Journal of Computer Science 9 (11): 1602-1617, 2013
ISSN: 1549-3636
© 2013 Science Publications
doi:10.3844/jcssp.2013.1602.1617 Published Online 9 (11) 2013 (http://www.thescipub.com/jcs.toc)

A CONTEXT-BASED TECHNIQUE USING TAG-TREE FOR AN EFFECTIVE RETRIEVAL FROM A DIGITAL LITERATURE COLLECTION

Muthuraman Thangaraj and Vengatasubramanian Gayathri

Department of Computer Science, Madurai Kamaraj University, Madurai, India

Received 2013-08-07, Revised 2013-09-24; Accepted 2013-10-05

ABSTRACT

The increasing growth of information in online digital libraries causes an increasing need to develop techniques to retrieve. In the digital library, findability-finding the user required information is a hectic task than those of usability. The major issues in findability are (a) topic diffusion: results of a traditional keyword based search, often leads to multiple topic areas, some of which are not interested to user; (b) lack of scoring mechanism: at present, digital libraries lack effective and accurate publication rankings. Thus the users are forced to scan a large result set, which leads them to miss the important ones; providing accurate publication scores can help users in reducing the time spent in searching and (c) selecting search keywords: users spend more time to choose their search keywords, which will express their information need. This study proposes TAG, a new context based retrieval technique that controls the topic diversity and overcomes the above mentioned issues effectively. Using IEEE publications as the test bed and IEEE thesaurus terms as context, our experiments indicate that the proposed retrieval technique effectively produces output results and considerably reduces the resultant set.

Keywords: Context-Based Search, Literature Collection, Topic Diffusion, Publications Ranking, TAG-Tree, Information Retrieval

1. INTRODUCTION

The digital library is an electronic library where the information is acquired, stored and retrieved in digital form. These libraries have diversified collection of information resources such as full texts of journals, conference papers, CD-ROM databases, thesis and dissertations, e-journals, e-books, examination papers, manuscripts and these are available to the users at any time. Many academic libraries, includes not only the familiar books and journals of the general collections, but many rare and unique materials.

Each year sees the introduction of new digital libraries promoted as valuable resources for education and other needs. Digital libraries offer diverse information resources in digital format. Traditionally, libraries have been warehouse of knowledge providing information services to the users.

Ranganathan (Abideen and Srivathsan, 2004), the father of library science has rightly mentioned' Right information to the right user at the right time in the right form'. It is observed that the features of digital library seem to reflect the vision of Ranganathan. Yet systematic evaluation of the implementation and efficacy of these digital library systems is often lacking, due to the traditional keyword based search.

Digital libraries provide instant access to all information, for all sectors of society, from anywhere in the world. This is simply unrealistic. This concept comes from the early days when people were unaware of the complexities of building digital libraries. Instead, they mostly like a collection of disparate resources and disparate systems, catering to specific communities and user groups, created for specific purposes. They also will include, perhaps indefinitely, paper-based collections.

Corresponding Author: Muthuraman Thangaraj, Department of Computer Science, Madurai Kamaraj University, Madurai, India

Further, interoperability across digital libraries of technical architectures, metadata and document formats will also be possible only within relatively bounded systems developed for those specific purposes and communities.

On the one hand, there seem to have an explosion of information with journals and magazines piling high in the book shelves of libraries. On the other hand, either because of limited knowledge on how to retrieve information or there is an insufficient amount of information available, the number of clients asking librarians for information is steadily increasing.

Any given query may fetch huge number of results. It is obvious that very few results are relevant to the user needs out of the huge set of results even though they contain the keyword. Thus, we need an effective searching technique in digital collection, to produce the best result. The main problem here is, the relation between the terms in the given query (Lamberti *et al*., 2009; Li *et al*., 2007) i.e., the meaning of the entire query is missing. Thus it is needed to consider the query as the contexts instead of considering just as keywords. Not only the keywords, but the synonym of it also plays an important role in the searching era.

These high growth rates introduced several challenges facing the information access capability of digital libraries. Some of the challenges that motivated the research work presented in this study are (a) large sizes and topic diversity of search output results; (b) lack of effective scoring functions for publications; (c) lack of effective scoring functions for search outputs (Bani-Ahmad, 2008); (d) supporting example-based search queries; and (e) scalable search-keyword suggestion to users.

The remainder of the study is organized as follows: Section 2 is devoted to the issues relevant to searching in the digital collection. In Section 3, we describe the working mechanism of TAG. Section 4 shows our performance evaluation result. Finally, Section 5 presents conclusion.

2. RELATED WORK

As the fabulous growth of the digital library in each year, the problems with indexing and searching a digital library is increased in a high rate. There are many digital literature systems that produce results based on the importance of the query keyword. These systems do not use contexts to organize search results.

In contextual web search approach, e.g., Y!Q Contextual Search (Kraft *et al*., 2006) and IntelliZap (Finkelstein *et al*., 2004), a context is captured around the user-highlighted text, from which queries are created. The users can specify contexts of interests before viewing search results and no structural and hierarchical information are used. Sometimes user need not give keyword to initiate the search, e.g., in (Coppola *et al*., 2010), according to the environment variables, contexts are selected automatically. Results are retrieved for the set of predefined query based on the corresponding context. The user can select from the list of results that are generated automatically.

A variety of categorization techniques, classification and clustering are proposed that will make the results more understandable. Scatter/Gather (Hearst and Pedersen, 1996) was one of the first clustering systems on top of the Information Retrieval engine, in which it groups documents based on the similarities in their contents. Grouper (Zamir and Etzioni, 1999) uses Suffix Tree Clustering (STC) that identifies sets of documents sharing common phrases. Lingo (Osinski and Weiss, 2004) uses Singular Value Decomposition (SVD) to find meaningful labels for the clusters. Findex (Kaki, 2005) seeks frequent words from the results to classify them. SemreX (Jin and Chen, 2008), a semantic overlay for desktop literature/document retrieval in peer-to-peer networks. Similarly other techniques like fuzzy systems, support vector machine are also used to cluster documents (You and Hwang, 2008; Saracoglu *et al*., 2007).

Similarly to improve search experience some systems use classifications of documents. In (Campbell *et al*., 2007), documents are classified based on the user's background information. Similarly in (Isa *et al*., 2008) Bayes formula is used to identify to which predefined group, this document belongs to. But if a single keyword represents multiple contexts, then this system will produce highly inaccurate results.

If these categorizations are done in online, then the most relevant document may not appear in the top of the result set, also partially relevant documents may be scattered around the list. Mostly search systems are based on the importance of the papers and/or the existence of the keywords. They do not give much importance for the context.

For checking the existence of the keyword, similarity techniques like Text-based (Chen and Chiu, 2010), Google based (Cilibrasi and Vitanyi, 2007; Aliguliyev, 2009) similarity is used. Even though there are many techniques are available, still the end users are struggling to get the desired information. Because, in a keyword - based search, the main ambiguity is that, a single word may have different meanings, where as different words may also refer to the same thing. Thus we need to search by considering the context of the given query.

In Context Based Search (CBS) (Ratprasartporn *et al*., 2009), during pre-querying, publications are assigned into pre-specified ontology-based contexts and query-independent context scores are attached to papers with respect to the assigned contexts. When a query is posed, relevant contexts are selected, search is performed within the selected contexts, context scores of publications are revised into relevancy scores with respect to the query at hand and the context that they are in and query outputs are ranked within each relevant context. The major drawback in this system is that for searching within each selected context, all the publications in the database are verified linearly. Thus it takes more number of comparisons and which in turn increases the retrieval time.

As an alternate Search-and-Distribute-to-Contexts (SDC) approach is also handled here in order to utilize the context information. In SDC, the same strategy is followed as in CBS, to assign papers to Contexts and to compute the context scores of each paper. When a query is given, unlike CBS, it first performs a keyword-based search, across all the publications from which it finds the contexts and publications that falls in. Then re-ranks the publications within each located contexts. Since the query is matched against the whole database, increases the computation overhead. The meaning of the query is not conveyed, because of the keyword-based search.

