FINDING EXCEPTION FOR ASSOCIATION RULES VIA SQL QUERIES

Luminita DUMITRIU

Department of Computer Science and Engineering, "Dunarea de Jos" University, str. Domneasca nr.47, Galati 6200 Romania.

Abstract: Finding association rules is mainly based on generating larger and larger frequent set candidates, starting from frequent attributes in the database. The frequent sets can be organised as a part of a lattice of concepts according to the Formal Concept Analysis approach. Since the lattice construction is database contents-dependent, the pseudo-intents (see Formal Concept Analysis) are avoided. Association rules between concept intents (closed sets) A=>B are partial implication rules, meaning that there is some data supporting A and (not B); fully explaining the data requires finding exceptions for the association rules. The approach applies to Oracle databases, via SQL queries.

Keywords: data mining, association rules, exceptions, SQL.

1. INTRODUCTION

The data mining research has very much grown in the last decade. The major applications attempt to define customer's profile (for retail, transportation or other services), to detect frauds (for healthcare&insurance or telecommunications (Cox, *et al.*, 1997)), to analyse credit risks (in the financial domain), etc. The discovery of association rules, one of the most researched techniques for data mining, attempts finding frequent patterns among large sets of data attributes. Most of the recent work has been concentrated on developing efficient mining algorithms (Agrawal, *et al.*, 1993, Agrawal, *et al*, 1996, Brin, *et al*, 1997, Dumitriu, *et al*, 2000, Lin and Dunham, 1998, Park, Chan and Yu, 1995, Savasere, Omiecinski and Navathe, 1995, Toivonen, 1996, Zaki, *et al.*, 1997). There are also efforts on expressing a coherent association theory (Zaki, *et al.*, 1998). A major part of the research was focused on dealing with transactional databases, but there are important contributions in the study of relational database systems (Agrawal and Shim, 1996, Nestorov and Tsur, 1997, Sarawagi, Thomas and Agrawal, 1998, Thomas and Sarawagi, 1998).

2. FREQUENT SETS AND ASSOCIATION RULES

The description of the association rules mining was first given by Agrawal et al. (1993). The set of *items* or *attributes* are designated by the literals $I = \{ I_1, I_2, \ldots, I_n\}$. A *record* (or *transaction*) contains some of the items of I, for the transactional data base case, or contains their presence information, for the relational data base case. We will denote this relation through the inclusion operator, $\subset$. The input data for the mining algorithms consists in a set of records. Any set of items of I is called an *itemset*. An association rule is a relation between itemsets, $A \Rightarrow B$, where A and B are contained in some transaction, and $A \cap B = \emptyset$. A is the antecedent of the rules, and B is the consequent.
An itemset is associated with a measure of frequency, called *support*, and support (X) denotes the ratio between the number of records that contain X and the total number of records in the data set. For a rule, the support measure refers to the $A \cup B$ set. The strength of an association $A \Rightarrow B$ is measured by the *confidence* of the rule determined as support $(A \cup B)$/support (A).

This paper was recomended for publication by V. Mînzu

Mining association rules is finding all the rules that exceed two user-specified thresholds, one for support, min_sup, and one for confidence, min_conf. An itemset that exceeds the support threshold is a *large itemset*. Let S be a large itemset, for any $A \subset S$ and support (S)/support (A)$\geq$min_conf, $A \Rightarrow S\text{-}A$ is an association rule. Therefore, classically finding association rules consists in two stages:

- Discovering all large itemsets. This stage is classically split into two parts: candidate-generation step, of an incremental manner, and large item selection, counting the support of the candidates and pruning the ones that are not large;
- Determining the rules with enough confidence.

The main algorithms are sequential or parallel, running on the entire data set or only on a training set, use different approaches to reduce the number of data base scans or the amount of storage memory.

3. FORMAL CONCEPT ANALYSIS

The theory of formal concept analysis was introduced by Wille (1982), and correlated with association rules mining by Zaki and Ogihara (1998). Let I be the set of items and let T be the set of records. Let s be a mapping between the power set of I and the power set of T, which associates to a set of itemsets all records that contain at least one of them. Let t be a mapping between the power set of T and the power set of I, that associates to a set of records all itemsets contained in them. The composition $c=t \circ s$ is proven to be a closure operator.

The context (T, I, $\subset$) and the mappings s and t define a Galois connection between $\wp$(I) and $\wp$(T).

A concept in this context is a pair (X, Y) of closed sets, where $X \subseteq T$ and $Y \subseteq I$, with t(X)=Y and s(Y)=X (according to this, c(X)=X and c(Y)=Y, so X and Y are closed sets). X is the extent of the concept, while Y is the intent of the concept.

