Mining Disjunctive Association Rules

Marcus C. Sampaio, Fernando H. B. Cardoso, Gilson P. dos Santos Jr., Lile Hattori

15 aug. 2008
Abstract

Association rule mining is one of the most important and well-researched techniques of data mining, since the seminal papers by R. Agrawal et al. [1, 2]. It aims to induce associations among sets of items in the transaction databases or other data repositories. Ever since, several algorithms for specialized association tasks have appeared: quantitative association rules, generalized association rules, association rules extended with negation, among many other specializations. To the best of our knowledge all these existent approaches that enhance the classic model of association rules do not still cover other important specialized association tasks. In this paper, we propose and justify a model of disjunctive association rules, accompanied by an algorithm that induces rules in accordance with this model. In at least one type of practical application, the algorithm showed that it is efficient and even indispensable: it is the problem of software change impact analysis.
Chapter 1

Introduction

Association rule mining is one of the most important and well-researched techniques of data mining, since the seminal papers by R. Agrawal et al. [1, 2]. It aims to induce associations among sets of items in transaction databases or other data repositories. Essentially, an (conjunctive) association rule is an implication in the form of $X \Rightarrow Y$, where $X$ (the antecedent rule) and $Y$ (the consequent rule) are sets of items, and $X \cap Y = \emptyset$. Two user-defined thresholds for association rules are called minimal support and minimal confidence. Support $s$ of an association rule is defined as the percentage/fraction of records that contain $X \cup Y$ to the total number of records in the database or repository. Confidence $c$ of an association rule is defined as the percentage/fraction of the number of transactions that contain $X \cup Y$ to the total number of records that contain $X$. If $s$ and $c$ exceed minimal support and minimal confidence values, respectively, then the association rule $X \Rightarrow Y$ must be induced.

Sometimes, it is more comfortable to probabilistically ratiocinate on association rules. The link between association rule and probability is a corollary of the support and confidence definitions: $s(X \Rightarrow Y) \equiv P(X \cup Y)$ and $c(X \Rightarrow Y) \equiv P(Y \mid X)$. ($P(X \cup Y)$ is the probability that all the items in $X \cup Y$ are present in a transaction; $P(Y \mid X)$ – conditional probability – is the probability of $Y$ given that $X$ has occurred.)

Since the middle of the 90’s several algorithms for specialized association tasks have been developed: quantitative association rules [10], generalized association rules [9], and to some extent the work on sequential patterns [8, 4] and association rules extended with negation [11], among many other papers. To the best of our knowledge all these existent approaches that enhance the classic model of association rules do not cover other important specialized association tasks: disjunctive association rules. In this paper, we propose a disjunctive association rules (DAR rules) algorithm. A motivational example for the need of DAR rules is shown below.

Object-oriented software systems are constantly subjected to changes. Before the execution of a change it is important to plan and estimate its cost. In order to estimate the cost, software engineers need to identify in the software which other elements must be also changed to maintain the software’s integrity. This activity is called software change impact analysis [3]. One source of information apart from source code is the software repository from which information about what elements have changed together in the past can be ob-
tained through the use of a data mining algorithm. Notice that a repository is usually very voluminous, what turns prohibitive its manual inspection.

Typically, a software engineer seeks answers for two types of questions: (1) Given the commit of the class A, what is the probability that classes B, or\(^1\) C, or D are also atomically committed with A?; and (2) If classes B or C or D are committed, what is the chance that A is also committed?

The first question could be formulated in terms of this rule:

\[
A \Rightarrow B \ OR \ C \ OR \ D, \ s > s_{\text{min}} \ c = P((B \ OR \ C \ OR \ D) \ | \ A) > c_{\text{min}} \tag{1.1}
\]

For the second question we could have this other rule:

\[
B \ OR \ C \ OR \ D \Rightarrow A, \ s > s_{\text{min}} \ c = P(A \ | \ (B \ OR \ C \ OR \ D)) > c_{\text{min}} \tag{1.2}
\]

Both rules have the same support \( s = P(A \cap (B \ OR \ C \ OR \ D)) \).

The rules \( A \Rightarrow B \ OR \ C \ OR \ D \) and \( B \ OR \ C \ OR \ D \Rightarrow A \) are examples of DAR rules. DAR rules can be induced by our new algorithm of disjunctive association rules – DAR algorithm – that is the focus of this paper.