To overcome these issues New Context-Based Search (NCBS) (Thangaraj and Gayathri, 2011a) uses its searching structure to hold the contexts and its synonyms along with its publications. The searching structure is a combination of B^+-tree and inverted list. Contexts are extracted from the documents in the corpus using pattern extraction based techniques. All the documents in the corpus are classified based on the context regardless of the query and are mapped into the NCBS structure, has a combination of B^+-tree and Inverted List.

When a query is given, the relevant context is identified and returned with its synonym as well as the appropriate document of the context. The main drawback of this method is it can search only with the context, not with its synonyms. It will just return the list of synonyms. Thus improved version NCOSBS (Thangaraj and Gayathri, 2011b) will search both in Context and its synonyms. The data structures used are B^+-tree and hash table. Here it searches contexts first in the context tree; if available then proceeds searching to its publications and finally its related synonyms are returned. If it is not available then, the relevant hash table's look up is done; find its relevant synonyms tree, it is where the given query is searched against synonyms. The major drawback of this system is to search either context or synonyms and not in a combined form.

3. TAG ARCHITECTURE

A new architecture called as TAG is formulated to address these issues. TAG uses Context-Based Search, in which the query can be done using the keywords and its synonyms.

The various functional components of the TAG architecture are:

- TAG Extractor: Parts of the Publications are extracted from digital collection, for the construction of Contexts and for indexing
- TAG Indexer: Publications are indexed based on the context. Publications that match the particular Context are mapped in the TAG-tree. Publications are assigned the first level scoring
- TAG Suggester: Helps user to select right terms for the query with the help of usage history
- TAG Retriever: Retrieves the relevant publications based on each Context that are relevant to the given query with the help of Thesaurus. Also next level scoring is assigned to the publications
- TAG MRanker: Publication results of various Contexts are merged. Finally based on the different levels of scoring, publications are ranked, which in turn passed to the users

Note that the first two tools of TAG are independent of query and pre-executed. The remaining tools need the query as an input and executed at on-line. The overall architecture of TAG is shown in the **Fig. 1**.

Using the TAG Extractor, publications in the database are pre-processed. Then Contexts are identified from the publications using extraction. Then these publications are mapped to the TAG-tree based on these identified Contexts, by the TAG Indexer. TAG Suggester parses the publications at offline. By this information at background, TAG Suggester suggests the right terms for constructing the query. Once the query is given by the user, then TAG Retriever retrieves the relevant Contexts from the TAG-tree, at the same time, relevant results are retrieved from the previous search log. From these result sets, publications are retrieved along with its scores. Now using the TAG MRanker, both the result sets are merged and ranked based on the scores of the publications. These ranked list of publications, are then returned to the user as the final result set.

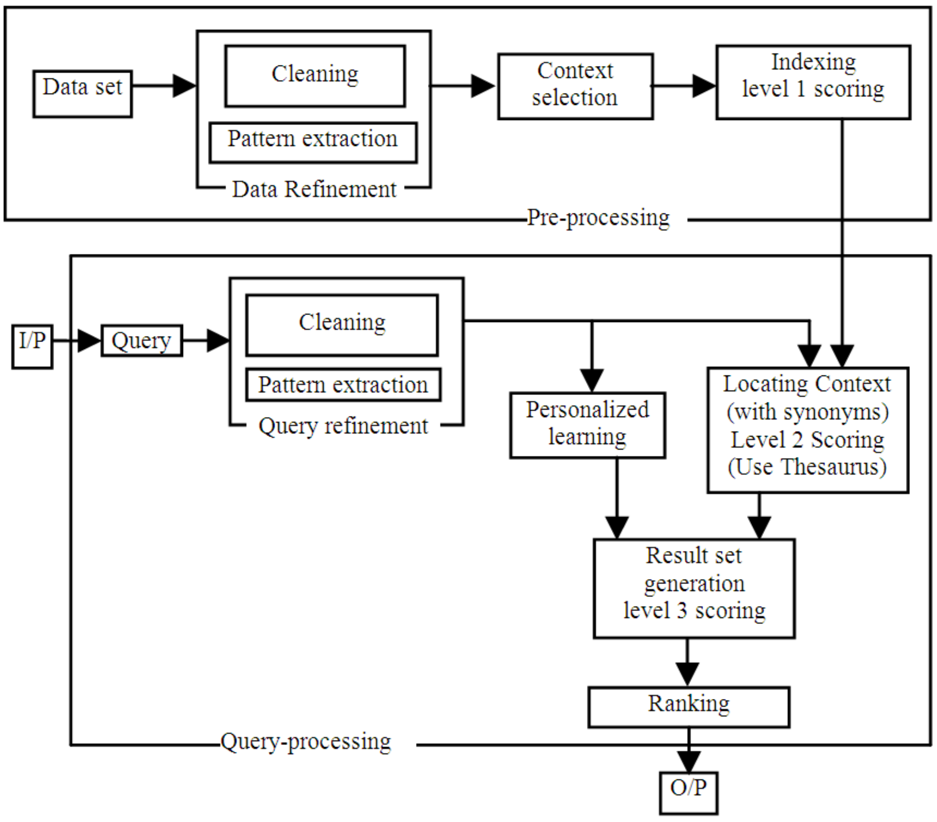


Fig. 1. TAG architecture

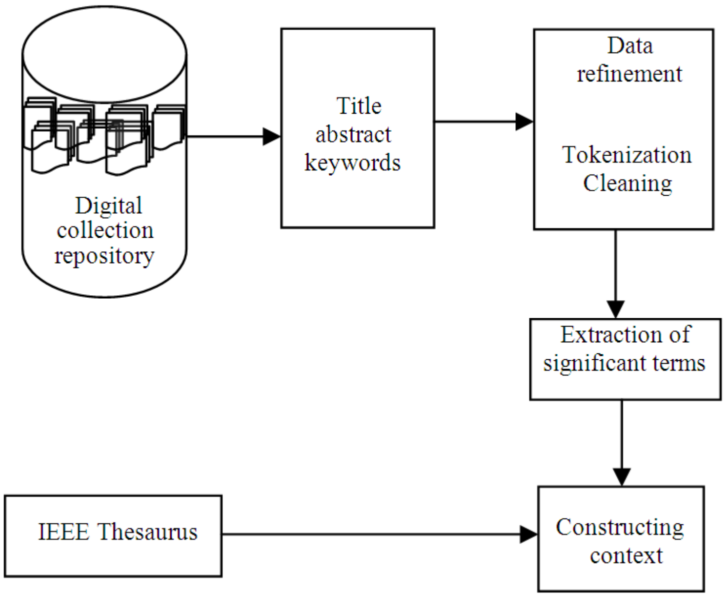


Fig. 2. Workflow of TAG extractor

3.1. TAG Extractor

This tool is used to extract Contexts from the digital collection. Based on these Contexts, the publications are categorized. The workflow of this TAG Extractor is depicted in **Fig. 2**.

It is advantageous to parse the two areas such as publication title and abstract of publication: **(a) publication titles** since (i) the number of tokens in a title are an order of magnitude less in count than the tokens of the full document and (ii) publication titles are significantly less likely to have ambiguous tokens (like impersonal pronouns) than the full document even though, in rare occasions, authors choose for their articles humorous, but irrelevant names. Such titles are humorous and easy to be remembered by users and they have great value in navigational queries in which the user has a particular target. On the other hand, these titles negatively affect the performance of informational queries, in which the user is looking for sources that provide background knowledge about the search topic (Lee *et al*., 2005). To solve this approach, we also suggest preprocessing **(b) abstracts of publications** in addition to titles and **keywords** are also extracted.

These extracted parts of the publications are then tokenized. These tokens are cleaned, by the process such as stop words removal. Terms from IEEE Thesaurus are used as Contexts. In addition to that significant terms of publications are also considered to best define the Context (as in NCBS).