Every context (T, I, $\subset$) can be associated with a Galois lattice of concepts, with join and meet operators derived from the closure operator, c. The Galois lattice can be represented by a Hasse diagram. Between a pair (X_1, Y_1) and (X_2, Y_2) of concepts, the relation $(X_1, Y_1) \geq (X_2, Y_2)$ means that $Y1 \subset Y_2$ and $X_1 \supset X_2$. A frequent concept has support(X) $\geq$min_sup. All frequent itemsets are uniquely determined by the frequent concepts. There can be frequent itemsets that are not closed sets, but they are included in closed sets and are sharing the same support. These itemsets do not need to be generated (though, classical algorithms do generate them). They are called *pseudo-intents*.

A *partial implication rule* (c_1, c_2, conf) is associated with a pair of concepts that satisfy $c_1 \geq c_2$, where conf is the *precision* determined as support(Y_2)/ support(Y_1).

Association rules are represented at the intent level of a concept, as $Y_1 \Rightarrow Y_2\text{-}Y_1$, with c_2 frequent and p$\geq$min_conf. Whenever Y_1 is a pseudo-intent and Y_2 is its intent, we have a global implication rule, with conf=1 (due to the same support).

Note. If (c_1, c_2, p) and (c_2, c_3, q) are implication rules, (c_1, c_3, p*q) is also an implication rule.

4. DISCOVERY OF ASSOCIATION RULES FROM RELATIONAL DATABASES

The dawn of association rules mining was focused on the transactional data bases, mainly to mine the market basket data. The last few years a new interest is shown into mining relational databases (Agrawal and Shim, 1996, Chamberlin, 1996, Dumitriu, *et al*, 2000, Nestorov and Tsur, 1997, Sarawagi, Thomas and Agrawal, 1998, Thomas and Sarawagi, 1998).

According to Sarawagi, Thomas and Agrawal (1998), there are three different ways to interact with a relational database during the mining process:

- Using a *cursor interface*. There can be two variations to this approach, one called *loose-coupling* (Agrawal and Shim, 1996) and another one called *stored procedure* (Chamberlin, 1996). The former uses different address spaces for the DBMS and for the mining engine, the later encapsulates the mining algorithm into a stored procedure that runs in the same address space as the DBMS.
- Using a temporary *data cache* for the contents of the database, which is discarded when the mining process completes.
- Using *user-defined functions* (UDFs) (Chamberlin, 1996), running in the same address space as the DBMS and appropriately placed into the SQL queries.

Their conclusions were as follows:

- the loose-coupling approach suffers due to the high cost of context switching between the DBMS and the mining engine, but, just like the stored procedure, it offers higher flexibility and no extra storage requirements;
- the data cache promises better performance but requires additional disk space for caching;
- UDFs have better argument passing performance than stored procedure, but since they are doing most of the processing, there can be high development costs (involved by the significant code rewrites (Agrawal and Shim, 1996)).

All these algorithms share the same mining process as the classical Apriori: generating candidates and counting support for determining the large itemsets and, afterwards, generating association rules.

5. ASSOCIATION MEASURES

The measures mentioned in the domain literature (Kodratoff, 1998) that are used for the evaluation of $A \Rightarrow B$ relations are:

- measures of statistical significance;
- unexpectedness measures.

Some statistical measures are the support, the confidence, the interest and the conviction of a relation.

The interest of $A \Rightarrow B$, $int(A \Rightarrow B)$, is defined as the confidence of the relation divided by the support of B, $supp(A \Rightarrow B)/supp(A)*supp(B)$. This measure evaluates the pair (A, B) and not the implication, since its formula is commutative.

The conviction of $A \Rightarrow B$, $conv(A \Rightarrow B)$, is defined as $supp(A)*supp(\neg B)/ supp(A \Rightarrow \neg B)$, meaning $1/int(A \Rightarrow \neg B)$. This is commutative too, measuring the weakness of the pair $(A, \neg B)$.

There are several unexpectedness measures, for example, the implication intensity, the exception measure and the surprise measure.

The implication intensity is computed by comparing the cardinal of A=>(D-B), meaning the number of elements contradicting A=>B, with the cardinal of A'=>(D-B'), where $|A'|=|A|$, $|B'|=|B|$ and A' and B' are picked randomly. This comparison should show how strong is the implication versus a random one.

The exception measure searches for complementary implications for A=>B, as $A \cup A' \Rightarrow \neg B$, trying to cover the exceptions of A=>B in the data. Generally, the complementary implications should have a small support (due to the strong association between A and B), but a high confidence.