An important challenge to be faced in the construction of the DAR algorithm is the potential explosion of disjunctive association rules, in much larger degree than the known problem of explosion of conjunctive association rules \([2]\). To see why, consider again the first of the above rules, and also the 3 rules \( A \Rightarrow B; \ A \Rightarrow C; \) and \( A \Rightarrow D \). It is clear that, even though

\[
\begin{cases}
  s(A \Rightarrow B) < s_{\text{min}} \\
  s(A \Rightarrow C) < s_{\text{min}} \\
  s(A \Rightarrow D) < s_{\text{min}}
\end{cases}
\tag{1.3}
\]

it is still possible that

\[
s(A \Rightarrow B \ OR \ C \ OR \ D) > s_{\text{min}} \tag{1.4}
\]

Another important discussion raised by disjunctive association rules is the interest of the rules. Given, for example, these two rules

\[
\begin{cases}
  A \Rightarrow B, \ (I) \\
  A \ OR \ C \Rightarrow B, \ (II)
\end{cases}
\tag{1.5}
\]

rule II is interesting or not? The least that one can say is that if support of rule I is equal to support of rule II then rule II is redundant (when C exists A also exists), and therefore, rule II should not be interesting. If, however, support of rule II goes higher than the one of rule I then rule II is not redundant and it would be interesting.

The remainder of this paper is organized as follows. We comment some related work in Chapter 2 and formalize our model of DAR rules in Chapter 3. Our algorithm DAR is described in Chapter 4, and its performance is evaluated in Chapter 5. Finally, in Chapter 6 we summarize and conclude our work.

\(^1\)Not-exclusive OR.
Chapter 2

Related Work

The literature about association rule mining is vast. In this chapter, we restrict the discussion to the closest works of ours, considering the current state and the perspectives of our research in disjunctive association rule mining. The chapter finishes with considerations about the complexity, in general, of algorithms of association rules.

Algorithm Apriori [2]. The problem of inducing association rules between items in a large database of sales transactions – market basket analysis – is the goal of the algorithm Apriori and of a light modification of it, AprioriTid. Their classic model of association rules is how explained in the beginning of Chapter 1: conjunctive rules, with the metrics support and confidence. The best features of the two algorithms can still be combined into a hybrid algorithm, called AprioriHybrid. At the end of the paper, the authors already pointed out the need of enhancing the model with multiple taxonomies and other improvements. Along the time, association rules showed to be very useful in other several application domains.

Generalized Association Rules [9]. An interesting extension of the seminal algorithm Apriori and its two congeners, AprioriTid and AprioriHybrid, is like this: given a large database of transactions composed of a set of items, and a taxonomy (is-a hierarchy) on the items, the Cumulate algorithm finds associations between items at any level of the taxonomy. For example, given a taxonomy that says that jackets is-a outerwear is-a clothes, the algorithm may infer a rule that “people who buy outerwear tend to buy shoes”. This rule may hold even if rules that “people who buy jackets tend to buy shoes,” and “people who buy clothes tend to buy shoes” do not hold.

Query Flocks [11]. Intuitively, a query flock is a generate-and-test system, in which a family of queries on association rules that are identical, except for the values of one or more parameters, are asked simultaneously. The answers to these queries are filtered and those that pass through the filter test enable their parameters to become part of the answer to the query flock. The setting for a query flock system is:

- A language in which we can express association rules as queries that are parametrized by one or more parameters. The extensions of conjunctive clauses are negated sub goals and arithmetic sub goals;

- A language to express filter conditions about the results of a query.
One of the examples in [11] is on a medical relational database:

**QUERY**

```prolog
answer(P) :-
exhibits(P,$s) AND
treatments(P,$m) AND
diagnoses(P,D) AND
NOT causes(D,$s)
FILTER
  COUNT(answer.P) >= 20
```

whose interpretation is: give me the set of patients P that exhibit a symptom $s$, are receiving medicine $m$, have disease $D$, and yet disease $D$ doesn’t explain symptom $s$. The filter requires that there are at least 20 patients taking medicine $m$ and exhibiting unexplained symptom $s$.

Reliable Models of Associations [12]. A. Veloso et al. address the problem of inducing reliable models within the context of association mining over time. This is a challenging problem because a model, no matter how accurate, can only predict based on associations mined in the old data. A new approach is proposes which, by detecting how the associations are evolving over time, is able to induce more reliable models in dynamic databases.

