An Effective Technique for Context – Based Digital Collection Search

M. Thangaraj, *Member*, *IACSIT* and V. Gayathri

***Abstract*—At present the major issues in searching digital collections are (a) topic diffusion: results returned by a keyword based search, fall into multiple topic areas, which are not interested to users; (b) there is no effective scoring mechanism; so the users are forced to scan a large result set, which leads them to miss the important ones. In order to help the users, we propose a technique, NCBS (New Context Based Search). This approach uses the data structures such as B^+-Tree and an inverted list. The extensive study shows that the proposed approach effectively controls the diversity of output topics, reduces the size of the search results, and has better performance than the existing method.**

***Index Terms*— Context–Based Search (CBS), digital library, b^+-tree, inverted list, digital collection.**

I. Introduction

The web is rapidly growing and becoming a huge repository of information, with several billion pages and more than 300 million of users globally [1]. This information volume causes many problems that relate to difficulty of finding, organizing, accessing, and maintaining the required information by users.

There are two major issues encountered, while searching in a digital collection. First issue is topic diffusion. Results returned by a search query often fall into multiple topic areas, which diffuses the topic, not all of which are of interest to users. Second issue is relevance. Relevance depends on how well retrieved documents satisfy the user needs. In order to avoid the problems mentioned, a specific searching mechanisms to be established.

The digital library is an electronic library where the information is acquired, stored and retrieved in digital form. Digital libraries provide instant access to all information, for all sectors of society, from anywhere in the world. For a given query, it may return huge number of results. It is obvious that very few results are relevant to the user needs out of the huge set of results. Thus, we need an effective searching mechanism in digital collection, to produce the best result.

The remainder of the paper is organized as follows: Section II is devoted to the issues relevant to searching. In Section III, we describe the architecture for NCBS. Section IV shows our performance evaluation result. Finally, in Section V we present conclusion.

Manuscript received February 15, 2013; revised June 18, 2013.

The authors are with Department of Computer Science, Madurai Kamaraj University, Madurai, TN, India (e-mail: thangarajmku@yahoo.com, gayathrivengatmku@yahoo.com).

II. Related Work

There are many digital collection search systems, such as Google Scholar, IEEE Xplore, and etc., available online. These systems produce results based on the relevancy to the query term and/or the importance of the papers. They do not use contexts to organize search results.

A number of categorization techniques have been proposed to make search results more understandable. Two widely-used categorization techniques are clustering and classification. Clustering creates categories (or contexts) by grouping similar documents together while classification assigns documents to a set of predefined categories [2].

Clustering can also be further classified as flat clustering and hierarchical clustering [3]. Even though more techniques are available, still we have some problems in the retrieval. In an algorithm named Semantic Forests [4] uses an electronic dictionary to make a tree for each word in a text document. The root of the tree is the word from the document, the first branches are the words in the definition of the root word, and the next branches are the words in the definitions of the words in the first branches, and so on.

Finally, the trees are merged into a scored list of words. The premise is that words in common between trees will be reinforced and represent "topics" present in the document. The main drawback here is, it passes twice, to produce the desired result. In which, the first pass could be made to get all documents which match the desired topics and a second pass made to eliminate the ones which discuss unwanted aspects of the topic. There is no effective technique/method to extract the desired information from a dictionary or thesaurus; and finally it doesn't provide the relevant words.

In CoFS [5]-[7], users in this system use tags to describe files or resources of special interest. A set of tags assigned to a file by a user is called a tag-based context. For each user, his interesting files are organized into the appropriate contexts. A directed acyclic graph of tags is created for each user to help him navigate from one context to another to retrieve his files. Instead a common directed acyclic graph with an inverted index is enough. Thus, it leads overhead of more number of directed acyclic graphs which is poor in updating.

Similarly in Context Based Search (CBS) [8], 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

DOI: 10.7763/IJMLC.2013.V3.341

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 retrieval time.

III. Proposed Work

In this work, we propose a technique named NCBS, which effectively retrieves the relevant document from the collection. It has two major segments such as pre–processing the digital collection and the query evaluation. The following steps are used in the pre-processing phase:

1) *Pattern Extraction: From the data set, patterns are extracted to organize the documents in the digital collection*.
2) *Constructing Searching Structure: Documents in the digital collection, which are matched to the extracted patterns, are mapped to the searching structure*.