Briefly, a Context is in a pattern form, which consists of three tuples <prefix>, <context> and <suffix>. Significant words are assigned to <context> tuple, where as the words surrounding the significant words are assigned to <prefix> and <suffix>.

3.2. TAG Indexer

Contexts based on which the publications are categorized, are constructed using TAG Extractor. Now this section shows how the publications are assigned to these Contexts. TAG-tree is a combination of B^+-tree and list as shown in **Fig. 3**. At first TAG-tree is constructed and the Contexts created by the TAG Extractor are then mapped to it. Finally the publications are assigned to its relevant Contexts.

The TAG-tree is organized based on the contexts with its prefix and suffix terms. The leaf node has the Context and a pointer to the relevant document. Every leaf node of a B^+-tree points to a synonyms list. The synonyms list has the set of synonyms for the given context. Each context is mapped into the TAG-tree as an individual bucket element. In the internal nodes of the TAG-tree, it has only the Context, that is, pattern with the three tuples <prefix><context><suffix>. But, the leaf node has Context, its Cluster information and a pointer to a list that holds synonyms (refer NCBS for more information).

The Context taken in this study is nothing but thesaurus terms. It is better to find a way to determine a relationship between the term and each publication and decide whether the publication should be categorized to the term. Expert intervention is needed for the effectiveness categorization when the number of publications and contexts are small. However, the number of contexts and publications are very large. Manual assignment is not practical and also very time-consuming.

To automatically assign publications to Contexts, the existence of the Contexts are verified in the publications. First, the context terms in the publications are highlighted. Then all the synonyms of the Contexts are also highlighted, by refining once again, now the publications containing the Context patterns are added to the respective publication cluster called P-Cluster. Publications of the P-Clusters are assigned scores based on the relevance between the publications and their respective context.

3.3. TAG Suggester

Studies show that users spend considerable amounts of time in search sessions to properly select keywords and to modify their search keywords in order to successfully locate publications. A search-keyword suggester may help users choose keywords properly and thus, users are less likely to face unsuccessful search attempts.

TAG Suggester is based on the prior analysis of the publication collection at hand. The working mechanism of TAG Suggester is depicted in **Fig. 4**. Initially publications are parsed using the Link Grammar Parser (LGP), is a syntactic parser of English. As stated in the previous section, the three important parts of the publications are used for parsing. The linkages between the tokens of the publications are stored in the LGP Database along with the parsed tokens. Parsing is pre-executed and not dependent on queries.

When the user starts typing the Context keyword, Token Predictor (TP) is called to make suggestions on the first few letters given by the user. But the suggestion scope of TP is reduced based on the terms fed already in the current session. When the user starts typing, TP fetches the LGP database for the tokens which starts with the given keyword letters. In addition, it fetches the usage history for the tokens. If this is not the first call to TP for the current session, then TAG-tree is also searched with the already completed Context keyword.

