A TAG⁺ - Tree for Retrieval from Multiple Domains of a Publication System

M. Thangaraj¹, V. Gayathri²

¹Associate Professor, Department of Computer Science, Madurai Kamaraj University, Madurai, TN, India.
²Research Scholar, Department of Computer Science, Madurai Kamaraj University, Madurai, TN, India.

ABSTRACT
It is very difficult for the researchers to collect the literature for their research. The digital libraries of the publication systems are very useful for them. Publication systems have set of literatures of various domains. The effectiveness of the Publication system is measured from various aspects; especially indexing and retrieval. This paper shows how effectively the TAG-tree is used for a Publication system. The results show that it performs well when compared to other existing techniques.

Keywords: Context-based search, TAG-tree, TAG⁺-tree, Publication search, Information Retrieval.

1. INTRODUCTION
In the traditional library, people get the help of Information Specialist, who assist others to find information. In this modern technology, the traditional libraries are promoted to digital libraries, which is readily available in our systems through online. The role of the Information Specialist is now played by Information Retrieval Systems (IRS). Thus making the IRS more efficient and potential enough to satisfy the user information need is the major research issue in this field.

In general, retrieving the relevant documents from a digital collection is based on matching the character strings rather than retrieving meanings. The user has to select such a powerful string (query) that represent the information need accurately, and return the relevant documents. It is undoubtedly known fact, that all information needs cannot be expressed by a unique character strings. A single word may have different meanings. This may lead to topic diffusion. This indirectly increases the result set with unwanted results.

The only solution is making the IRS to treat the query as Context instead of a set of character strings [6]. Contexts are the keywords along with its synonyms. When the IRS treats the query as Context, then it will find the relation between the keywords of the query. Thus making the IRS effective, it has to perform Context-based Search.

For an effective retrieval, the concentration needs to be given to the indexing system also. When the collection increases tremendously, we can retrieve the relevant documents, only if it is indexed in such a way that it is easily reached. Earlier the linear approach is used for indexing, which increases the disk access time. Thus the hierarchical approach is useful, which consumes a less amount of disk access and also limits the number of comparisons. There are more researches done and it shows that the tree based algorithms are good. To reach the object within one comparison, Hash based techniques are useful. The hash value of the object is identified based on which the respective bucket of the hash table is found directly. Thus it needs only O(1) time which also reduces the computational cost.

The rest of the paper is organized as follows. We discuss related work in Section 2. In Section 3 we describe the general architecture of the system, whereas experimental results are presented in Section 4 and conclude in Section 5 with the future directions.

2. RELATED WORK
There are more researches were done in this field of retrieval. When the growth of the information volume increases, there occurs more issues such as the change in the behaviour of the data and the relationship between the documents, the way of handling it and so on. Thus it stills the thirst area of research.

Various data structures were found in the literature used in various steps of retrieval. In [3], kd-tree is used for query matching, which is a binary tree with storage requirements proportional to the number of records to be searched. A record is a term or document represented by a vector of length k. The tree has to traverse until the leaf is reached to get the closet documents. It is time consuming, since it takes more number of comparisons. The distance from the root is maintained for all nodes. For very large document collections, this method of finding nearest neighbours to the query can be computationally expensive.

To overcome the issue discussed above, in [4], J⁺-tree is used, which is a height balanced tree. The difference between B⁺-tree and J⁺-tree is that, it increases the size of the buckets of the parent to hold more elements with no change in the height of the tree. When the bucket size increases more, it has to compare more number of elements to reach the element in the child. Very frequently indexing is done by considering the characters in the keywords. For example, in [9], Suffix Tree is constructed with the terms based on the character sequence. This is feasible only if the index terms are very small. There are more indexing structures to handle large volumes of data are available in the literatures. For example, R-tree
[2], R	extsuperscript{2}-tree [7] and SKD-tree [5] were used for representing spatial data on multidimensional spaces.

In order to handle the large volumes of textual documents on a multidimensional space, D-tree is proposed in [1]. It is same as R-tree, in which R-tree models the spatial objects with overlapping subspaces, whereas the D-tree indexes textual documents with overlapping keywords. It can facilitate to have a document cube with more dimensions. An internal node of a D-tree contains the unique identifiers of the documents that are in the node and the links to the child nodes. Leaf node contains only the unique identifiers of the documents. Each dimension of the document cube may refer to different topics. It will group all the documents that have the same set of keywords. The issue here is, if it is queried with a set of keywords, which is used in different topics, the resultant set may contain mostly the irrelevant document. Also finding the related documents is very poor, since it is affected by topic diffusion.