Note that pattern extraction and construction of searching structure are pre-executed and not dependent on queries. The following steps are used in the query evaluation:

1) *Processing the query: checks whether the given query is in the structure of the pattern, and extracts the pattern from the given input text.*
2) *Identification of Context: The context relevant to the query pattern is extracted using the searching structure*.
3) *Synonyms Selection: The relevant synonyms for the identified context are also extracted*.

The architecture of this model is given in Fig. 1. The data set is nothing but the information in the digital collection pertaining to multiple topics. Using the pre-processing phase, the documents in the digital collection are classified based on the context extracted (using pattern extraction technique). When a query is specified, the relevant pattern is extracted from the collection of patterns. Pattern matching is performed in the searching structure with the extracted query pattern. The relevant context is identified, then searching in the list to retrieve the synonyms and return the result along with its synonyms.

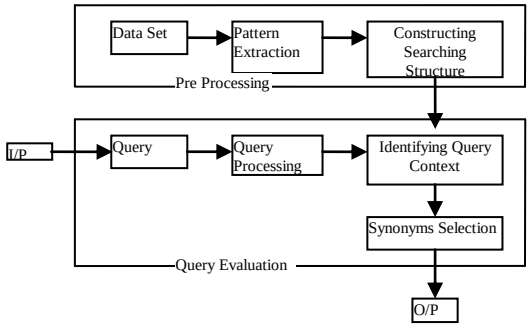

Fig. 1. NCBS architecture.

A. Pattern Extraction

This section presents a pattern extraction technique that constructs patterns from a context's data set. The constructed patterns are then used to assign documents to contexts: Significant terms (phrases), which are terms related to a context, are constructed from frequent terms (phrases) located in the documents.

Patterns are constructed from significant terms as follows. A pattern consists of three tuples: <Left> <Middle> <Right>, where each tuple is a set of words. Significant words (i.e., words in the significant terms) appearing in the data set are assigned to <Middle> tuple. Words surrounding the significant words are assigned to <Left> and <Right> tuples. The number of words for <Left> and <Right> tuples are determined by a window size. The <Middle> tuple represents the <context> tuple, and the <Left> <Right> tuples denotes the <Prefix> and <Suffix> tuples.

B. Constructing Searching Structure

The earlier section has described how the digital collections are classified into contexts using pattern extraction based technique. In this section mapping of classified contexts into a searching structure are discussed.

The searching structure is a composition of a B^+-tree and inverted list. The B^+-tree is organized based on the context 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 an inverted list. The inverted list has the set of synonyms for the given context.

Each pattern is considered as a separate context. Each context is mapped into the B^+-tree (Context Tree) as an individual bucket element. Patterns that are extracted by virtually walking from one to another are considered as the descendant patterns. These descendants are mapped into the tree as a child to the context of the pattern, from which these descendants are extracted.

In the internal nodes of the Context tree, it has only the Context, that is, pattern with the three tuples <prefix> <context> <suffix>. But, the leaf node has Context and a pointer to a list that holds synonyms. The synonyms of the context are mapped into an inverted list (synonyms list).

And also each element in the leaf node has a pointer to the document in the data set, which actually represents the relevant Context. Using this pointer, the document related to the specified context can be retrieved efficiently. The searching architecture with B+-Tree and inverted list are shown in Fig. 2.

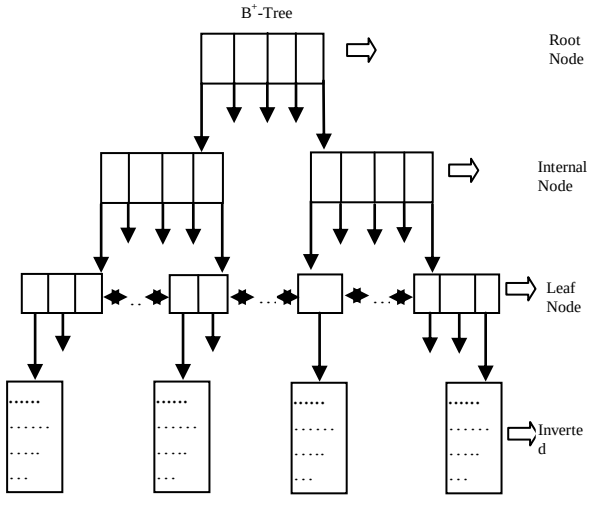

Fig. 2. Overview of searching structure.