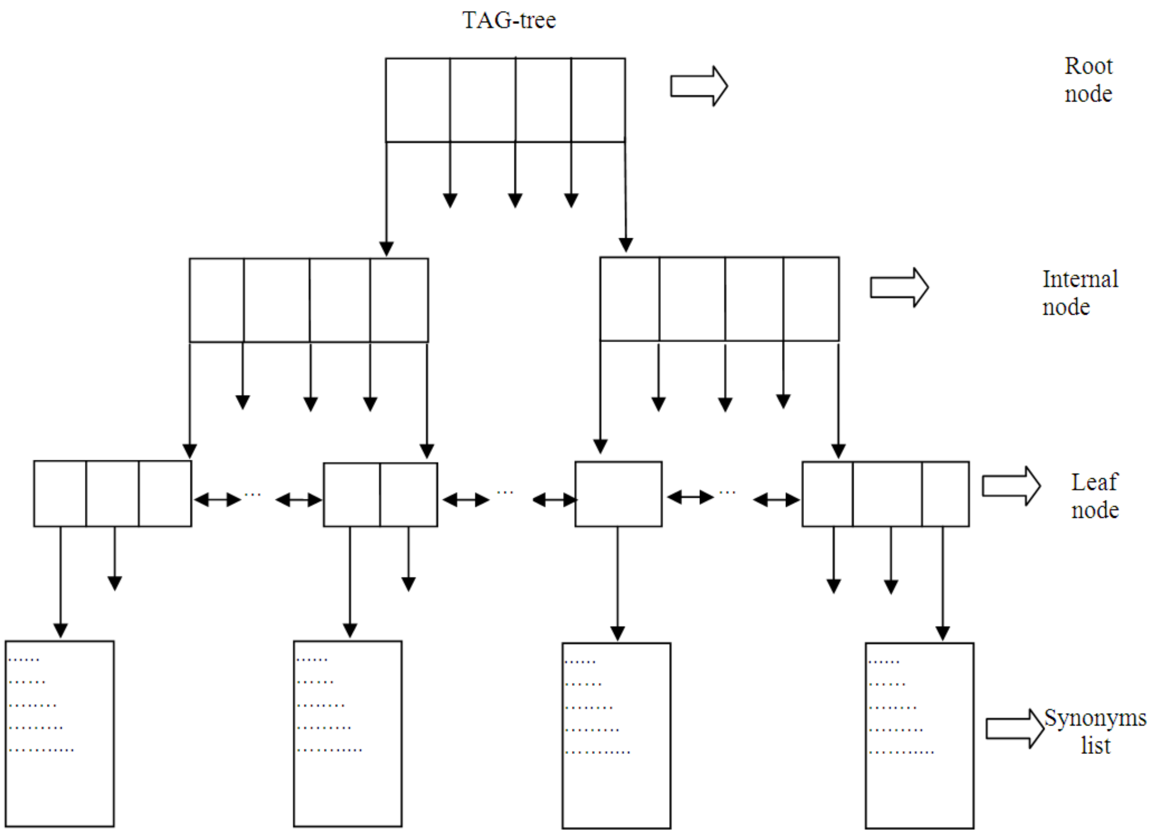


Fig. 3. TAG-tree an overview

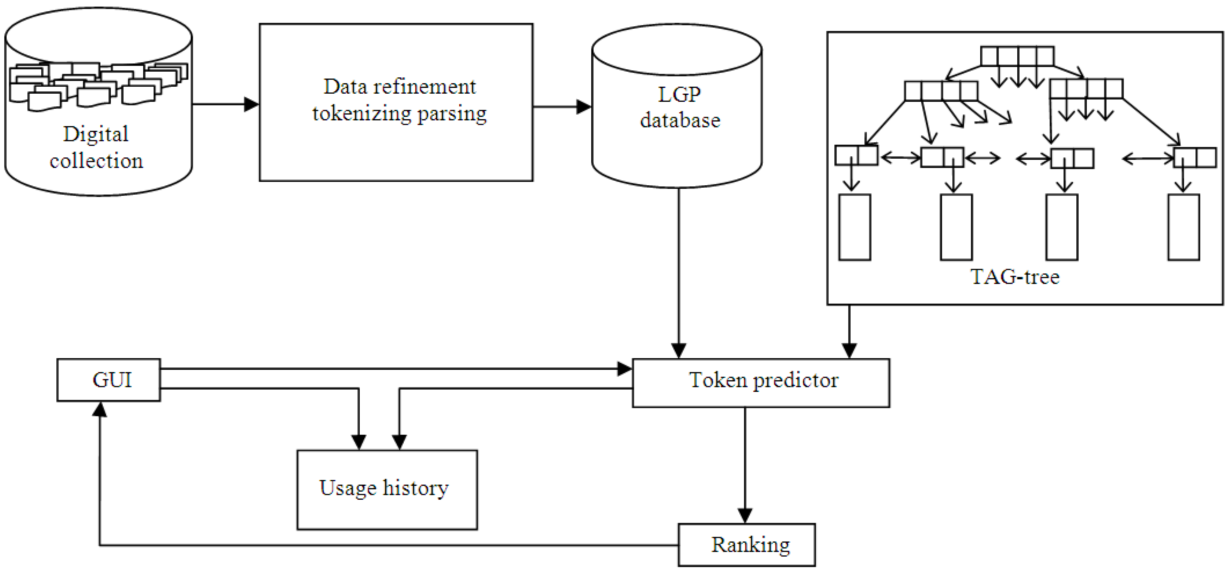


Fig. 4. Work flow of TAG suggester

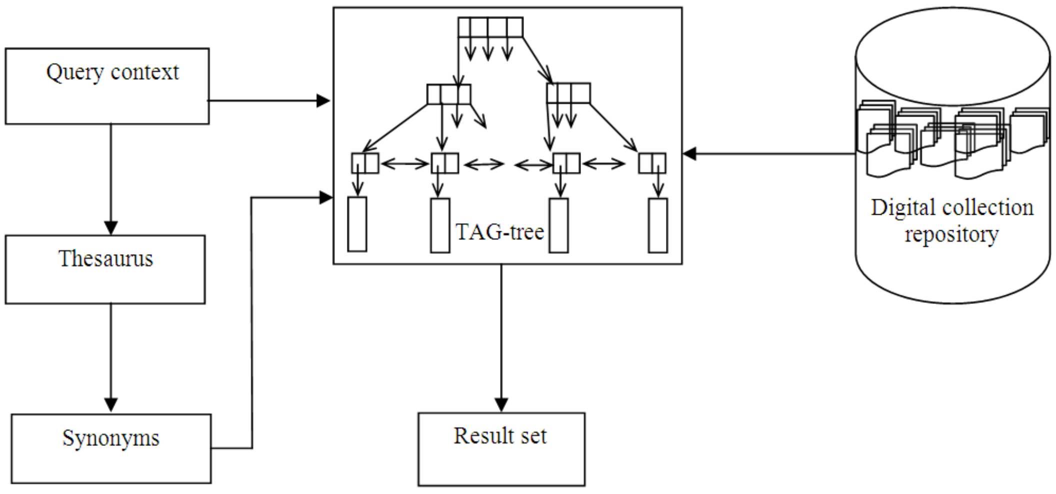


Fig. 5. Workflow of TAG retriever

In all these fetches, scores are allotted to the results based on their relevancy/importance. Finally TP will pass all these results to ranking module, which will arrange and select the most appropriate (limited) suggestions for the search keyword.

3.4. TAG Retriever

This tool is mainly for retrieving the publications that are relevant to the given query. When the user successfully chooses his query Context with the help of TAG Suggester, then the TAG Retriever is invoked by passing the query Context as input. The workflow of TAG Retriever is shown in **Fig. 5**.

The TAG-tree is searched against the given query Context. Initially, the prefix tuple is searched: If it is available, then the subsequent tuples are searched in its sub tree; otherwise, searching is performed with the next tuple. Similarly each tuple is considered for searching, when it is found, then the further searching is done at its sub tree; otherwise searching continues with the next tuple.

When a query tuple is searched in the tree, it may appear at the <prefix> tuple or at the <context> tuple of the node in the tree. Mostly it appears at the <context> tuple.

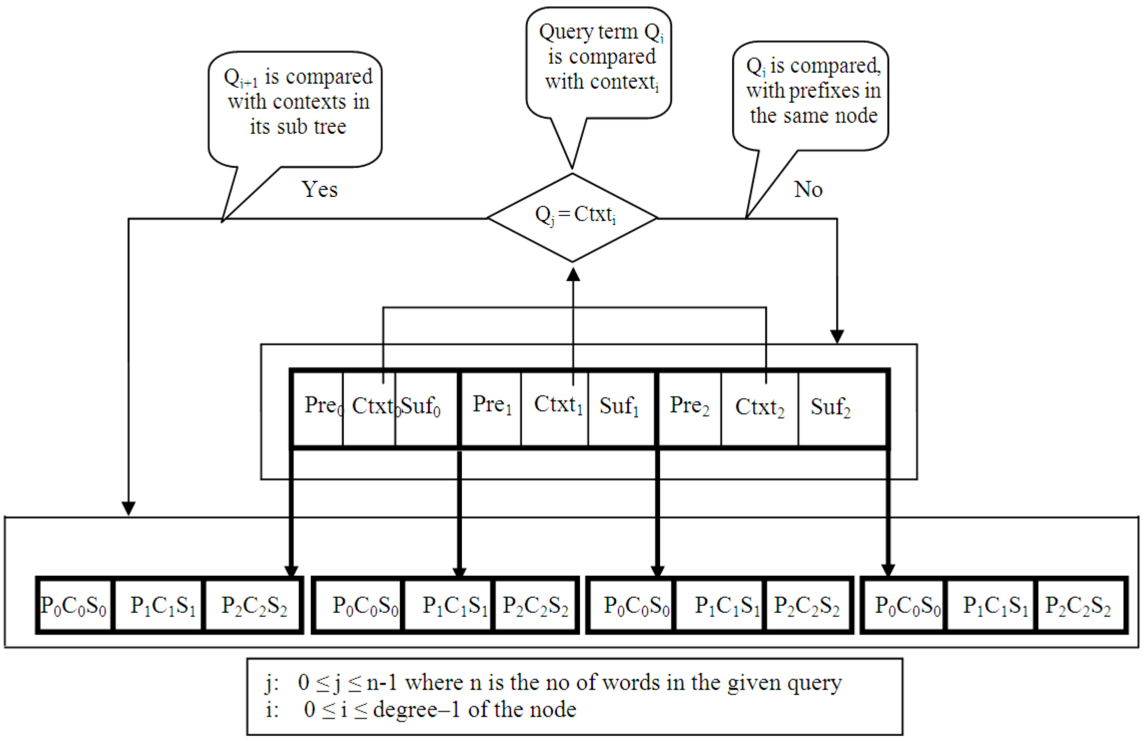
Thus searching starts with the <context> tuple of the node, if it matches with the query tuple, then the subsequent tuples are searched in its sub tree (**Fig. 6**). In contrast, if the query tuple doesn't match with <context> tuple, then it is compared with the <prefix> tuple of the same node (**Fig. 7**). Suppose, the query tuple matched with the <prefix>, then the searching for the next tuple is done with the <context> tuple of the same node. Then, the searching mechanism repeats the searching in the same fashion to retrieve the required information.

When searching is stopped without getting the exact context as in the query, then the Context, up to which the search mechanism found its match with the query context, is returned as a result. When there is no match occurs, then the Contexts in the root of the Context tree are returned as a suggestion for the user's reference. Instead of getting out with empty result set, the user can get some information to make improvement in their searching query task.

Once the relevant Context is found, then its P-cluster is retrieved from the digital collection repository. These publications are added to the result set along with its score (which is assigned during indexing). Then the thesaurus is fetched for the synonyms/Relevant Terms (RT) for the given query Context. These synonyms are then searched in TAG-tree. P-clusters of the synonyms' Contexts are also be added to the result set along with its score. Now second level scoring is assigned based on the relevance between the selected context and the query context. Finally TAG Retriever returns the publications along with its two levels of scoring.