The surprise measure searches pairs of implications as follows: for A=>B with low interest (ordinary association), $A \cup \neg A' \Rightarrow B$ with high interest. Comparing the interest expressions, we observe that the difference comes from the confidence measure of the implications (since supp(B) is the same). This means that the A=>B association works better under certain restrictions.

6. MY APPROACH

The main difference with the approaches mentioned above is caused, on one hand, by the formal concept analysis theory. In a traditional algorithm there is always found the incremental manner in which, starting from frequent items, larger and larger candidates are built and validated against the data base contents. This leads to the generation of pseudo-intents even though they do not reflect the actual data contents. On the other hand, the main tool in accessing a DBMS, the SQL, is not a procedural language. This makes implementing an Apriori-like algorithm unnatural.

First we will present some observations.

Lemma 1. Any transaction (record) T in the database is the intent of a concept.

Proof. Let's assume that T is not an intent. Consequently, there exists an itemset T' in the database, where $T \subset T'$ and supp(T)=supp(T'). Since $T \subset T'$, the T' transaction contributes to T 's support, but T does not contribute to the support of T', therefore supp(T)>supp(T'). Contradiction. QED.

Any itemset s has a transaction set that contributes to its support. If we consider s', the intersection of the transactions in this set, we can prove that s' is a closed set.

Lemma 2. Any closed itemset is equal to the intersection of the transactions, represented as sets of present items, in its supporting transaction set.

Proof. Let {T1, T2, …, Tn} be the supporting transaction set for a closed itemset s and $s'=T1 \cap T2 \cap \ldots \cap Tn$ be the intersection itemset. Since {T1, T2, …, Tn} is the supporting transaction set for s, for any i=1..n, $s \subseteq Ti$, i.e. $s \subseteq s'=T1 \cap T2 \cap \ldots \cap Tn$ and, subsequently, $supp(s) \geq supp(s')$. But $s'=T1 \cap T2 \cap \ldots \cap Tn$, so for any i=1..n, $s' \subseteq Ti$, therefore all Ti belong to the supporting transaction set for s'. Consequently, $supp(s') \geq supp(s)$. Considering that $supp(s) \geq supp(s')$, $supp(s') \geq supp(s)$ and $s \subseteq s'$ we can state that, unless s=s', s is a pseudo-intent. QED.

Lemma 3. The non-empty intersection of two closed itemsets (intents) is also a closed itemset.

Proof. Let s1 and s2 be the two closed itemsets and S1 and S2 their supporting transaction sets. According to Lemma 2., each of s1 and s2 can be written as the intersections between all transactions in their supporting set. The intersection between s1 and s2, namely s, becomes the intersection between transactions in $S1 \cup S2$. Let S be the supporting transaction set for s. Since $s \subseteq s1$ and $s \subseteq s2$, both S1 and S2 are included in S, therefore $S1 \cup S2 \subseteq S$. Let's assume that s is a pseudo-intent, so there exists s', where $s \subset s'$ and they share the same supporting transaction set. Because $S1 \subseteq S$ and $S2 \subseteq S$, we can infer that $s' \subseteq s1$ and $s' \subseteq s2$. This leads to $s1 \cap s2 = s \subset s' \subseteq s1 \cap s2$. Contradiction. QED.

6.1. Finding Exceptions

In our view, we will use two queries (one for the management of the data base structure) and a user-defined function for format conversion purposes to determine all large intents, results denoted as "large_intents" (Dumitriu, *et al*, 2000).

Further, we build all association rules, as proper partial implications, $Y_1 \Rightarrow Y_2 - Y_1$, where Y_1 and Y_2 are large intents and $Y_1 \subset Y_2$, selecting those with enough confidence.

If we recall the observation in section 2, we can notice that association rules between intents of non-adjacent concepts can be derived from those between intents of adjacent concepts. To spear the storage

