Advances in Wireless and Mobile Communications.
ISSN 0973-6972 Volume 10, Number 2 (2017), pp. 333-340

http://www.ripublication.com

RECBD APPROACH - RETRIEVAL OF EFFICIENT AND CONTENT BASED RELATIVE DATA

Professor Dr. Sudan Jha

School of Computer Engineering,
KIIT University, Bhubaneswar,
Odiha – 751024, India.

Abstract

It is well understood that the data mining confides a huge number of discovery algorithms and these algorithms help in generating a large number of patterns and rules. In most of the cases, the image processing and pattern recognition have been successful, but in some of the cases due to the exceeding size of the databases, it gets fails in retrieving the right information or in particular accomplishing the exact knowledge discovery process. Despite of huge number of discovery algorithms, and associated with same number of generated rules, identifying and then making analytical treatment, the task of knowledge discovery process becomes difficult. For this to become useful, an approach must be initiated which can manage rule selection flexibly or using approach based flexible tools. Among the several approaches of association rule mining, one of the approach called "similar" rules worked out well in some considerable number of rules. But again the restriction is that this "similar" approach didn't do well when the number was large, thus creating huge number of clusters. More flexible approach called "RECBD APPROACH" is presented in this paper. "RECBD" approach allows the identification of rules on the priority of their importance to the user DMQs (Data Mining Queries), or templates or other database queries. This approach is a reverting approach of grouping approach; but it has been used to specify supervised learning and unsupervised learning classes of rules (for both association and episodic rules). The importance of DMQs has been illustrated through inductive database concept, where a user is allowed to access querying the data as well as the query patterns, rules, and models extracted from these data. As an abstract example, let us provide a user with a list of association rules. These rules are ranked by their confidence and support. But this might not be a good way of organizing the set of rules as this method would overwhelm the user and not all rules with high confidence and

support are necessarily interesting for a variety of reasons. "RECBD" approach organizes the set of rules in grouped manner such that the retrieval of data becomes content based effectively and in a knowledgeable way.

Keywords: Algorithms, Data Mining Queries (DMQs), Data mining, Selection, Association rule, Association and Episodic rules, Patterns, Rules, Models, Signature,

I. INTRODUCTION

Inverted files have been used extensively in text retrieval (Moffat & Zobel 1996, Zobel, Moffat & Ramamohanarao 1998). Their main application is for supporting partial match retrieval, which is basically subset queries. This section describes the use of inverted files for supporting set retrieval. Given a collection of target sets, D, these files are - an inverted file which contains all distinct values in D. This inverted file consists of a directory, for each value in this directory, an inverted list that stores a list of references to all occurrences of this value in the database is maintained, i.e. list of references to target sets containing the value. Consider the market basket data containing four transactions and five items. Using this database an inverted file index can be created and is shown in Figure 1.

An inverted list of an item is presented as $<n; tid_1 \ldots\ldots tid_n>$, where n is the number of transactions in which an item appears, followed by transaction identifiers (tids). As shown in the figure, the inverted list of the item book is <2;1,4> because it appears in two transactions, namely, transactions 1 and 4. If D contains a large number of items, the search values in the directory are usually stored in the B-tree. Helmer and Moerkotte (1999) have adapted inverted files for set retrieval and modified the inverted list by also storing the cardinality of the target set with each target set reference, so that set queries can be answered more efficiently by using the cardinalities as a quick pre-test. As an example, the inverted list of the item book becomes <2;(1,3), (4,2)> effective, it is not a perfect solution.

In the produced inverted file, the subsequent subset queries may be processed as:

1. An appropriate list is fetched for each item in the query
2. After then, all those lists are intersected
3. Due to this intersection, a list of references will be contained that contains the query set

However, the equality queries are also processed in the same way as above, however in subset queries we can improve the processing by first checking those whose cardinality is not equal to the query and then eliminating them. When evaluating superset queries, all lists associated with the values in the query set are retrieved. Then the number of occurrences of each reference appearing in the retrieved lists is counted. A reference

whose number of occurrences is not equal to the cardinality of its set is eliminated. The existence of such a reference means that the reference appears in the lists associated with the values that are not in the query set, so that its set cannot be a subset of the query set Helmer and Moerkotte (1999) compare the performance of three signature based indexes against that of the inverted file index in processing equality, subset and superset queries. They conclude that the inverted file index structure dominated other index structures for subset and superset queries in terms of query processing time. Kouris et al. (2004) have used the inverted file index to improve the performance of an Apriori-based algorithm in discovering association rules. The index is accessed during support counting so that instead of reading the original database, the mining algorithm scans the inverted file index stored in memory. Tuzhilin and Liu (2002) use inverted file indexing scheme for querying multiple sets of discovered association rules. A comparison between inverted files and signature files is also studied by Zobel et al. (1998) and Carterette and Can (2005).