To avoid these issues, our previous work TAG [8], in which a context based retrieval technique was proposed. It has various functional components: (a) TAG Extractor, which performs the pre-processing and constructs the Contexts. (b) TAG Indexer: Publications are indexed based on the Contexts in the TAG-tree. (c) TAG Suggester: Helps user to form the informative query. (d) TAG Retriever: Retrieves the publications based on each relevant Context with the help of Thesaurus. (e) TAG MRanker: Resultant publications of various Contexts are merged. Finally based on the different levels of scoring, publications are ranked and returned. TAG-tree uses the data structures B	extsuperscript{T}-tree and list. The work was tested with the publications of a single domain. The issue of topic diffusion and size of the resultant set was controlled and minimized compared to the earlier work.

3. TAG FOR A PUBLICATION SYSTEM

This section shows how efficiently the TAG can be used for a publication system. TAG	extsuperscript{2} is an updated architecture, which uses the various modules of TAG technique, except the TAG-tree. TAG	extsuperscript{2} performs the following steps.

1. Acquisition: The information to be indexed is acquired from the publications. Full-texts of the publications are not available at all the times. Thus the general information freely available is title, abstract and keywords. This public information is acquired along with its domain details.
2. Cleaning: The title, abstract, and keywords from all the publications are cleaned through various steps. At first these information are tokenized. Then stop words from these tokens are removed.
3. TAGging: Applying TAG technique for indexing with the advanced indexing structure.
4. Querying: The query received from the user has the domain information along with the query pattern. When a query is posed, the corresponding domain is identified and it’s relevant TAG-tree is searched for the given query pattern.
5. Result Set Generation: As in the TAG technique, the results from TAG-tree and Parser are merged and ranked. And finally the most relevant, limited results are returned to the user.

The overview of the TAG	extsuperscript{2} is shown in the Figure 1.

At first the TAG Extractor is used: the domain name, the title, the abstract and the keywords of the publications are extracted from the publication repository. This information is tokenized, and frequently used stop words, (for ex., a, an, the, is & etc) are identified and removed. The remaining tokens are indexed in the TAG	extsuperscript{2} structure. For each domain, the subject terms, and its synonyms to be indexed, are identified by domain experts. Based on these terms, the cleaned tokens selected by the TAG Extractor from the publication repository are indexed in the TAG	extsuperscript{2} structure. The user has to specify, the domain of the query. When the user starts typing the query the TAG Suggestor helps the user by suggesting with queries based on the specified domain. These suggestions were made with the help of the previous history and by the parser (as such in TAG technique).

![Figure 1](image-url) **Figure 1** The TAG	extsuperscript{2} architecture

3.1 TAG	extsuperscript{2}: A modified indexing structure

Figure 2 shows the architecture of the new indexing technique TAG	extsuperscript{2}. TAG	extsuperscript{2} is a combination of hash table and TAG trees. Each bucket of the hash table is connected to a TAG tree, which is filled with the Contexts (keywords and
synonyms) of the particular domain. TAG tree is a combination of B'-tree and list. The contexts are filled in the B'-tree and the synonyms are filled in the list, which is connected at the leaf nodes of the B'-tree. Thus the size of the hash table is the number of domains in the publication system.

As in TAG technique, the Contexts of each domain are in the form of the pattern such as <prefix> <context> <suffix>. From each domain, the key terms are extracted with the help of Subject Experts manually. Only in the leaf nodes of TAG-tree contains the publications of its pattern along with a pointer to the list which contains its synonyms. When a query is posted, TAG Retriever will locate the relevant contexts from the TAG' indexing structure to get the relevant publications. Also it retrieves the publications which are indentified from the parser database. Finally the publications from the stored Usage History, which are relevant to the given query, are all combined together. Based on the scores of these resultant publications, they are ranked. The top most limited resultant publications are returned, as a result set.

To retrieve the result from the TAG’-tree, the following algorithm is used.