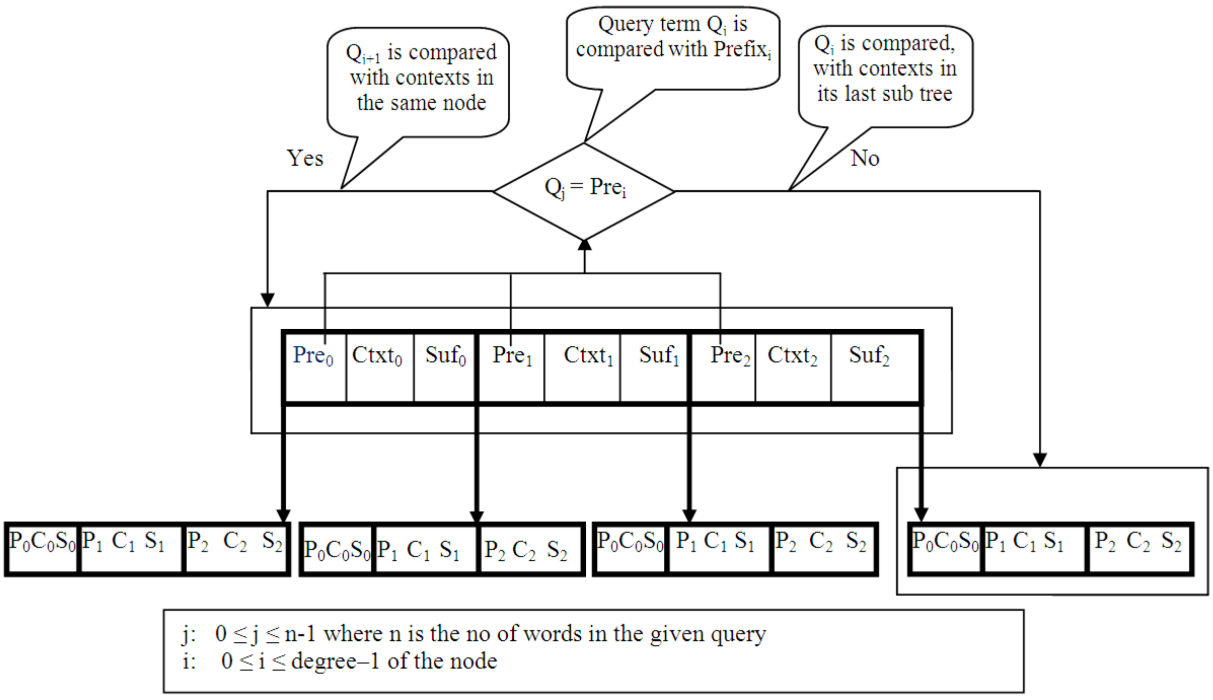
3.5. TAG MRanker

TAG Merge Ranker (TAG MRanker) is used for merging all the result sets of publications and ranks it based on their score. Different levels of scoring are assigned to the publications. Finally all these result sets (at different levels) are merged. Based on the scores, publications are ranked and returned to the user.



Query term Q_j is compared with $context_i$
Q_{j+1} is compared with contexts in its sub tree
Q_j is compared, with prefixes in the same node
Yes
No
$Q_j = Ctxt_i$
Pre_0 $Ctxt_0$ Suf_0 Pre_1 $Ctxt_1$ Suf_1 Pre_2 $Ctxt_2$ Suf_2
$P_0C_0S_0$ $P_1C_1S_1$ $P_2C_2S_2$
j: $0 \le j \le n-1$ where n is the no of words in the given query
i: $0 \le i \le degree-1$ of the node

Fig. 6. Searching against the contexts of the contexts



Q_{j+1} is compared with contexts in the same node
Query term Q_j is compared with $Prefix_i$
Q_j is compared, with contexts in its last sub tree
Yes
No
$Q_j = Pre_i$
Pre_0 $Ctxt_0$ Suf_0 Pre_1 $Ctxt_1$ Suf_1 Pre_2 $Ctxt_2$ Suf_2
$P_0C_0S_0$ P_1 C_1 S_1 P_2 C_2 S_2
j: $0 \le j \le n-1$ where n is the no of words in the given query
i: $0 \le i \le degree-1$ of the node

Fig. 7. Searching against the prefixes of the contexts

While constructing P-clusters, TAG Indexer assigns level 1 score to the publication. This score is based on the relevance between the Context and the publication. Level 2 scoring is assigned by the TAG Retriever. The publications of the real query Context is assigned score more than the publications that belong to the synonyms Contexts. Thus the relevance between the given query Context and the identified Context are considered. Hit ratios of the publications are used to assign level 3 scoring.

The user may utilize the suggestions made by TAG Suggester. Since the suggestions are made from the available digital collection repository, the suggestions are clearly right Contexts. Level 4 scoring is assigned to the publications based on the selection of the query. If the user has selected the query from the suggestions made by the TAG Suggester, then the chosen context's P-cluster are assigned higher scores than the others. All these scores of the publications are summed up to produce the overall score of the publications. Thus publications of the various result sets are merged. Based on the overall scores, publications are ranked, which in turn passed to the users.

3.6. TAG Algorithms

This section presents the various algorithms used in TAG technique. The algorithm for constructing P-clusters of each Context is given below:

```
Algorithm                 : Paper-Context Mapping
Data Structure            : See Fig. TAG Structure
Input     : t – threshold
            Context ctxt[] – set of Contexts
Output    : Pset[] – An array of mapped Context-papers

i: $1 \leq i \leq m$ (m – Total number of Contexts)
j: $1 \leq j \leq n$ (n – Total number of Papers in the database)

for each Context $ctxt_i$
   for each Paper $p_j$ in the database
      if $p_j$ contains $ctxt_i$
      {
         $p_j$.score = SimScore ($ctxt_i$, $p_j$)
         if $p_j.score \geq t$
            add $p_j$ to Pset[i]
         }
```

Publications in the digital collection repository are searched for the Context. If the publication contains the pattern of the particular Context, then the similarity between the Context and the publication are computed, which is assigned with the score of it. In the same way, scores for all publications are computed for a particular Context. If the score is greater than or equal to the threshold value, then the publication is added to the P-cluster of the specific Context. In this way, for all Contexts, P-clusters are constructed.

The algorithm used for locating the Contexts is explored in the following. If a query is given, then it is searched against the TAG-tree. The resultant Context's P-cluster along with its score is added to the Result Set. Now synonyms of the identified Context (s_i) are found:

```
Algorithm         : Retrieval
Data Structure    : See Fig. TAG Structure
Input             : Query q, Node nodepointer
Output            : RPset – Resultant Publications
Let RPset-Resultant publications id with its scores
    result-node of Context Tree
    Syn-node of Context Tree
find(q,head)-Context Identification function

ConSynRet(Query q, Node nodepointer)
{
    result = find(q, head);
    add Pids, scores of result to RPset; //Level1 Score
    //Insert Pid and level 1 score alone,
    // keep other scores as 0
    for each synonym $s_i$ of result
    {
        syn = find($s_i$, head);
        add Pids, scores of result to RPset; //Level2
Score
        //if Pid already exists, only update level2 score
        //else insert Pid and level2 score alone &
        //keep other scores as 0
        }
    }
```

Each synonym is searched against the TAG-tree and its P-clusters are also added to the Result Set with its scores. The algorithm for searching against the TAG-tree for the given input is shown below:

```
Algorithm         : Context Identification
Data Structure    : See Fig. TAG Structure
Input     : Query q, Node nodepointer
Output    : Context with its document as well as related
synonyms

Let i, $0 \leq i <$ degree of the Context Tree;
j, $0 \leq j \leq 2$ (j – each tuple in the query);
```

x – node of Context Tree;
docs-id– Ids of relevant publications of the Context;
<P_i>-Prefix tuple of the ith node segment of the nodepointer;
<C_i>-Context tuple of the ith node segment of the nodepointer;
<S_i>-Suffix tuple of the ith node segment of the nodepointer;

```
find (Query q, Node nodepointer)
{
   if (nodepointer is NULL) return (NULL);
      if (nodepointer is a leaf)
      {
         x = find (q, nodepointer);
         return(x, docs-id, Synonyms );\\x is the context
         }
      else
      {
    for each tuple <q_j> in the query context
        for i = 0 to degree-1
        {
           if (<q_j> = <C_i>)
           {
           x = find (q_{j+1}, nodepointer -> child_i);
           if (x = NULL)
              return (current Context from Tree Leaf);
           return (x);
           }
           else if (<q_j> = <P_i>)
                return (find (<q_{j+1}>, nodepointer));
           }
        \\ if q_j is not available in any of the node
        \\ segments in nodepointer, then, search for it
        \\ in the last subtree
        x = find(<q_j>, nodepointer -> child_j);
        if (x!=NULL) return (x);
        }
    return headnode;
    }
```

Each tuple in the query Context is searched against the TAG-tree. First the <prefix> tuple is searched, if it is found, then its sub tree is searched for the remaining tuples. If a tuple is not found in the buckets of the TAG-tree node, then it is searched in its last sub tree. In case, the current tuple is not found even there, then the next tuple is searched for. In this way, the TAG-tree is searched for the given query tuple. In worst case, if any of the query tuple is not found, at last the head node of the TAG-tree is returned as the suggestions to the user to refine his query. This may be encountered, if the user doesn't consider the suggestions made by the TAG Suggester.