II. REVIEW OF SET AND SEQUENCE RETRIEVAL

The retrieval of data objects on set-valued attributes (for short set retrieval) is an important research topic with wide areas of applications. A significant amount of today's stored data consists of records with set-valued attributes (i.e. attributes that are sets of items). Set-valued attributes are extensively used in object oriented databases to represent an object's multivalued attribute (Ishikawa, Kitagawa & Ohbo 1993), in multimedia databases representing objects inside an image (Rabitti & Zezula 1990), and in data mining applications representing basket market data (Morzy & Zakrzewicz 1998). Although advanced database systems, such as nested relational or object-oriented database systems, provide the means to store set-valued attributes in the databases, they do not provide language primitives or indexes to process and query such attributes. Furthermore, some of the existing index structures proposed in the database literature, for example, (B+ trees (Comer 1979) and R trees (Guttman 1984), etc.), are not designed to fully support set value manipulation in general. Therefore, new types of index structure have been proposed in the literature to support queries on set-valued attributes, namely, signature files and inverted files.

One of application domains that would benefit from the possibility of performing efficient querying on sets is data mining. Several data mining techniques rely on excessive set processing, especially in the case of mining association rules using the Apriori family of algorithms. Shifting these computations from the data mining algorithms to the database engines could result in considerable time savings. Efficient set retrieval is also useful during pattern post-processing for the selection of discovered association rules according to user-defined criteria, or for the querying of the database against association rules to identify transactions that satisfy certain criteria. Recently the set retrieval using signature files has also been used as a basis for sequential patterns retrieval

IV. ARCHITECTURE AND EXECUTION

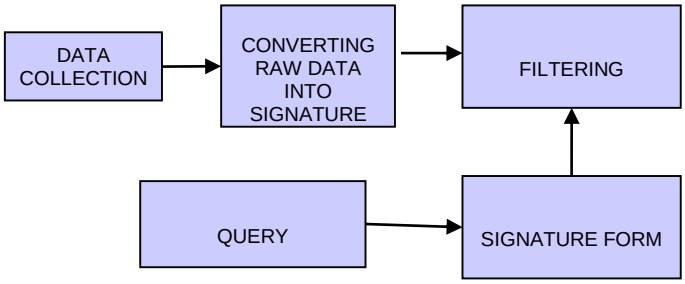

Figure 2: Data Flow Diagram

After this address of the problem of Super pattern and sub pattern queries are analyzed. In this filtering step, the false-drop process is done. The yielding output in the form of UML diagram is as below: -