Algorithm: MRetrieval

Data Structure: See Figure 2.
Input: 
\( d \) – domain id
\( q \) – Query \( (q_p, q_c, q_s) \)
Output: \( RPset \) – Resultant Publications.
Let \( RPset \) – Resultant Publications’ id with its scores
\( q_p \) – Prefix tuple of the given query
\( q_c \) – Context tuple of the given query
\( q_s \) – Suffix tuple of the given query
mRetrieval(domain_id \( d \), Query \( q \))

\[
hvd = \text{hash}(d); //\text{computes the hash value for the queried domain id } d
\]
ConSynRet(q, hvd, dtp); //Retrieval algorithm in TAG [8]
return \( (RPset) \);

4. PERFORMANCE STUDY

In this section we describe our experiments to assert the usefulness of TAG’ indexing technique in the task of indexing/retrieval of documents in multi-domain publication system. Many experiments were conducted on this system, but presented only most important results. This system is compared to the earlier model D-tree. These 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. The test data was created with a multi-domain publication system. For testing, 20000 full text articles from 20 different domains were collected manually from various publication systems like IEEE, Medline and so on. This collection is queried with some phrases that are common for more than one domain to evaluate the performance of the proposed approach.

4.1 Indexing Time

After the pre-processing of the documents, it is indexed with the selected structure. This study focuses on indexing time for the documents by varying the size of the data, shown in figure 3.
Due to the linear handling approach and the multidimensional property of D-tree, it involves in more number of comparisons for indexing. D-tree uses the approximate matching, which will increase the number of documents as well as computational time for a particular topic, where as our TAG$^+$ technique uses the pattern based approach that consumes less amount of time for indexing. The performances of these two structures were measured and TAG$^+$ technique performs better than D-tree.

4.2 Retrieval Time

Figure 4 shows the retrieval time for different context. In this study, the maximum size of the collection is fixed to 20000, and the systems are queried for documents in various domains.

The search boundary in a TAG$^+$ method is very minimum compared to the D-tree method. This leads to access of most needed documents from a minimal data set. The retrieval performance of a new method improves much compared to D-tree.

4.3 Relevancy

The main issue in searching is that, retrieved results should satisfy the user’s information need. The size of the collection increases in various domains, it is too difficult to retrieve the exact documents by means of a single query. Figure. 5 & 6 discusses the relevancy measure of these techniques. To test the topic diffusion problem, the queries were chosen in such a way that, the terms involved in the query were common to more than one domain. For this study the entire collection is queried and the results were evaluated manually by the domain experts.
Since TAG\(^+\) uses a hash table to locate the queried domain, the topic diffusion is reduced considerably. In D-tree there is no specific approach to locate the queried domain. It considers the query as a set of keywords. Thus the meaning of the query is not taken while searching the collection. Hence the topic diffusion is high when compared to our TAG\(^+\) approach. From this study it is very clear that our TAG\(^+\) approach outperforms than the existing techniques.

**5. CONCLUSION AND FUTURE DIRECTIONS**

The proposed TAG\(^+\) is an effective technique for indexing and retrieval for a vast publication system. The technique was implemented and performance was analyzed. This technique enormously reduces the size of the resultant set and overcomes the problem of topic diffusion.

This work can be extended to facilitate the retrieval of back volumes of a particular topic. Also personalization can be done to predict the user behavior, which in turn helps to identify the Context.

6. ACKNOWLEDGEMENT

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

**References**


**AUTHORS**

M Thangaraj received his post-graduate degree in Computer Science from Alagappa University, Karaikudi, M.Tech. degree in Computer Science from Pondicherry University and Ph.D. degree in Computer Science from Madurai Kamaraj University, Madurai, TN, South India in 2006. He is now the ASSOCIATE PROFESSOR of Computer Science Department at Madurai Kamaraj University. He is an active researcher in Web mining, Semantic Web, Information Retrieval and Big-Data Analytics and has published more than 60 papers in Journals and Conference Proceedings.
He is a member in the Society of Digital Information and Wireless Communications (SDIWC) and is a senior member of IACSIT. He has served as program chair and program committee member of many international conferences held in India for about one decade.

V Gayathri received her post-graduate degree in Computer Science in 2008 and M.Phil. degree in Computer Science in 2010 from Madurai Kamaraj University, Madurai, TN, India. She is currently a Ph.D. SCHOLAR and PROJECT FELLOW in ICBR (UGC-MRP) in Computer Science Department, Madurai Kamaraj University, Madurai, TN, India. Her research interests include Context-based Search, Information Retrieval and Text Mining. She is a member in the Society of Digital Information and Wireless Communications (SDIWC).