Apriori-based Pattern Mining [4]. De Amo et al. investigate sequential patterns whose specification needs the first-order temporal logic. They focus on the problem of mining multi-sequential patterns that are not expressible in propositional temporal logic. They propose two Apriori-based algorithms, PM (Projection Miner) and SM (Simultaneous Miner), to perform this mining task.

A problem common to any algorithm of association rules is its complexity, that is always high. Complexity is more discussed in the next chapter.

### 2.1 On the Complexity of Algorithms of Association Rules

The task of association rule mining can be reduced to the problem of finding all itemsets that are frequent with respect to a minimal support [2]. For practical applications, looking at the whole search space is doomed to failure. In fact, a linearly growing number of items still implies an exponential growing number of itemsets, and this is of the nature of the problem. In [6], the authors investigate the efficiency of six algorithms of association mining by carrying out several runtime experiments. It turns out that the runtime behavior of the algorithms is much more similar as to be expected. In summary, (1) the times of the algorithms are approximately the same ones, and (2) no algorithm behaves very well when it induces association rules with more than 10 items on average and with low support.

The conclusion of all this is that the basic optimization rule of algorithms of association rules, as it is proposed in [2], is very known and have been implemented, and that these
rules are the only ones that seem to be essential, although the complexity of the algorithms of association rules is still exponential. The basic optimization rule will be reviewed at the time of the discussion of our DAR algorithm.
Chapter 3

Formal Model Of Disjunctive Association Rules

In this chapter, we precise: (1) our model of disjunctive association rules (DAR rules); (2) the notion of interesting DAR rule; and (3) the DAR problem statement.

Let \( X = \{i_1, i_2, \ldots, i_n\} \) be a set of items. \( X_C = i_1 \ AND \ i_2 \ AND \ldots \ AND \ i_n \) – or simply \( i_1i_2\ldots i_n \) – is a conjunctive term of \( X \), and \( X_D = i_1 \ OR \ i_2 \ OR \ldots \ OR \ i_n \) is a disjunctive term of \( X \).

A DAR rule is of one of the following types, involving conjunctive and disjunctive terms from the set of items \( X \cup Y \mid X \cap Y = \emptyset \):

<table>
<thead>
<tr>
<th>Type</th>
<th>Support</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_C \Rightarrow Y_C )</td>
<td>( s = P(X_C \cup Y_C) )</td>
<td>( c = P(Y_C \mid X_C) = \frac{s}{P(X_C)} )</td>
</tr>
<tr>
<td>( X_C \Rightarrow Y_D )</td>
<td>( s = P(X_C \cup Y_D) )</td>
<td>( c = P(Y_D \mid X_C) = \frac{s}{P(X_C)} )</td>
</tr>
<tr>
<td>( X_D \Rightarrow Y_C )</td>
<td>( s = P(X_D \cup Y_C) )</td>
<td>( c = P(Y_C \mid X_D) = \frac{s}{P(X_D)} )</td>
</tr>
<tr>
<td>( X_D \Rightarrow Y_D )</td>
<td>( s = P(X_D \cup Y_D) )</td>
<td>( c = P(Y_D \mid X_D) = \frac{s}{P(X_D)} )</td>
</tr>
</tbody>
</table>

Consider this set of items: \( \{X, Y, Z, V, W\} \). Here are four examples of valid DAR rules:

---

Consider this set of items: \( \{X, Y, Z, V, W\} \). Here are four examples of valid DAR rules:
1. \( XY Z \Rightarrow VW \) - type \( X_C \Rightarrow Y_C \)
2. \( XYZ \Rightarrow V \ OR \ W \) - type \( X_C \Rightarrow Y_D \)
3. \( X \ OR \ Y \ OR \ Z \Rightarrow VW \) - type \( X_D \Rightarrow Y_C \)
4. \( X \ OR \ Y \ OR \ Z \Rightarrow V \ OR \ W \) - type \( X_D \Rightarrow Y_D \)

Here are two examples of invalid DAR rules:
1. \( XY \ OR \ Z \Rightarrow VW \) - this is not a DAR rule
2. \( X \ OR \ YZ \Rightarrow V \ OR \ W \) - this is not a DAR rule

To illustrate the calculation of the support and of the confidence of a DAR rule, consider this repository, representing a set of transactions with five items – \( X, Y, Z, V, W \) – and thirteen transactions:

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>V</th>
<th>W</th>
<th>Quant.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Column “Quant” indicates the amount of times that such a transaction appears in the repository. Support and confidence values of the valid DAR rules from previous example are:

<table>
<thead>
<tr>
<th>DAR Rule</th>
<th>Support ( s )</th>
<th>Confidence ( c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( XY Z \Rightarrow VW )</td>
<td>( s = P(XYZ \cup VW) = 0% )</td>
<td>( c = 0% )</td>
</tr>
<tr>
<td>( XYZ \Rightarrow V \ OR \ W )</td>
<td>( s = P(XYZ \cup (V \ OR \ W)) = \frac{1}{13} \approx 8% )</td>
<td>( c = \frac{\frac{1}{13}}{\frac{1}{13}} = 100% )</td>
</tr>
<tr>
<td>( X \ OR \ Y \ OR \ Z \Rightarrow VW )</td>
<td>( s = P((X \ OR \ Y \ OR \ Z) \cup VW) = \frac{3}{13} \approx 23% )</td>
<td>( c = \frac{\frac{3}{13}}{\frac{4}{13}} = \frac{3}{4} \approx 75% )</td>
</tr>
<tr>
<td>( X \ OR \ Y \ OR \ Z \Rightarrow V \ OR \ W )</td>
<td>( s = P((X \ OR \ Y \ OR \ Z) \cup (V \ OR \ W)) = \frac{1}{13} \approx 8% )</td>
<td>( c = \frac{\frac{1}{13}}{\frac{1}{13}} = 100% )</td>
</tr>
</tbody>
</table>
Henceforth, we will denote a DAR rule simply by \( X \Rightarrow Y \), leaving implied that \( X \) (\( Y \)) can be a conjunctive or disjunctive term.

We are able to derive an interesting conclusion from the example above. The Apriori algorithm, that only induces conjunctive rules, would not have induced any one of the association rules above (the only conjunctive rule in the example has zero support and zero confidence), i.e., rules with \( s > s_{\text{min}} \) and \( c > c_{\text{min}} \) from \{X, Y, Z, V, W\}. However, we can notice the tendency to an explosion of non-conjunctive association rules. A trade-off must exist between the parsimony of Apriori algorithm and the prolixity of DAR. Such verification has led to the concept of interesting DAR rule.

### 3.1 Interesting DAR Rules

It is not enough to quantify the usefulness of the interest of a DAR rule only by focusing on how greater support and confidence values of the rule are from minimum support and confidence. We argue that a DAR rule \( X \Rightarrow Y \) is not interesting if another DAR rule containing the items of the first as subset, and with approximately the same support, can be obtained. In this case, there is no loss of the association among the items.

Returning to the repository of transactions and examples of DAR rules, suppose that the task is to induce, from the repository, DAR rules with \( s_{\text{min}} = 25\% \) and \( c_{\text{min}} = 20\% \). Here are two possible rules:

1. \( X \ OR \ Z \Rightarrow V \quad s = \frac{4}{13} = 31\% \quad c = \frac{4}{13} = 25\% \)

2. \( X \Rightarrow V \quad s = \frac{4}{13} = 31\% \quad c = \frac{4}{13} = 80\% \)

Rule 1 must not be of interest because it is covered by rule 2, and can be considered redundant. In other words, when \( Z \) and \( V \) exists it does not matter if \( X \) exists. This is expressed by the common support of the two rules. A disadvantage of rule 1 is that its confidence is considerably smaller than the one of rule 2, exactly because of the presence of \( X \) in it. Unlike rule 1, rule 2 could be considered interesting, because it is not covered by any other rule.

We can still be more flexible. If support of rule 1 is a little larger than support of rule 2, the influence of \( X \) would continue to be negligible in relation to \( Z \), and rule 1 would also be considered as not interesting. Note that the user must define the notion of “little larger”.

Consider \( X_D' \subseteq X_D \) in the following sense: if \( x' \in X' \) then \( x' \in X \). \( X_D' \Rightarrow Y \) is close to \( X_D \Rightarrow Y \) if there is no \( X_D'' \) between \( X_D \) and \( X_D' \). (The same definition applies for \( X \Rightarrow Y \), close to \( X \Rightarrow Y_D \); and \( X_D' \Rightarrow Y_D' \) close to \( X_D \Rightarrow Y_D \)). Another way to say is that \( X_D \Rightarrow Y \) is an increased rule of \( X_D' \Rightarrow Y \).
**Example 1.** \( Z \Rightarrow V \) is close to \( X \) OR \( Z \Rightarrow V \), or then \( X \) OR \( Z \Rightarrow V \) is an increased rule of \( Z \Rightarrow V \), because \( \{Z\} \subseteq \{X,Z\} \) and \( \{Z\} \) and \( \{X \text{ OR } Z\} \) are both disjunctive terms.