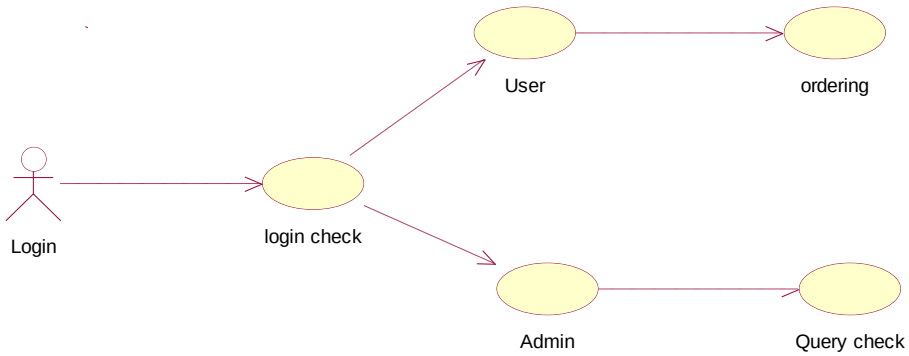

Figure 3: - The UML Diagram

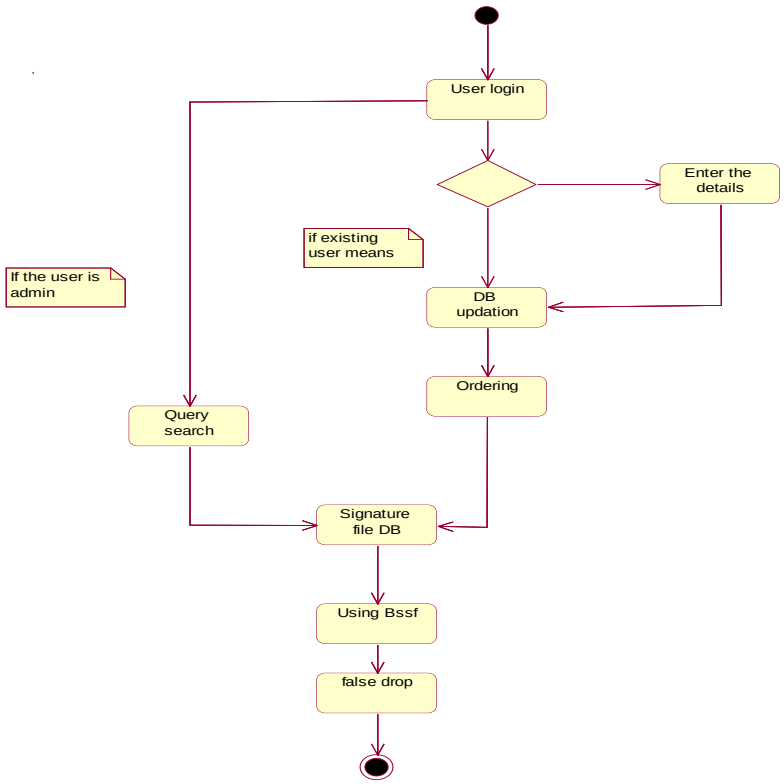

Figure 4: - The Activity Diagram

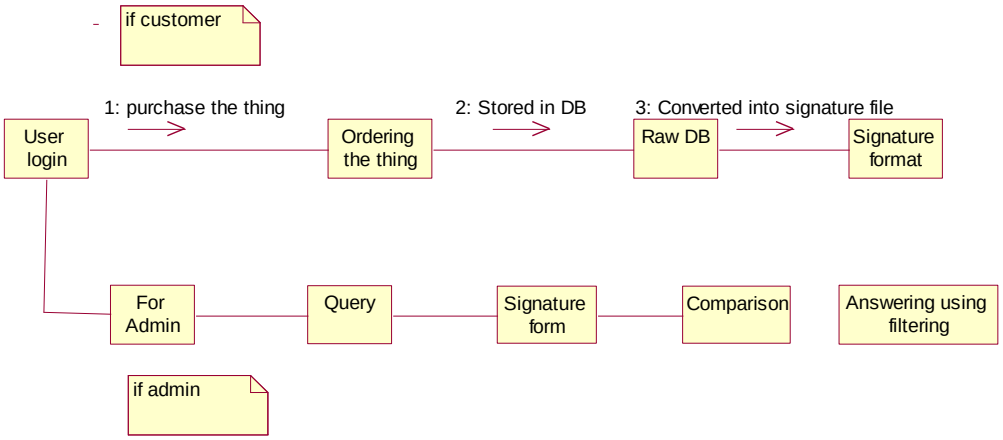

Figure 5: - The Collaboration Diagram

Figure 6: - Sequence Diagram

V. SET RETRIEVEL USING SIGNATURE FILES

The purpose of using signature files in set retrieval is to filter out the non-qualifying data objects. The basic idea is to represent the set-valued attribute of data objects into bit patterns, called signatures, and store them in a separate file which acts as a filter to eliminate the non-qualifying data objects when processing set queries. A signature failing to match the query signature guarantees that the corresponding object can be ignored. Therefore, unnecessary object access is prevented. Since direct set comparisons are very expensive, using signatures as filters can speed up query processing in set retrieval.

In order to filter out the non-qualifying objects, set retrieval is done. "Signature files in set retrieval" - the fundamental approach is to represent them into but patterns. Now these bit patterns are called signatures. These signatures are stored in the separate files. When processing set, queries starts, these separate files act as a filter and thus starts eliminating the non-qualifying data objects. A particular object is said to be unqualified, when a signature or the file fails to match the query. As a result, the filtering of those non-qualifying is done. Since direct set comparisons are very expensive, using signatures as filters can speed up query processing in set retrieval.

VI. METHODS FOR GENERATING SET SIGNATURES

Various methods are approached in generating the set signatures. In set retrieval with signature files, for each target set, a target signature is generated and then these target signature are stored in the signature file. A number of signature generation methods have been proposed by Faloutsos and Chistodoulakis (1987) in the context of text retrieval. These methods are Word Signatures (WS), Superimposed Coding (SC), Bit-Block Compresion (BC), and Run Length Encoding (RL). In **Word Signature** (WS), elements of target sets are mapped with a particular pattern of a given length. These

predetermined patterns are noted as "Word Signatures". The Word Signatures are sequenced to generate the target signatures. In **Superimposed Coding** (SC), again each element in a target set is mapped this time to a binary bit pattern. These binary bit patterns are called as an element signature. All element signatures have a given / prescribed (F) bit length with m bits which are set to '1'. Whilst m < F, F becomes the length of a signature, while m is called the weight of an element signature. Then, a target signature is obtained by bit-wise OR-ing (superimposed coding) element signatures of all the elements in the target set. Thirdly in **Bit-Block Compression** (BC), the signature extraction process for BC is similar to SC. The difference is that the original size (length) of the signature, designated as B, is large, and for each element of a target set only one bit is set to '1' (i.e., m = 1). As a result, the bit vector B of the set signature is sparse. Therefore, before storing the signature, B is divided into groups of consecutive bits of size b and compressed.