space we can query for the association rules between the intents of adjacent concepts, alias “generating_set”, inferring the others afterwards.
The results of the association mining, at this stage, account for proper partial implications. These implications can not be considered logical implications, since there can be some data supporting A∪¬B. Therefore, finding the exceptions to the rule can be extremely revealing for the user. We remind that the exception implication is due to low support, observation that leads to the necessity of mining all intents, not only the large ones. Thus, from the mining SQL query we remove the support threshold constraint. The alias of these results will be "all_intents".
After generating all association rules (for reasons of clarity we will designate these results as "association_rules") one should try finding the exceptions to one of the rule. For each association rule, we are trying to find the itemset C that, together with A, covers best the A∪¬B area of data. The A∪¬B has supp(A)-supp(A∪B) data support. The confidence of the exception rule, expressed by supp(A∪C∪¬B)/ supp(A∪C), should be high. Thus, whenever one record contains both A and C, it would also not contain B. Covering the A∪¬B area of data, might demand several C itemsets, or a best choice of C. In our implementation we have selected the later, so evaluating this best choice means finding the C itemset, with low support for A∪C, for which supp(A∪C∪¬B), expressed as supp(A∪C)-supp(A∪C∪B), is closest to supp(A)-supp(A∪B).

```
-- mining the exceptions for the association rules
select t1.a, t1.ab, t3.ac,
        min(evaluate(t1.sup_a, t1.sup_ab,
        t3.sup_ac, t2.sup_abc)) from
        (select a, sup_a, ab, sup_ab from
        association_rules) t1,
        (select  a abc, sup_a sup_abc from
        all_intents) t2,
        (select  a ac, sup_a sup_ac from
        all_intents) t3,
where t1.ab like intersect(t2.abc, t1.ab) AND
        get_ac_from(t1.a, t1.ab, t2.abc) like t3.ac
group by t1.a, t1.ab;
```

The get_ac_from function processes the A, A∪B and A∪B∪C itemsets representations in order to determine the corresponding A∪C itemset.
The "evaluate" function computes an expression that combines:

- the square of the support value for A∪B∪C (a low support for A∪B∪C implies that A∪C does not account for the presence of B)
- and the square of (supp(A) - supp(A∪B)) - (supp(A∪C) - supp(A∪B∪C)) (that quantifies the way A∪C covers the A∪¬B area of data).

We are studying different forms for this expression, in order to analyze the efficiency of the selection.
If the user chooses to store only the "generating_set" instead of "association_rules", we can always infer new rules. Finding exceptions for them means modifying the query to act on a specified association rule, as follows:

```
-- mining the exceptions for the association rules
select t3.ac, min(evaluate(sup_a, sup_ab,
        t3.sup_ac, t2.sup_abc)) from
        (select  a abc, sup_a sup_abc from
                all_intents) t2,
        (select  a ac, sup_a sup_ac from
                all_intents) t3,
where ab like intersect(t2.abc, ab) AND
        get_ac_from(a, ab, t2.abc) like t3.ac;
```

Since we are considering building dynamic queries, there is no problem in filling the a, b, sup_a and sup_b values with the ones of the selected association rule.
The minimum value of the "evaluate" function that leads to the best exception may not be good enough. This is either due to the not so low support of A∪B∪C, or due to the way A∪C covers the A∪¬B area of data. In the later case, we are also thinking of finding several disjunctive exceptions to achieve the cover of the A∪¬B area of data.

6.2. Maintaining Association Rules

Sometimes, maintaining the large itemsets can be more important then discovering them. The on-line approach we consider consists in creating database triggers on *insert*, *delete* and *update* commands on the database. On these triggers we update the "all_intents" results.
On insert commands, the new record will be considered for intersection with every intent. The intersection may be:

- equal to the intent, in which case the support of the intent will be incremented;
- equal to another intent, in which case no action is taken;
- different from all intents, in which case the new intent is inserted in the "all_intents" database and its support is counted.

On delete commands, the record will be considered for intersection with every intent. The intersection may be:

- equal to the intent, in which case the support of the intent will be decremented;
- equal to another intent, in which case no action is taken.

Afterwards, all intents with 0 support are discarded.
On update commands, a delete command for the old record and an insert command for the new record will be considered. Discarding the no-support intents will be accomplish after processing the commands.

7. CONCLUSIONS AND FUTURE WORK

Our approach is fundamentally different then all the work in finding association rules from relational databases. All other approaches tried to implement the classical procedural algorithms in SQL. Since SQL is not a native procedural language, implementing procedural algorithms is unnatural. More, the classical algorithms find artificial results (the pseudo-intents) that are not really supported by the data. These artificial results increase the number of generated association rules with items that have no significance (rules with a confidence of 100%, called global implications, that suggest redundancy). Still we will take into account, as further development, generating the global implications, too.

At this moment our research is not yet fully developed as an application. We are using the JDBC to link the dynamically generated SQL queries with the rule generation engine and the user interface.

We are also considering the dynamical selection of the data to be mined from several relations of the database.

8. REFERENCES

Agrawal, R., Imielinski, T. and Swami (1993) "Mining association rules between sets of items in large databases", in Proceedings of 1993 ACM SIGMOD International Conference on Management of Data, Washington D.C., pp. 207-216.