**Corollary.** If \( X_D \Rightarrow Y \) is an increased rule of \( X'_D \Rightarrow Y \) then \( s(X_D \Rightarrow Y) \geq s(X'_D \Rightarrow Y) \). This is true because \( s(X_D) \geq s(X'_D) \).

A DAR rule \( X'_D \Rightarrow Y \) (in the same way, \( X \Rightarrow Y'_D \) and \( X'_D \Rightarrow Y'_D \)) is interesting if and only if it satisfies two conditions:

1. no interesting rule close to it exists;
2. if it has one increased rule \( X_D \Rightarrow Y \)
   
   \[
   (s(X_D \Rightarrow Y) - II) \leq s(X'_D \Rightarrow Y) \leq (s(X_D \Rightarrow Y) + II), \]

   where \( II \) ("interest interval") \( \geq 0 \) is defined by the user.

   (To facilitate the reading, we also say that \( s(X_D \Rightarrow Y) \equiv s(X \Rightarrow Y) \).)

**Example 2.** Suppose \( II = 0.02 \), and note that \( s(X \text{ OR } Z \Rightarrow V) = 0.31 \). \( Z \Rightarrow V \) is interesting because its support is \( 0.29 \leq 0.31 \leq 0.33 \) and it is supposed that other close rule of it does not exist. The interpretation is that the support of \( Z \Rightarrow V \) is approximately the support of its increased rule \( X \text{ OR } Z \Rightarrow V \), and because of this \( Z \Rightarrow V \) is interesting.

**Example 3.** On the other hand, \( X \text{ OR } Z \Rightarrow V \) is not interesting because \( Z \Rightarrow V \) is interesting and close to it.

Figure 3.1 illustrates the notion of interesting rule.

We can now state the DAR problem.

### 3.2 Problem Statement of DAR

Given a set of transactions and a user-specified interest interval, the problem of mining DAR rules is to induce all interesting DAR rules that have support and confidence greater than the user-specified minimum support and minimum confidence respectively.
Figure 3.1: Interesting Rule
Chapter 4

Inducing Disjunctive Association Rules: Algorithm DAR

DAR is an algorithm that induces interesting DAR rules from a data repository. Figure 4.1 gives the algorithm.

The rationale for the algorithm is as follows. Suppose the DAR rule \( A (\text{antecedent}) \Rightarrow C (\text{consequent}) \). If \( s(A \Rightarrow C) \geq s_{\text{min}} \) then obviously \( s(A) \geq s_{\text{min}} \) and \( s(C) \geq s_{\text{min}} \), and \( A \) and \( C \) are called \( \text{largeTerm}_A \) and \( \text{largeTerm}_C \). The inverse is more restricted: \( \text{largeTerm}_A \) and \( \text{largeTerm}_C \) are necessary conditions but not enough for \( s(A \Rightarrow C) \geq s_{\text{min}} \). Considering this, the algorithm consists basically of two phases: in phase 1 the supersets \( \text{largeTerms}_{A(C)} \) – \{term \mid (\text{term}_i = \text{largeTerm}_A(C)) \land (\text{term}_i \in \{\text{conjunctive, disjunctive}\})\} – are generated; and phase 2 generates the interesting DAR rules \( A \Rightarrow C \) from \( \text{largeTerms}_{A(C)} \).

The parameters of the algorithm are: repository to be mined (dataset), user-specified minimum support and minimum confidence, allowed antecedents (\( \text{allowedAntecedents} \)) and allowed consequents (\( \text{allowedConsequents} \)). For instance, if the items of the repository are \{X, Y, Z, V, W\}, we could have \( \text{allowedAntecedents} \subseteq \{X, Y, Z\} \), and \( \text{allowedConsequents} \subseteq \{V, W\} \). The interpretation must be: only \( X \) or \( Y \) or \( Z \) (\( V \) or \( W \)) would appear in the antecedent (consequent) of a DAR rule. This restriction allows the algorithm to reduce the search space of DAR rules in order to alleviate to potential explosion of rules, as discussed previously. By default, any item of the repository can appear either in the antecedent or in the consequent.