Prefix	Context	Suffix	Pointer to the Document	Pointer to Synonyms List

Fig. 3. Context tree leaf record structure.

The structure of a record (bucket element) in the leaf node of the Context Tree is shown in Fig. 3. The leaf node in the Context tree has structure containing pattern details such as prefix, context, suffix, pointer to the document relevant to the context, and pointer to the synonyms list. The remaining node of the Context tree contains only the pattern details such as prefix, context, suffix, and a pointer to its children.

C. Identification of Context

The searching structure is searched against the query pattern to identify the relevant context. Initially, NCBS search for the left tuple: 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.

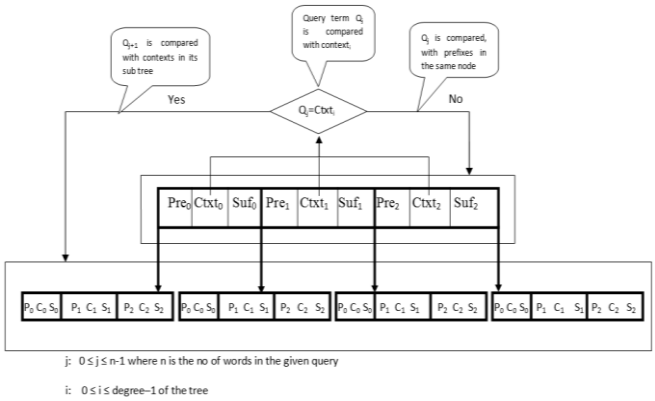

Fig. 4. Searching against the contexts of the contexts.

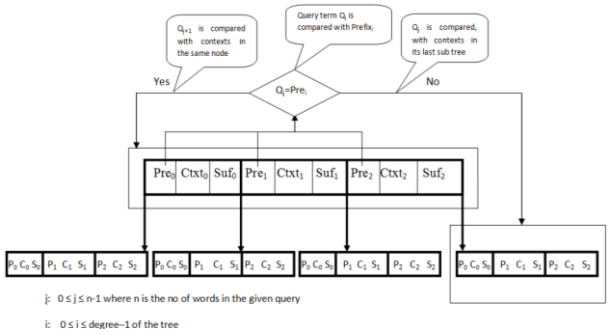

Fig. 5. Searching against the prefixes of the contexts.

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. 4). 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. 5). 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.

D. Synonyms Selection

When a context is identified in the leaf node, then it has to return the result, along with its synonyms. Each leaf node record has a link to an inverted list, which contains the synonyms of that context. The searching mechanism follows the link to retrieve the synonyms. The list holding synonyms are traversed to fetch all the synonyms relevant to the context. As the leaf node has a pointer to the list, it is very effective for immediate retrieval of synonyms.

IV. Performance Analysis

This NCBS approach has been implemented using C++. The set of experiment explores the effects of data and query parameters on performance. All experiments reported in this section are done on Pentium 4 2.x GHz with 256MB RAM and 80 GB of secondary storage, running Windows 2000. In order to test the model and to find the performance, about 5000 documents have been created and various queries have been implemented.

From the experiments performed, it's come to know that the traditional method has the lowest storage cost. In particular, the CBS approach has less storage cost when compared to the NCBS approach. However, the storage costs may not be crucial, since large capacity storage devices are available for lowest cost. Therefore, it may be preferable to design models that provide good performance, even if they have large storage requirements.

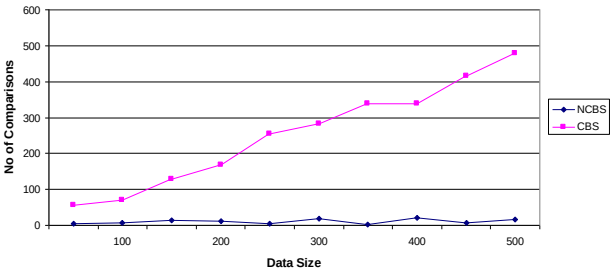

Fig. 6. Comparisons during searching.