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H. and Inkeri Verkamo, A. (1996) "Fast discovery of association rules", in Advances in Knowledge Discovery and Data Mining, ed. U. Fayyad et al., AAAI Press: Menlo Park, CA, pp. 307-328.

Agrawal, R. and Shim, K. (1996) "Developing Tightly-Coupled Data Mining Applications on a Relational Database System", in Proceedings of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining, ed. E. Simoudis et al., AAAI Press, Portland, Oregon, pp. 287-295.

Brin, S., Motwani, R., Ullman, J. and Tsur, S. (1997) "Dynamic itemset counting and implication rules for market basket data", in Proceedings of 1997 ACM SIGMOD International Conference on Management of Data, SIGMOD Record, Vol. 26, No.2, pp. 255-263.

Chamberlin, D. (1996) "Using the New DB2: IBM’s Object-Relational Database System", Morgan Kaufmann, San Francisco, pp. 207-216.

Cox, K., Eick, S., Wills, G. and Brachman, R. (1997) "Visual Data Mining: Recognizing Telephone Calling Fraud", in Data Mining and Knowledge Discovery Journal, Kluwer Academic Publishers, vol. 1, pp. 225-231.

Dumitriu, L., Pecheanu, E., Istrate, A. and Segal, C. (2000) “Finding association rules from relational databases”, in Proceedings of the International Conference Data Mining 2000, Cambridge, pp..

Dumitriu, L., Pecheanu, E., Istrate, A. and Segal, C.(2000) "Finding association rules from relational databases via SQL queries", in Control Engineering and Applied Informatics Journal, Mediamira Science Publisher, vol. 2, pp.59-64.

Houtsma, M. and Swami, A. (1995) "Set-oriented mining of association rules in relational databases", in Proceedings of the 11th International Conference on Data Engineering, IEEE Computer Society Press, Los Alamitos, pp.25-34.

Kodratoff, Y. (1998) "Research Topics in Knowledge Discovery in Data and Texts", invited talk at the 3rd SIGMOD’98 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD), Seattle, WA.

Lin, D-I. and Kedem, Z.M. (1998) "Pincer-Search: a new algorithm for discovering the maximum frequent set", in Proceedings of the 6th International Conference on Extending Database Technology, Lecture Notes in Computer Science, Springer-Verlag, 1377, pp.105-113.

Lin, J-L. and Dunham, M.H. (1998) "Mining association rules: Anti-skew algorithms", in Proceedings of the 14th International Conference on Data Engineering, IEEE Computer Society Press, Los Alamitos, pp.125-133.

Nestorov, S. and Tsur, S. (1997) "Using DB2’s Object Relational Extensions for Mining Association Rules", Technical Report TR 03690, Santa Teresa Laboratories, IBM Corporation.

Park, J.S., Chen, M. and Yu, P.S. (1995) "An effective hash-based algorithm for mining association rules", in Proceedings of 1995 ACM SIGMOD International Conference on Management of Data, San Jose, pp.175-186.

Sarawagi, S., Thomas, S. and Agrawal, R., (1998) "Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications", in Proceedings of 1998 ACM SIGMOD International Conference on Management of Data, Seattle, pp. 343-354.

Savasere, A., Omiecinski, E. and Navathe, S. (1995) "An efficient algorithm for mining association rules in large databases", in Proceedings of the 21st International Conference on Very Large Data Bases, ed. U. Dayal et al., Morgan Kaufmann, Los Altos, pp. 432-444.

Thomas, S. and Sarawagi, S. (1998) "Mining Generalized Association Rules and Sequential Patterns Using SQL Queries", in Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, New York, pp. 344-348.

Toivonen, H., (1996) "Sampling large databases for association rules", in Proceedings of the 22nd International Conference on Very Large Data Bases, ed. T.M. Vijayarama et al., Morgan Kaufmann, Los Altos, pp. 134-145.

Wille, R. (1982) "Restructuring lattice theory: an approach based on hierarchies of concepts", in Ordered Sets, Proceedings of NATO Advanced Study Institute, D. Reidel Publisher Co., pp. 445-470.

Zaki, M.J., Parthasarathy, S.,Ogihara, M. and Li, W., (1997) "New algorithms for fast discovery of association rules", in Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, ed. D. Heckerman et al., AAAI Press, pp.283-29.

Zaki, M.J. and Ogihara, M. (1998) "Theoretical Foundations of Association Rules", in Proceedings of the 3rd SIGMOD’98 Workshop on DMKD, Seattle, WA, pp 7:1-7:8.