Run Length Encoding (RL). The RL method is similar to both SC and BC. It differs from BC only in the compression method. RL records the distances between the positions of bits with value '1'. Of these four methods, the most commonly used method in set retrieval is the superimposed coding (Helmer & Moerkotte 1999, Ishikawa et al. 1993, Tousidou, Bozanis & Manolopoulos 2002, Morzy & Zakrzewicz 1998). Therefore, for the rest of this chapter, unless stated otherwise, it will be assumed that the superimposed coding is used to generate set signatures. Figure 6.3 illustrates the generation of set signature using the superimposed coding when the value of F = 8 and m = 2.

VII. CONCLUSION

For the content based retrieval of various patterns, a target oriented signature based index is been used successfully. The paper thus signifies the creation of patterns that are temporal and then to adjudge their equivalency through the equivalency sets thus generating the signatures. These are belongings of the equivalent sets mentioned earlier. The work basically focused on the sequential organization of the RECBD approach via series of experiments, comparisons among the outcomes of those experiments and performance evaluation substantially. This is performed in both of the signature files, sub patterns and design patterns. A speculative and predictive performance is achieved which helps in improving the performance of these patterns. In the ongoing work, the visualizing techniques are being combined with the RECBD approached system so that the performance is monitored in the run time and further enhancements in the retrieval sets can be done.

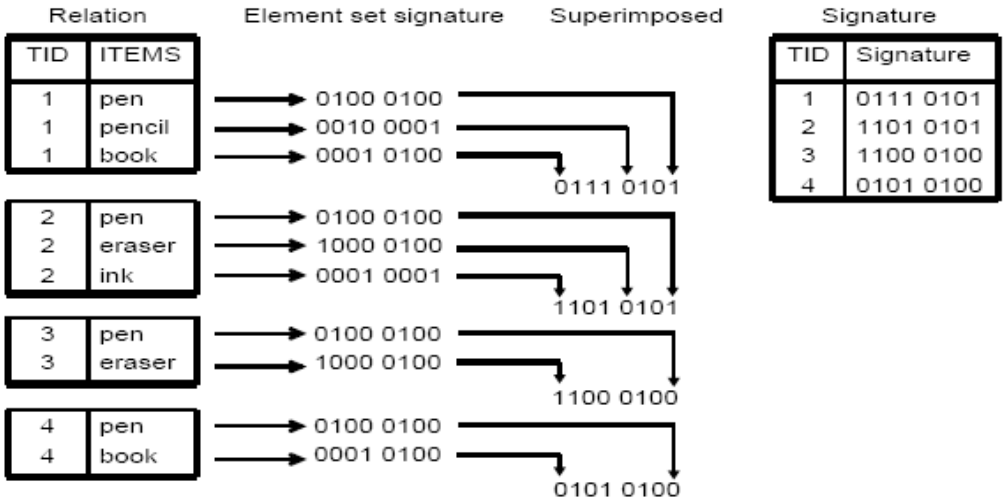

Figure 7 Generating signature using superimposed coding

REFERENCES

[1] Jeanquier S (2006) An Analysis of Port Knocking and Single Packet Authorization. M.Sc. Thesis (Royal Holloway, University of London).

[2] Krzywinski M (2005) Port Knocking From the Inside Out. *hakin9* **5**.

[3] Rash M (2005) Combining Port Knocking and Passive OS Fingerprinting with fwknop

[4] Krzywinski M (2003) Port Knocking: Network Authentication Across Closed Ports [txt]. *SysAdmin Magazine*

[5] Graham-Cumming J (2004) Practical Port Knocking. *Dr. Dobb's Journal* **366**

[6] Doyle M Implementing a Port Knocking System in C, *Department of Physics, University of Arkansas*

[7] Maddock B (2004) Port Knocking: An overview of Concepts, Issues and Implementations. *SANS Institute*

[8] M. Krzywinski, .portknocking.org,. URL: http://www.portknocking.org, accessed Nov.

[9] http://new.linuxjournal.com/articles/web/2003-06/6811/6811l1.htm