The algorithm of TAG-Suggester is depicted in the following. When the user starts entering the query keyword, TAG Suggester has to give appropriate suggestions to complete the query. User entered input (W) is parsed; the completed (CSK) and Uncompleted (USK) keywords are identified first. If the completed keywords are not available i.e., this is the first trial of the current session, then LGP database is searched to find the tokens that starts with USK; as well as previous query Contexts are also be searched. Then the results of both of these are merged; based on the importance, suggestions are ranked and the top-k suggestions were presented to the user:

```
Algorithm          : TAG Query Suggestor
Data Structure     : See Fig. TAG Structure
Input      : String W-Current Search Terms
       Node head-Pointer to the root of TAG-tree
Output    : Suggestions[] - An array of Suggestions

Let SK1, SK2, SK3 are suggestions
find_suggestion(StringW, Node head)
{
    CSK = Completed Search Keyword in W
    USK = Uncompleted Search Keyword in W

    if(CSK = "" && USK !=" ")
    {
        SK1 = findLGP(USK, All)
        SK2 = findLoG(USK)
        W' = SK1 U SK2
        }
    else if(CSK != "" && USK = "")
    {
        SK1 = findTAGtree(CSK, head)
        SK2 = FindLoG(CSK)
        SK3 = FindLGP(CSK)
        W' = SK1 U SK2 U SK3
        }
    else if(CSK != "" && USK != "")
    {
        SK1 = findLGP(USK)
        SK2 = findTAGtree((CSK + USK), head)
        SK3 = FindLoG(CSK, USK)
        W' = SK1 U SK2 U SK3
        }
    return W'
    }
```

If CSK is filled by already fed terms, then TAG-tree is searched against the CSK to produce the suggestions to complete the current query. At the same time, both LGP database as well as the previous history is also be searched. In the LGP database, it finds the possible linkages for the CSK, based on which suggestions from it are made. Finally all these results are merged and ranked to provide top-k suggestions.

If both CSK and USK is available for the current session, then TAG-tree is searched for the Contexts that has tuples like CSK, followed by the tuple starts with USK. In LGP, tokens which start with USK in the possible list of tokens that satisfies the possible linkages of CSK are searched for. Similarly in previous history query Contexts that has CSK followed by the token that starts with USK is searched for. At last all these results are merged and ranked to equip user with top-k suggestions.

4. PERFORMANCE ANALYSIS

This section shows the efficiency of the TAG technique by discussing various experiments. For this study we have conducted many experiments, but presented only vital results. TAG technique is compared with the earlier models such as CBS and SDC in order to explore the strength of TAG. These three techniques were implemented in JAVA. All experiments reported in this section were conducted on Processor Intel Core i7 @ 2.30GHz with 8 GB RAM and 1 TB hard disk, running Windows 7. As a test bed, in our digital collection, 2 Lakhs full text articles were downloaded from IEEE Journals in the field of Computer Science and various Contexts were queried to find the performance.

The objectives of the experiments were categorized as follows:

- Prime purpose is to assess the retrieval performance of the indexing structure
- Study the impact of topic diffusion with large digital collection

4.1. Parameters and Assumptions

In this section, the parameters used in the analysis and the basic assumptions of the study are given in **Table 1**.

Some assumptions made to facilitate our experiments study on this retrieval structure as:

- The searching value should be uniformly distributed
- All the clusters are with adequate information
- Each query should be supported with context for the effectiveness of the retrieval

Table 1. Parameters used in the analysis

Parameter	Meaning
H	Height of the TAG-tree
r	Number of papers with query relevance
s	Stop words
Q	Query
P	Prefix word
C	Context word
S	Suffix word
T	Number of Contexts
L	Key Length
As	Average size of the paper
R	Number of clusters
D	Database size
U	Cluster size
NP_i	Number of publications in the ith cluster
A_t	Number of publications that satisfies the pattern 't'
E_c	Number of patterns for a specific Context 'c'
z	The maximum number of synonyms of a Context

These assumptions are commonly found in all the analytical model for accessing literature collection and they do not affect the relative merits of this search structure.

4.2. Storage Cost

Normally, the storage cost of an index can be expressed as the number of clusters used by the index, divided by the number of clusters that are absolutely necessary to store the instances Equation 1:

$$CC = m \times PT_{ci} \times P_{pt} \quad (1)$$

Where:

m = The number of contexts

PT_{ci} = The number of patterns for each context$_{ci}$ where: ci = $1 \le ci \le m$

P_{pt} = the publications that satisfies the pattern pt where, pt lies in $1 \le pt \le t$.

In all experiments performed, we have obtained that TAG technique have lowest storage cost. However, the storage cost is negligibly small, since large capacity of storage devices are widely available today for lower cost. Therefore, it may be preferable to design models that provide good performance, even if they have a large storage requirements.

4.3. Retrieval Cost

The following is the retrieval cost (RR) for retrieving the result set from TAG-tree Equation (2 and 3):

$$CR = \begin{cases} h + NP_i & \text{if } Q \in B \\ 2h + NP_i & \text{otherwise} \end{cases} \tag{2}$$

$$RR = \left(\left(CR + S_{id}\right) + \sum_{j=1}^{ns} CR_j\right) + cn \tag{3}$$

Where:

CR	=	The cost of retrieving a Context from TAG-tree
B	=	The buckets of the TAG-tree
Sid	=	The cost for identifying the synonyms
Ns	=	The number of synonyms of a particular context, which is $0 \le ns \le z$
CR_j	=	jth context retrieving cost
c	=	The cost for clustering the publications

4.4. Preprocessing and Cluster formation

In section 3.1 and 3.2 Preprocessing and P-cluster formation are discussed. Usually text document contains 30% of usual, generic terms like an, the, where. While indexing if we include all these terms as such, will increase the overhead of manipulating it. The removal of those words will not make any impact on the meaning of the pattern. To avoid computational overhead, various unique stop words are identified and removed from the parsed digital collection.

Many unique patterns can be generated with a different combination of words. **Fig. 8** shows the growth of P-clusters, when the size of the database get increased. The figure depicts that the CBS approach uses more number of clusters when compared to the TAG. Normally, the increase in number of clusters will degrade the search performance. It is inferred that the proposed method performs well compared to the other methods.

The time taken to construct a P-cluster by varying the size of the digital collection is presented in **Fig. 9**. The objective of this experiment is to observe the time variations with respect to change in data size. CBS and TAG methods use pattern-based approaches.

CBS construct more number of clusters based on minimum relevance and scores. As the resultant set of the minimum relevance are high, it takes maximum time to construct P-clusters. In contrast, TAG construct clusters based on exact relevance with score that minimizes the construction time. The SDC method is not taken for the study because the cluster formation is done only after the retrieval. The results show that the proposed method outperforms the CBS.

Thus, the number of tokens that matches the minimum relevance are high, which in turn computation of their scores and further operations are also takes a considerable amount of time. But in case of TAG, only the papers that are exactly relevant are under computation for their relevance score and further operations. Thus the time taken for TAG to construct P-clusters is minimum over CBS. In SDC, clustering is done at the end of the retrieval, not in the preprocessing. Thus clustering phases of CBS and TAG cannot be compared to SDC.

4.5. Retrieval

The retrieval efficiency is a major challenge when the size of the digital collections increases. This study shows the retrieval performance of the different methods. The **Fig. 10** discusses the measuring of retrieval time for different context. For this study, Ten different Context were randomly chosen and size of the digital collection is fixed as 2 lakhs publications.

