Method for Efficient Content-Based Retrieval of Relative data by filtering out the non-qualifying data objects

Prof. Dr. Sudan Jha
School of Computer Engineering, Campus 15
KIIT University, Patia Bhubaneswar - 751024

Abstract- Many rule discovery algorithms in data mining generate a large number of patterns/rules, sometimes even exceeding the size of the underlying database, with only a small fraction being of interest to the user It is generally understood that interpreting the discovered patterns/rules to gain insight into the domain is an important phase in the knowledge discovery process. However, when there are a large number of generated rules, identifying and analyzing those that are interesting becomes difficult. For example, providing the user with a list of association rules ranked by their confidence and support 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.

Therefore, to be useful, a data mining system must manage the generated rules by offering flexible tools for rule selection. In the case of association rule mining, several approaches for the post processing of discovered association rules have been discussed. One approach is to group "similar" rules which work well for a moderate number of rules. However, for a larger number of rules it produces too many clusters. A more flexible approach is to allow the identification of rules that are of special importance to the user through templates or data mining queries. This approach can complement the rule grouping approach and has been used to specify interesting and uninteresting classes of rules (for both association and episodic rules).The importance of data mining queries has been highlighted by the introduction of the inductive database concept, which allows the user to both query the data and query patterns, rules, and models extracted from these data.

Keywords – Algorithms, Data mining, Selection, Association rule, Data Mining Queries, 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, an inverted file consists of a directory containing all distinct values in D and, for each value in the directory, an inverted list that stores a list of references to all occurrences of this value in the database, that is a 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 $\langle n; tid_1, \cdots, tid_n \rangle.$, 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 $\langle 2; 1, 4 \rangle$ 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 $\langle 2; (1, 3), (4, 2) \rangle$effective, it is not a perfect solution.

After an inverted file has been built, subset queries can be processed as: for each item in the query set the appropriate list is fetched, and then all those lists are intersected. The result of intersection contains a list of references to sets that contains the query set. The equality queries are processed the same way as with subset queries, but can be improved by eliminating all references to target sets whose cardinality is not equal to the query set cardinality. 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).

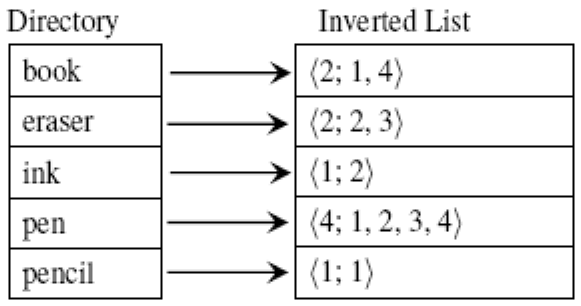

Figure 1 File index creation

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

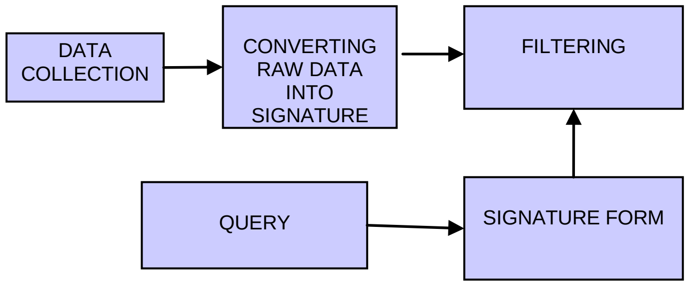

Figure 2: - Data Flow Diagram

1. Converting the raw database into equivalent id-based file.
2. Id based database is converted into equivalent signature file.

In this module the query is converted into equivalent signature form i.e. into bit form In this module the (content-based queries), Let D be a temporal pattern database and q be a query pattern. The four forms of content-based queries that this research supports include the following:

1. Sub pattern queries. Find those patterns in D that contain q.
2. Super pattern queries. Find those patterns in D that are a sub-pattern of q.
3. Equality queries. Find those patterns in D equal to q.
4. K-nearest subpattern queries. Find the k most similar patterns in D to q. Super pattern queries are useful when searching for the characteristic parts of a large pattern, while k-nearest sub pattern queries limit the number of patterns generated by sub pattern or super pattern queries. The temporal pattern is converted into equivalent sets.

```
Input: A database D of temporal patterns; Output: SignatureFile
1: for each p 2 D do
2: EðpÞ ¼ Equivalent SetðpÞ
3: sigp ¼ SignatureðEðpÞÞ
4: Insert hsigp; pidpi into SignatureFile
5: end for
6:return SignatureFile
```

```
Pseudocode of evaluateSubPattern using BSSF
Input: Temporal pattern database D, a query pattern q
Output: AnswerSet
1: EðqÞ ¼ Equivalent SetðqÞ
2: sigq ¼ SignatureðEðqÞÞ
3: Retrieve the bit slices corresponding to the bit position
set to “1” in sigq
4: Perform a bitwise intersect operation on the retrieved
bit slices
5: for each entry where “1” is set in the resulting intersect
bit slice do
6: Add the corresponding pidp into the PID list
7: end for
8: for each pidp in the PID list do
9: Retrieve p from D
10: if p w q then
11: Add p into AnswerSet
12: end if
13: end for
14: return AnswerSet
```

```
Pseudocode of evaluateSuperPattern using BSSF
Input: Temporal pattern database D, a query pattern q
Output: AnswerSet
1: EðqÞ ¼ Equivalent SetðqÞ
2: sigq ¼ SignatureðEðqÞÞ
3: Retrieve the bit slices corresponding to the bit position
set to “0” in sigq
4: Perform a bitwise union operation on the retrieved bit
slices
5: for each entry where “0” is set in the resulting union bit
slice do
6: add the corresponding pidp into the PID list
7: end for
8: for each pidp in the PID list do
9: Retrieve p from D
```

```
10: if p v q then
11: Add p into AnswerSet
12: end if
13: end for
14:return AnswerSet
```

