.1 Schema of Proposed Index Repository

In tedious task of information gathering, key role is played by the Data Structure. The schema, used for indexing process has been changed so as to improve its efficiency and ease in construction. Following is the data structures of the Proposed Index Repository as shown in Fig.6.1

Term	Doc _Id	Pre_ link	Nxt_ link	Depth	In- links	Out_ links	Rank	Freq	Position _Info	Click_ count

Fig 6.1 Schema of Proposed Index Repository

.

The detail description of all the fields in the proposed index repository is shown in Table 6.1.Each row contains the different terms in the schema and the column contains the detailed description of the term.

In the Proposed index repository, Dictionary consists of the pre_link which provide pointer to previous related term in the dictionary and nxt_link which provide pointer next related term in the dictionary, terms and the document frequency i.e. the number of documents containing the term, while Postings provide the related document information. The last field in the document postings is the term frequency which gives the number of occurrences of the term in the document. The position postings give the positions in a document at which term appears. So, the schema can also be described as:

Dictionary_ Document Postings_ Position Postings

where

Dictionary: <pre_link ,term, doc_freq, nxt_link>

Document Postings: <Doc_ID, depth, inlinks, outlinks, rank, click_count, term_freq>

Position Postings: <pos1, pos2, … posn>

6.2 Construction of Proposed Index Repository

In this work, few changes have been done to the data structure of the traditional Index Repository. So, how these changes have been made is explained in the following few steps as given below:

In the next section, an external sorting algorithm is suggested as a solution, for the problem of insufficient memory size and the slow speed of indexing process.

1. Traditionally, indexer built index repository by using the information in the Page Repository and using various different approaches. But in this proposed index repository, information from the Term Cluster Database has also been used by the indexer.
2. Indexer uses the cluster information and knows about the terms which are in same cluster and are semantically related to each other.

3. So, while creating the index repository, indexer links all the similar terms to each other. For this indexer uses the concept of doubly link-list.

Table 6.1 : Description of Proposed Index Repository

Field	Description
Term	A normalized token in the page.
Doc_Id	The Document Id in which specified terms is present.
Pre_link	It points to the previous semantic similar term in the dictionary.
Nxt_link	It points to the next semantic similar term in the dictionary.
Depth	The depth of the document with Doc_Id in the web.
In-links	The number of back links of the document derived from the Link Store repository.
Out_links	The number of forward links of the document again derived from the Link Store repository
Rank	It is a score provided to a document generally based upon its link information e.g. Google's PageRank. The rank may also be provided on other parameters such as the popularity, content, depth, click count of a document etc.
Freq	It is the number of occurrences of the specified term in the document
Position_Info	At what positions, the term appears in the document.
Click_count	This field stores an integer number indicating the number of times, users clicked on the document. This information is fed to the indexer from the search engine logs.
Related Link	It provides link to the related term in the cluster depending upon the single term query.

4. With each term, two links are attached called pre_link and nxt_link. Pre_link will point to the term which is related to that term but is stored previously in the dictionary and nxt_link will point to the term which is related to that term but is stored ahead in the dictionary.
5. For each cluster, assign the pre_link of first term in cluster as NULL and assign the nxt_link of last term in cluster as NULL.
6. So that now dictionary contains 4 fields, pre_link, term, document frequency, nxt_link. And document listing contain Doc_ID, depth, inlinks, outlinks, rank, click_count, term_freq and position posting contains pos1, pos2, … posN.

6.3 Example Illustration

The organization of information in an index repository can be understood by the means of some sample terms and related page information as shown in Table 6.2 , which after actually being placed in the index get converted into a dictionary and a list of postings and this can be seen from Fig. 6.2.

In this example, few terms are indexed along with their other details. Values in all the fields are taken to show the construction of the sample Proposed Index Repository. Suppose the terms are Append, Born, Human, People, Person. Now suppose the similarity between all these terms are calculated with each other through two similarity function which are FBS and TBS as discussed in Term Similarity Analyzer.