The Fig. 6 shows the amount of comparisons needed during searching to identify the context. The NCBS needs only less number of comparisons, because it searches only to the relevant items using searching structure. But, the CBS approach has to scan through the entire collection in a linear order.

The Fig. 7 illustrates the comparison of the context retrieval performance of the two approaches. The time overheads are measured for retrieving the context to the given query. The time to retrieve the context in CBS approach is

drastically increased because, it searches linearly. The graph in Fig. 8 compares the synonyms retrieval performance of the two approaches. The synonyms are retrieved for the given query and the time overheads are measured.

The main goal of these experiments is to study the retrieval performance of the NCBS approach. The result of the study shows that NCBS approach performance is better than the CBS approach.

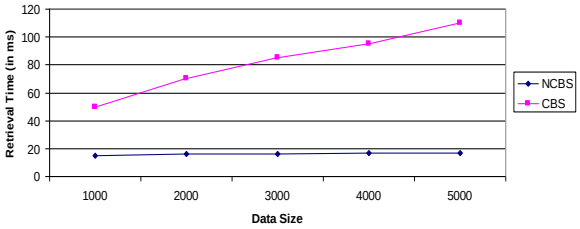

Fig. 7. Context retrieval time.

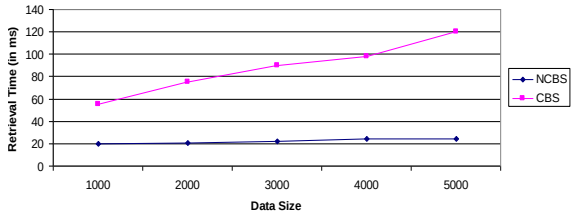

Fig. 8. Synonyms retrieval time.

V. Conclusion and Future Work

The existing algorithms and methods are having problems of retrieving the document with topic diffusion and large result set that are not match with the user's interest. The NCBS approach effectively controls the diversity of output topics and reduces the size of the search results.

The results of this study indicate the importance of the searching structure used in NCBS approach which improves the performance of searching in the digital collection. The same work may be further extended by applying some training algorithm for effective retrieval of data.

Acknowledgment

The authors wish to express their sincere appreciation to Prof. G. Arumugam, M. K. University, for his invaluable comments and feedback, and also thank the anonymous reviewers of ICMLC'11 for their significant comments.

References

[1] Survey on information explosion by netcraft secure survey upto. August 2012. [Online]. Available: http://news.netcraft.com/archieves/web_server_survey.html

[2] M. Kaki, "Findex: search results categories help users when document ranking fails," in *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems*, New York, USA, 2005, pp. 131-140.

[3] P. Ferragina and A. Gulli, "A personalized search engine based on Web-snippet hierarchical clustering," in *Proc. ACM WWW'05 Special Interest Tracks and posters of the 14th International Conf. on World Wide Web*, New York, USA, 2005, pp. 801-810.

[4] G. D. Henderson, P. Schone, and T. H. Crystal, "Text Retrieval via Semantic Forests," in *Proc. Text Retrieval Conf. (TREC)-7*, Gaithersburg, Maryland, 1998, pp. 583-594.

[5] H. B. Ngo, F. S. Chaussumier, and C. Bac, "A Context-based System for personal file retrieval," *ACM*, 2009.

[6] W. H. Cheng and D. Gotz, "Context–based page unit recommendation for web–based sensemaking tasks," in *Proc. The 14th international Conf. on Intelligent User Interfaces, IUI '09, ACM*, New York, NY, USA, 2009, pp. 107-116.

[7] C. A. N. Soules and G. R. Ganger, "Connections: using context to enhance file search," in *Proc. SIGOPS Oper. Syst. Rev*., 2005, vol. 39, no. 5, pp. 119 -132.

[8] N. Ratprasartporn, J. Po, A. Cakmak, S. B. Ahmad, and G. Ozsoyoglu, "Context-based literature digital collection search," *The VLDB Journal*, vol. 18, pp. 277–301, 2009.

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 an associate professor of Computer Science Department at M. K. University. He is an active researcher in Web mining, Semantic Web and Information Retrieval and he has published more than 60 papers in Journals and Conference Proceedings. He is a senior member of IACSIT. He has served as a 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 in Dept. of Computer Science, M. K. University, Madurai, TN, India. Her research interests include context-based search, information retrieval and semantic web.