Concerning phase 1, recall that each \( \text{largeTerm}_A(C) \) in \( \text{largeTerms}_{A(C)} \) can be a conjunctive term or a disjunctive term. For each term, the function \( \text{generateTerms()} \) of DAR algorithm has different generation optimization strategies.

For conjunctive terms, we adopt the classic optimization rule used in Apriori algorithm [2]: a term is recursively large if all of its sub terms are large – bottom-up strategy of construction of large terms. On the other hand, for disjunctive terms, the optimization strategy is inverse, or top-down: for example, only if the disjunctive term \( X \text{ OR Y OR Z} \) is large it is possible that some of the \( C_{3,2} \) derived disjunctive terms of size 2 are also large, in a recursive way – for the foundations, see Chapter 3.1.

Other two optimizations are obvious, AND(OR)-shortcut: if at least an item of a con-
DAR (DataSet dataSet, AttributeList allowedAntecedents,
    AttributeList allowedConsequents, integer maxAntecedentSize,
    integer maxConsequentSize, real minimum_support, real minimum_confidence)
{
    // Phase 1: Generation of the Large Terms
    // Terms of allowed antecedents
    ItemSetList largeTermsA = generateTerms(dataset, allowedAntecedents,
        maxAntecedentSize, minimum_support);
    // Terms of allowed consequents
    ItemSetList largeTermsC = generateTerms(dataset, allowedConsequents,
        maxConsequentSize, minimum_support);

    // Phase 2: Generation of the interesting DAR Rules
    RuleList rules = Ø;
    for all ItemSet A in largeTermsA {
        for all ItemSet C in largeTermsC {
            Rule rule = A ⇒ C;
            real support = calculateSupport(rule, dataset);
            if (support >= minimum_support) {
                real confidence = support / A.getSupport();
                if (confidence >= minimum_confidence) And interesting(rule)
                    {rules = rules ∪ {rule}};
            }
        }
    }
    return rules
}

Figure 4.1: Algorithm DAR
junctive (disjunctive) term is false (true) then the conjunctive (disjunctive) term is false (true).

The large conjunctive and disjunctive terms generated in phase 1 of DAR algorithm are input of phase 2. Function `calculateSupport()` returns the support of a rule $A \Rightarrow C$, where $A(C) \in largeTermsA(C)$, in an optimization: for each transaction, instead of checking the presence of $A$ and $C$ in a random order, the check is performed counting the item set $(A$ or $C)$ with least support first, so both item sets will be counted only when this item set appears. Once this item set appears in less transactions, no needless counting is going to be performed.

At this time, function `interesting()` guarantees that the induced rules are in accordance with the definition of interesting rule in Chapter 3.1. Notice that the possible sets of large terms, close to and increases (see Figure 3.1), are embedded into sets `largeTermsA` and `largeTermsC` (Figure 4.1) to find interesting rules.

Regarding the complexity of DAR algorithm, the problem is the generation of the large terms (see Chapter 2.1) in phase 1. But phase 2 also has a reasonable complexity, due to functions `calculateSupport()` and `interesting()`, mainly the first, that still has to scan the dataset, even though in an optimized way. But the tests of the algorithm have revealed that the performance of DAR is quite satisfactory (see the next chapter), or not a lot higher than the principal association rule algorithms [6] that only work with conjunctive terms.
Chapter 5

Experimental Evaluation

To assess the performance of DAR algorithm, we performed several experiments on a Toshiba Intel Centrino Duo (1.50 GHz each), 2GB of main memory, and running Ubuntu 8.04 (Linux Kernel 2.6.24-16). The data resided in the Ubuntu file system and was stored on a hard disk with 160GB and 7,200 rpm.

5.1 Generation of Synthetic Data

We generated synthetic transactions to evaluate the performance of DAR over a large range of data characteristics. These transactions simulate the transactions of a software repository. Our model of the “real” world is that software programmers tend to commit sets of classes together, from a universe of software classes. Suppose a program with classes $A$, $B$, $C$, $D$ and $E$. Some programmers might commit only $A$ and $B$, and others only $A$. Transaction sizes are typically clustered around a mean and few transactions have many classes.

To create a dataset, our synthetic data generation program takes the parameters shown in Table 5.1.