Table 6.2 Sample Index Information

Location	Terms	Doc ID	Pre_ link	nxt_ link	Depth	In-Link s	Out-Link s	Ran k	Freq	Position Information	Click Count
101	Append	D5	-----	-----	3	5	4	9	2	7,35	5
102	Append	D7	------	-----	5	4	25	6	4	4,6,26,99	21
211	Born	D2	----	----	2	3	8	7	3	8,13,25	9
-------	------	----	----	-----	-----	-----	-----	-----	-----	----------	---------
223	Human	D3	-----	-----	2	10	21	8	2	5,19	12
-------	------	----	-----	-----	-----	-----	-----	-----	-----	----------	---------
485	People	D9	223	512	3	12	22	15	4	4,16,24,43	17
-------	------	----	-----	----	-----	-----	-----	-----	-----	----------	---------
512	Person	D6	----	----	4	8	32	2	5	9,19,32,47,51	13

After calculating the similarity, suppose Human, People and person all terms belong to same cluster and are semantically related to each other. Then while creating the proposed index repository, every term in the same clusters are linked to each other using pre_link and nxt_link as shown for the term "people" in the left part of Fig. 5.9, called dictionary. So, now it can be seen that how the term "people" is stored in the Proposed Index Repository. Therefore, the schema can also be described as:

Where

Dictionary: < 223, 512, People, 3 >

Document Postings: < 9, 3, 12, 22, 15, 17, 4 >

Position Postings: < 4, 16, 24, 43 >

Now suppose, user submit a query "people", then all the pre_link and nxt_link of term person are traversed, to accumulate the terms semantically similar to "People". These terms are recommended to user as explained in section 3.7.In the similar way, the whole sample index information is organized which is called as Repository and shown in the Fig. 5.9.

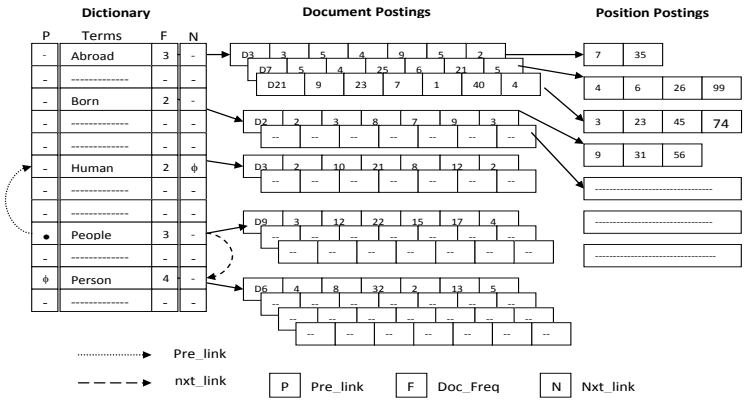

Fig. 6.2 Organization of Information in a Proposed Index Repository

7. QUERY RECOMMENDER

In the Proposed Optimization system, Query recommendation is the most important work with the help of which Search results are optimized. Query

recommendation module deals with recommending semantically similar queries to the user depending upon the query given by him. Traditional web search engines represent the most popular way to access information in the Web. The procedure of search engines is based on a simple pattern: for a query asked by a user, search engine responds with a list of search results. It is very difficult task for a user to scan the long list of retrieved search results and to find out the document actually needed by him. The user wants to find some concepts and then these concepts are written in words, many internet users are not well accustom to the way, query put on the search engine interface. So sometimes it happens that, even after going through a list of documents he is not being able to get the desired result.

So, Clustering of search results can help user navigate through large set of documents more efficiently. It lets user localizes interesting document faster. In this work, the component query recommender provide the user with the semantically related query along with their documents as a search result but in a coherent group, so that the user can pick any of the query first and then zoom it out to check for web documents satisfying that query.