The graph shows that the TAG retrieval is better than other method, since TAG uses a tree based search where as other method use linear search. TAG uses topic-based minimal search space which is created using <prefix> and <suffix>, is a main reason for effectiveness of the searching.

Though CBS uses pattern-based approaches, i.e., information from <prefix> and <suffix> are used, it performs linear searching. Thus TAG outperforms CBS even both uses pattern-based approaches. CBS finds directly the contexts that are relevant to the query context, where as SDC searches the digital collection as a whole for the query by considering it just as a keyword. Once the result set is generated, then it is distributed to different clusters depending upon the topic to which it belongs. Finally it retrieves the result set for the specific Context from it. Thus searching the whole digital collection and clustering the results, at last extracting the specific topic, takes more time compared to CBS. In this way, CBS outperforms than SDC.

4.6. Accuracy

The performance of accuracy depends on finding exact match for the given Context. To study the performance of accuracy, we have taken 2 lakhs papers with 11 randomly selected Contexts. The relevance of the papers for each selected Context were identified and evaluated. **Fig. 11** shows the accuracy of search in terms of various selected Context.

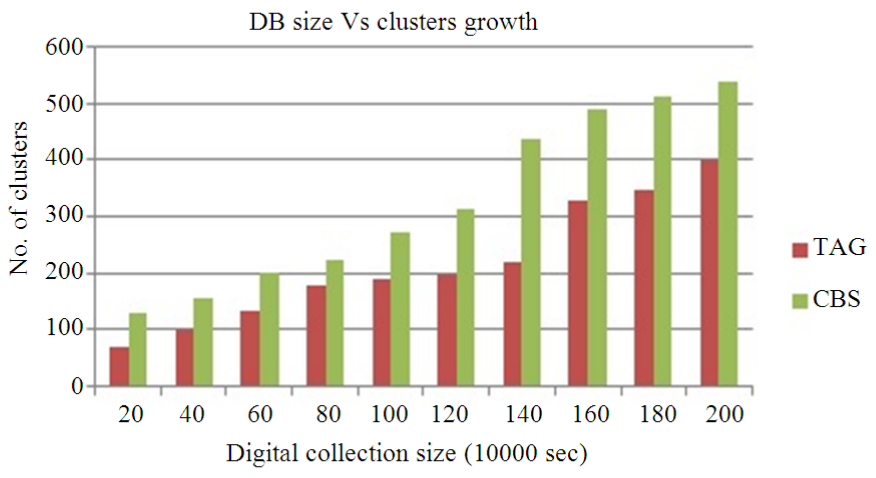


Fig. 8. Growth of Pclusters against the growth of digital

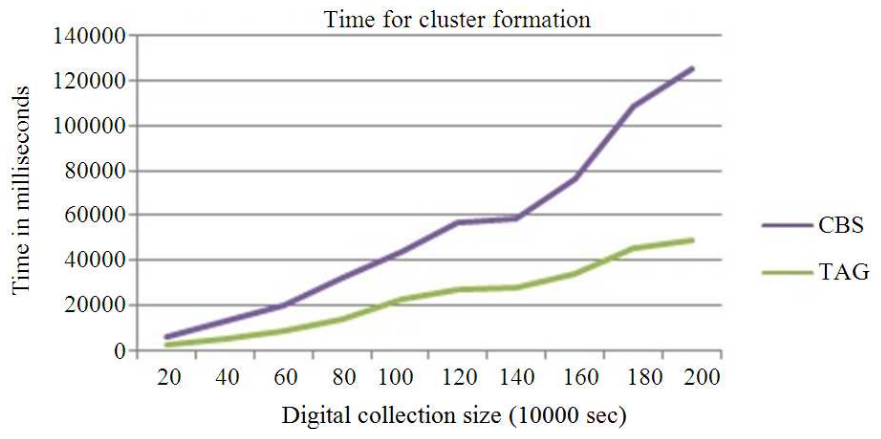


Fig. 9. Time taken for construction of Pclusters

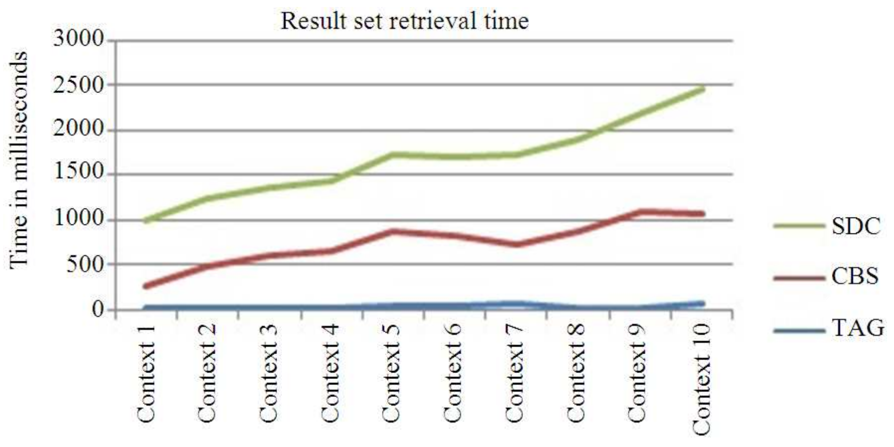


Fig. 10. Time taken to retrieve the result set

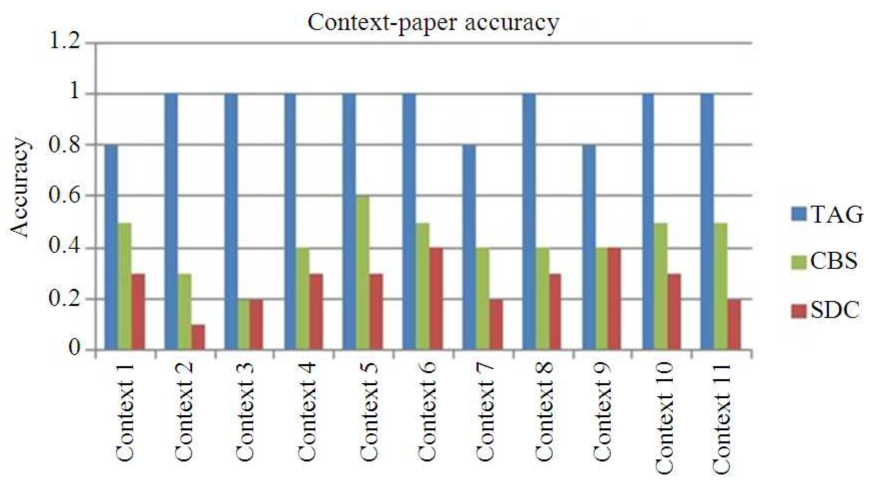


Fig. 11. Accuracy of the retrieved results (direct evaluation)

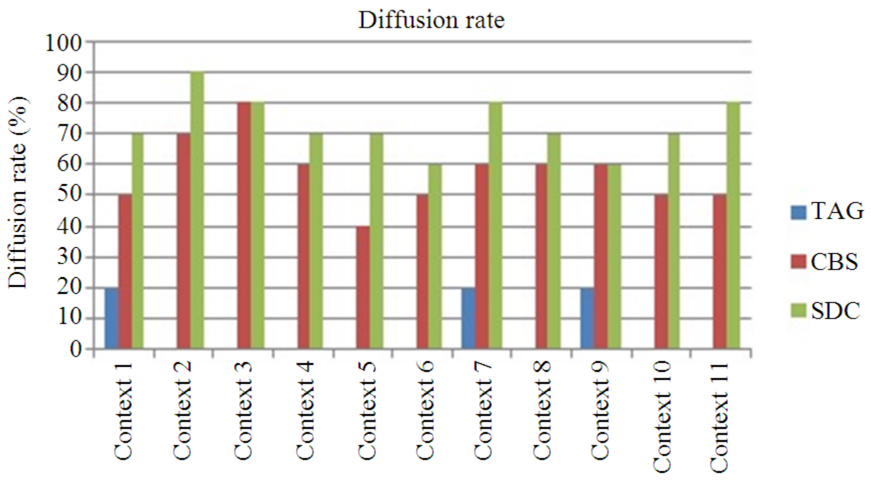


Fig. 12. Diffusion rate of the result set

SDC finds whether the given query terms appears in the article. The query terms may not be available in the article as given in the query. For example, the query context "process scheduling algorithms" is queried, the result set includes the articles of processor design, disk scheduling, deadlock avoidance algorithms.