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: -

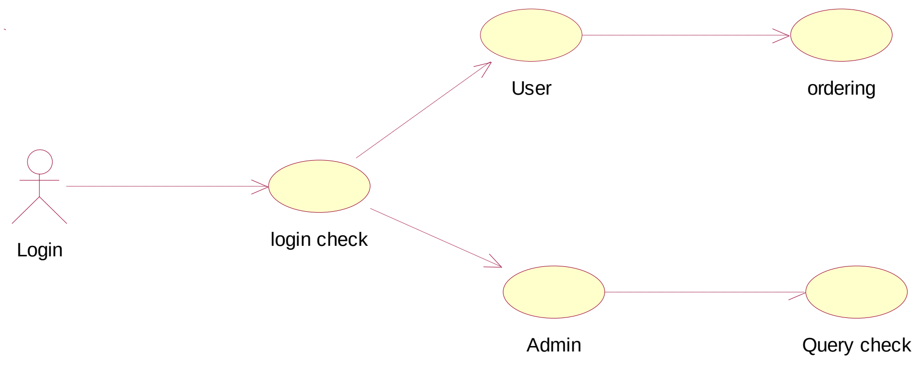

Figure 3: - The UML Diagram

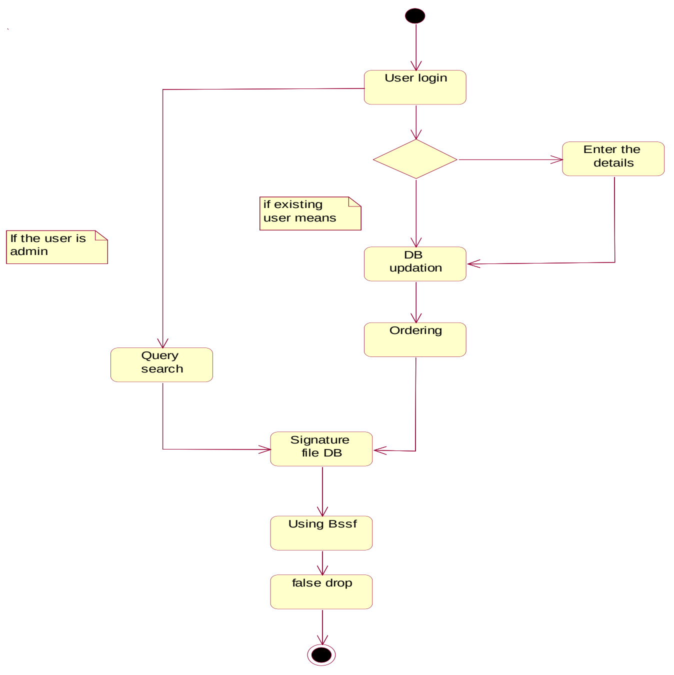

Figure 4: - The Activity Diagram

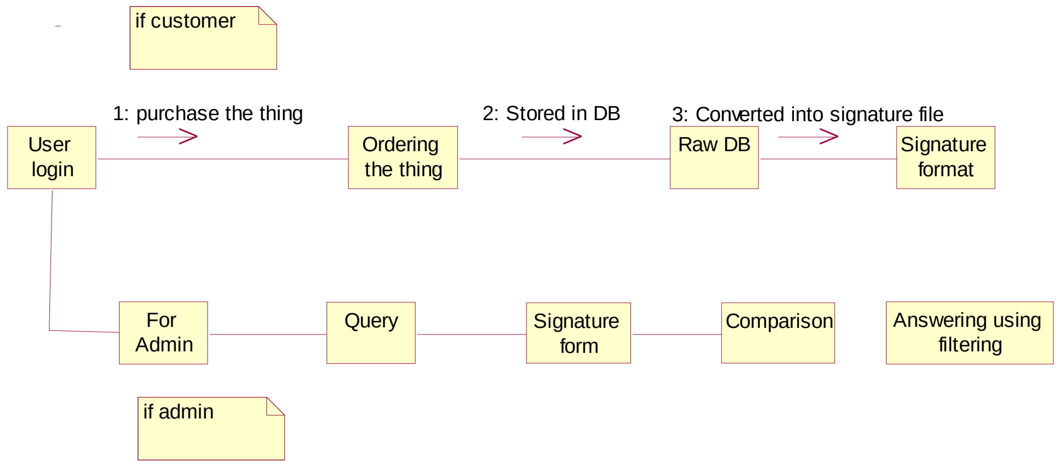

Figure 5: - The Collaboration Diagram

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.

VI. Methods For Generating Set Signatures

In set retrieval with signature files, a target signature is generated for each target set and 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). The description of each method for generating target signatures is given below.

1. Word signature (WS). In the WS method each element of the target set is hashed into a bit pattern of a certain length. These patterns, called word signatures, are then concatenated to form the target signature.

2. Superimposed Coding (SC). In the SC method, each element in a target set is hashed to a binary bit pattern called an element signature. All element signatures have F bit length, and exactly m bits are set to '1', where m < F. F is called 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.

3. 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.

VII. Conclusion

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.

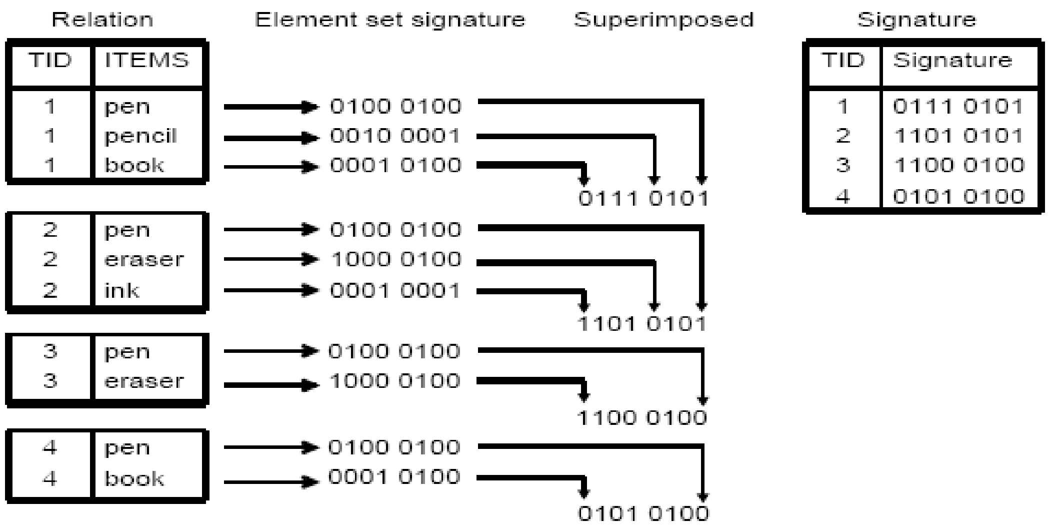

Figure 7 Generating signature using superimposed coding

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