7.1 Example Illustration

For query recommendation, an interface is designed for the user as shown in Fig. 5.11, named as Pencarian Search Result Interface. Continuing the example as taken in section5.6.4, when the user enters its query, for example "people" on this interface as shown on top in Fig. 7.1, and then he is given the list of documents satisfying his query. Also, he is recommended with more numbers of queries like Person, Human etc. which are semantically related to the query submitted by the user as shown in extreme left side of the Fig. 7.1. The organization of information for the query "people" in Proposed Index Repository is shown in Fig 6.2.

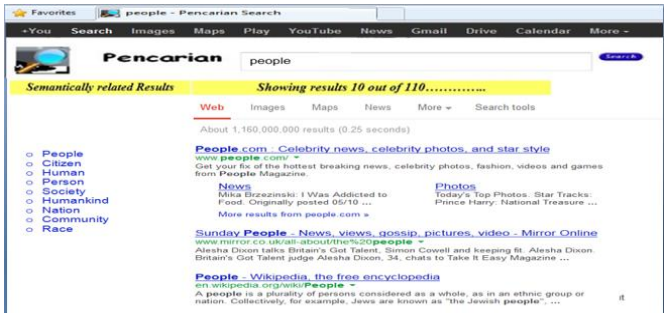

Fig.7.1 Pencarian Search Result Interface for query "people"

Suppose, the user chooses recommended query, "Society". Then, by clicking on the query society, it will zoom out to show the documents returned in response to his query as shown in Fig. 7.2.

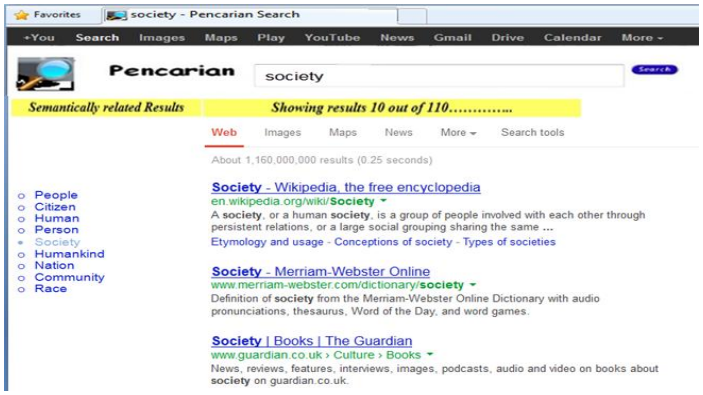

Fig.7.2 Pencarian Search Result Interface for semantically related query "society"

8 CONCLUSION AND FUTURE SCOPE

Since the last decade, there is rapid development of information technology and the rise of the internet and the Web has revolutionized the way people use and access information. Within few years, WWW has entered into the state of information overload from the age of information deficiency. This has given birth to a new, exiting domain of research referred to as Web Mining or Web Information Retrieval. While search engines have achieved quite good results in delivering answer for well formulated, precise queries, they have been less efficient as a tool for providing the efficient access and navigation on the web search results, which are presented to the user. So, recommending the user with semantically similar search results in a very compact and concise way by using term similarity and clustering algorithm may be the solution for the problem.
In this dissertation, an optimization system is proposed which also allow user to navigate and explore the number of queries which are semantically similar to the query input by him on search interface. Unlike the traditional search engine, in which user has to go through the large collection of web document presented as a search results for the query given by user.

This section lists the limitations of the Proposed Optimization System and the work which can be conducted in the future to improve the current system design.

- The limitation of the lexical resources such as WordNet is low coverage, which makes evaluation more difficult. Semantic similarity between words changes over time as new words are constantly being created and new meaning is also being assigned to the existing words. So, there is a scope of using NLP (Natural Language Processing) to make it much better and appropriate.
- The Proposed Optimization System is evaluated for a single term query, as the number of terms in the query increase, the complexity of the system will becomes very high. So in future, a lot of work can be done to reduce this complexity.