It just performs AND operation between the results of the individual query terms. Thus results are not that much accurate and size of the result set is very large. In CBS, even the occurrence of the pattern is considered, because of the stemming algorithm used (Porter Stemmer), domain words are not differentiated from the generic words. Thus CBS also not produces accurate results. TAG outperforms than these two earlier techniques, because of the features of TAG-tree, which narrows down the searching area, which brings the relationship that exist between the query terms.

4.7. Diffusion

This study focuses on the key issue of information retrieval, Topic diffusion. Some Contexts were selected randomly to find the publications in the particular Context.

Fig. 12 shows the diffusion rate of the contexts. Since CBS and SDC uses approximate matching. As described earlier, SDC is purely text-oriented, i.e., the relation between the query terms are not considered; in CBS, the stemming algorithm used is not enough to differentiate the domain specific terms from the generic terms; so, both diffuses from the topic of the query. TAG outperforms these earlier techniques, since the searching area is narrowed down and the relationship between the query terms is considered, as well as the query is considered as Context, i.e., the keywords along with their synonyms, also more information about the topic in the form of <prefix> and <suffix>.

4.8. Observation

- The size of the resultant set is reduced about 60%
- Diffusion rate is minimized due to pattern based approach; exact matching and effective cluster formation
- Suggester with maximum specificity makes the relevant query
- The retrieval efficiency of TAG is about 84% as compared to previous searching technique

5. CONCLUSION

This study presented architecture for Context-based retrieval in a digital collection. This study was implemented and compared with the existing methodologies. The performance analysis of these methods shows the effectiveness of the new architecture by controlling topic diffusion problem.

This study can further be extended towards tuning of indexing method, ontology based retrieval and similarity enhancement.

6. ACKNOWLEDGEMENT

This study is a part of Major Research Project (41-642/2012(SR)) funded by UGC, India.

7. REFERENCES

Abideen, S. and Srivathsan, 2004. Convergence of digital libraries with knowledgement. Proceedings of the International Conference on Digital Libraries, Feb. 24-27, TERI India, India, pp: 569-575.

Aliguliyev, R.M., 2009. A new sentence similarity measure and sentence based extractive technique for automatic text summarization. Expert Syst. Applic., 36: 7764-7772. DOI: 10.1016/j.eswa.2008.11.022

Bani-Ahmad, S., 2008. Research-pyramid based search tools for online digital libraries. PhD Thesis, Case Western Reserve University.

Campbell, D.R., S.J. Culley, C.A. McMahon and F. Sellini, 2007. An approach for the capture of context-dependent document relationships extracted from Bayesian analysis of users' interactions with information. Inform. Retrieval, 10: 115-141. DOI: 10.1007/s10791-006-9016-2

Chen, Y.L. and Y.T. Chiu, 2010. An IPC-based vector space model for patent retrieval. Inform. Process. Manage., 47: 309-322. DOI: 10.1016/j.ipm.2010.06.001

Cilibrasi, R.L. and P.M.B. Vitanyi, 2007. The google similarity distance. IEEE Trans. Knowl. Data Eng., 19: 370-383. DOI: 10.1109/TKDE.2007.48

Coppola, P., V.D. Mea, L.D. Gaspero, D. Menegon and D. Mischis *et al*., 2010. The context-aware browser. IEEE Intell. Syst., 25: 38-47. DOI: 10.1109/MIS.2010.26

Finkelstein, L., E. Gabrilovich, Y. Matias, E. Rivlin and Z. Solan *et al*., 2004. Placing search in context: The concept revisited. Proceedings of the World Wide Web Conference, (WWWC' 04), Hong Kong, China.

Hearst, M.A. and J.O. Pedersen, 1996. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. Proceedings of the 19th Annual International ACM Special Interest Group on Information Retrieval, ACM Press, USA, Zurich, Switzerland, pp: 76-84. DOI: 10.1145/243199.243216

Isa, D., L.H. Lee, V. Kallimani and R. RajKumar, 2008. Text document preprocessing with the bayes formula for classification using the support vector machine. IEEE Trans. Knowl. Data Eng., 20: 1264-1272. DOI: 10.1109/TKDE.2008.76

Jin, H. and H. Chen, 2008. SemreX: Efficient search in a semantic overlay for literature retrieval. Future Generat. Comput. Syst., 24: 475-488. DOI: 10.1016/j.future.2007.07.008

Kaki, M., 2005. Findex: Search results categories help users when document ranking fails. Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems, Apr. 2-7, ACM Press, USA., pp: 131-140. DOI: 10.1145/1054972.1054991

Kraft, R., C.C Chang, F. Maghoul and R. Kumar, 2006. Searching with context. Proceedings of the 15th International Conference on World Wide Web, May 22-26, ACM Press, Edinburgh, Scotland, UK., pp: 477-486. DOI: 10.1145/1135777.1135847.

Lamberti, F., A. Sanna and C. Demartini, 2009. A relation-based page rank algorithm for semantic web search engines. IEEE Trans. Knowl. Data Eng., 21: 123-136. DOI: 10.1109/TKDE.2008.113

Lee, U., Z. Liu and J. Cho, 2005. Automatic identification of user goals in web search. Proceedings of the 14th International Conference on World Wide Web, (WWW' 05), ACM Press, USA., pp: 391-400. DOI: 10.1145/1060745.1060804.

Li, Y., Y. Wang and X. Huang, 2007. A relation-based search engine in semantic web. IEEE Trans. Knowl. Data Eng., 19: 273-282. DOI: 10.1109/TKDE.2007.18

Osinski, S. and D. Weiss, 2004. Conceptual clustering using lingo algorithm: Evaluation on open directory project data. Proceedings of the International Conference IIS: Intelligent Information Processing and Web Mining, May 17-20, Springer Berlin Heidelberg, Zakopane, Poland, pp: 359-368. DOI: 10.1007/978-3-540-39985-8_38

Ratprasartporn, N., J. Po, A. Cakmak, S. Bani-Ahmad and G. Ozsoyoglu, 2009. Context-based literature digital collection search. VLDB J., 18: 277-301. DOI: 10.1007/s00778-008-0099-9

Saracoglu, R., K. Tutuncu and N. Allahverdi, 2007. A fuzzy clustering approach for finding similar documents using a novel similarity measure. Expert Syst. Applic., 33: 600-605. DOI: 10.1016/j.eswa.2006.06.002

Thangaraj, M. and V. Gayathri, 2011a. An effective technique for context-based digital collection search. Int. J. Mach. Learn. Comput., 3: 372-375. DOI: 10.7763/IJMLC.2013.V3.341

Thangaraj, M. and V. Gayathri, 2011b. A new context oriented synonym based searching technique for digital collection. Int. J. Mach. Learn. Comput., 1: 100-103. DOI: 10.7763/IJMLC.2011.V1.15

You, G.W. and S.W. Hwang, 2008. Search structures and algorithms for personalized ranking. Inform. Sci., 178: 3925-3942. DOI: 10.1016/j.ins.2008.06.009

Zamir, O. and O. Etzioni, 1999. Grouper: A dynamic clustering interface to web search results. Proceedings of the 8th International Conference on World Wide Web, (WWW' 99), Elsevier North-Holland, Inc., Toronto, Canada, pp: 1361-1374. DOI: 10.1016/S1389-1286(99)00054-7