- There is very large size of the databases (in particular, of the WWW). Therefore, the clustering algorithms must be very efficient and scalable to large databases. So, any different clustering technique can be used in future.
- The Similarity between the terms can also be fined using various other different approaches, so as to minimize the space and the time complexity of the Term Similarity Analyzer.
- The Data Structure of Index Repository can also be modified in such a way so as to better represent and link the similar terms.

REFERENCES

[1] Garcia-Molina, Hector. Searching the Web, August 2001
[2] Steve Lawrence and C. Lee Giles. Accessibility of information on the web. Nature, 400:107{109, 1999.
[3] Krishna Bharat and Andrei Broder. Mirror, mirror on the web: A study of host pairs with replicated content. In Proceedings of the Eighth International World-Wide Web Conference, 1999.
[4] Liu, X. (2010) 'Efficient maintenance scheme of inverted index for large-scale full-text retrieval', Second IEEE international conference on Future Computer and Communication (ICFCC). 21-24 May, 2010. Wuhan, China.
[5] Zobel, J. and Moffat, A. (2006) 'Inverted files for text search engines'. ACM Computing Surveys, Vol. 38, Issue 2, Article 6.
[6] Neelam Duhan,A.K.Sharma(2011)' A framework for utilising usage trends in the crawling and indexing process of search engines'. Published in International Journal of Knowledge and Web Intelligence, Volume 2 Issue 4.
[7] G. Salton. Automatic Text Processing. Addison-Wesley, New York, 1989.
[8] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In AAAI-2000: Workshop on Artificial Intelligence for Web Search, July 2000.
[9] D.A. Cruse. Lexical Semantics. Cambridge University Press, Cambridge, UK, 1986.
[10] George Davidson, editor. Thesaurus of English words and phrases: 150th Anniversary Edition. Penguin Books, London, UK, 2002.
[11] Donald L. Emblen. Peter Mark Roget: The Word and the Man. Longman Group, London, UK, 1970.
[12] Christiane Fellbaum, editor. WordNet: An Electronic Lexical Database. MIT Press, Cambridge, MA USA, 1998
[13] Ugo Scaiella, Paolo Ferragina,Andrea Marino, Massimiliano Ciaramita.Topical Clustering of Search Results. WSDM'12,Seattle, Washington, USA (2012)
[14] Hearst, M. A., and Pedersen, J. O. Reexamining the cluster hypothesis: Scat-ter/gather on retrieval results. In Proceedings of SIGIR-96, 19th ACM InternationalConference on Research and Development in Information Retrieval (ZÄurich, CH, 1996),pp. 76-84.
[15] Zamir, O., and Etzioni, O. Web document clustering: A feasibility demonstration.In Research and Development in Information Retrieval (1998), pp. 46{54.
[16] Zamir, O., and Etzioni, O. Grouper: a dynamic clustering interface to web searchresults. Computer Networks (Amsterdam, Netherlands: 1999) 31, 11-16 (1999), 1361-1374.
[17] Weiss, D. A clustering interface for web search results in polish and english. Master's thesis, Poznan University of Technology, Poland, June 2001.
[18] Osinski, S. An algorithm for clustering of web search result. Master's thesis, Poznan University of Technology, Poland, June 2003.
[19] Wroblewski, M. A hierarchical www pages clustering algorithm based on the vector space model. Master's thesis, Poznan University of Technology, Poland, July 2003.
[20] Zhang, D. Towards Web Information Clustering. PhD thesis, Southeast University, Nanjing, China, January 2002.
[21] Adam Schenker, Mark Last, A. K. Design and implementation of a web mining system for organizing search engine results. In Data Integration over the Web (DIWeb),First International Workshop, Interlaken, Switzerland, 4 June 2001. (2001), pp. 62-75.
[22] C. Carpineto, S. Osi´nski, G. Romano, and D. Weiss. A survey of web clustering engines. ACM Comput. Surv.,41(3):1–38, 2009.