The transaction sizes are picked from a Weibull distribution\[7\] with mean – equal to the estimated average size of the transactions – obtained from the experience with several software systems [5].

5.2 Test Plan

The tests consisted essentially of analyzing the behavior of the algorithm regarding the execution time. For that we varied the number of transactions, the number of classes and the value of the minimum support in this order. In the graphs presented below, each point (number of transactions (number of classes) (support), execution time) represents the average of 10 executions, with confidence interval of 99%.

\[1\]Preliminary studies [5] attest the similarity of repositories of software changes with the distribution.
Table 5.1: Parameters

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset #1</td>
<td>Number of transactions: [1,000..10,000] step = 1,000</td>
</tr>
<tr>
<td></td>
<td>Number of classes: 20</td>
</tr>
<tr>
<td></td>
<td>$Support_{min} = 0.05$</td>
</tr>
<tr>
<td>Dataset #2</td>
<td>Number of transactions: 2,000</td>
</tr>
<tr>
<td></td>
<td>Number of classes: [10..32] step = 2</td>
</tr>
<tr>
<td></td>
<td>$Support_{min} = 0.05$</td>
</tr>
<tr>
<td>Dataset #3</td>
<td>Number of transactions: 5,000</td>
</tr>
<tr>
<td></td>
<td>Number of classes: 20</td>
</tr>
<tr>
<td></td>
<td>$Support_{min} = [0.01,0.05..0.45]$ step = 0.05</td>
</tr>
</tbody>
</table>

For each dataset in Table 5.1, the antecedents’ space of the rules to be induced is the first half of the universe of classes (or the first half of the columns classes, counting from left to right), while the second half is the consequents’ space. Another realistic assumption, considering the impact analysis problem presented in the introduction, is that in an induced rule only one class should appear in the consequent [5].

For instance, for Dataset #1 the antecedents must be picked from the first 10 columns, while each individual consequent must be picked from the last 10 columns.

5.3 Results

Figures 5.1, 5.2 and 5.3 show the execution times of the DAR algorithm for Datasets #1, #2 and #3 given in Table 5.1, respectively. For Dataset #1 graph is Execution Time X Transaction. For Dataset #2 graph is Execution Time X Class. For Dataset #3 graph is Execution Time X Support. The scales of all graphs are linear, except the scale of Execution Time on graph from Figure 5.2, which is logarithmic. Also in all graphs, the points flanked by bars indicate the average time for 10 executions and the confidence interval of 99%, respectively.

Figure 5.1 shows 10,000 transactions, a universe of 20 classes and minimum support of 0.05. Notice that the execution time of DAR increases linearly. This is an excellent outcome, consequence of the optimizations done in the algorithm for the generation of the large terms ($largeTerms_{A(C)}$, in Chapter 4).

Figure 5.2 clearly suggests that the number of classes is a much more critical factor for the performance of DAR than the number of transactions, in spite of the optimizations. More precisely, the execution time tends to grow exponentially, although for at least 32 classes (and with 2,000 transactions and support 0.05) the time is more than reasonable. The cost is essentially due to the generation of large terms with more than 20 classes. It is comparatively the same problem of generation of large item sets of all induction of association.
rules algorithms, previously discussed in Chapter 2.1.

As expected, the support has a strong influence in the performance of DAR, similar to all association rules algorithms (Chapter 2.1). Figure 5.3 shows that for 5,000 transactions and 20 classes, the support tends to grow exponentially as it approaches zero. That is because for very low support values the number of large terms increases exponentially, and consequently the number of induced disjunctive association rules also increases exponentially.

5.4 Discussion

It is always good to insist: the task of DAR algorithm is much more challenging than the one addressed by Apriori algorithm, and many other algorithms of association rules mining, because disjunctive association rules reach minimum support much easier than the ubiquitous conjunctive association rules. In spite of that, the performance of DAR is comparatively close of those of the other algorithms, as the tests have shown.

Figure 5.1: Execution Time, along the number of transactions
Figure 5.2: Execution Time, along the number of classes

Figure 5.3: Execution Time, increasing the support
Chapter 6

Conclusions and Future Work

We presented a new algorithm, DAR, for inducing all significant disjunctive association rules in a database of transactions.

We point out the difference of DAR algorithm for other association rules algorithms. Although elaborated extensions of the classic model of (conjunctive) association rules have succeeded, all the concerning algorithms have a point in common: they induce only conjunctive rules. A much more challenging problem is the induction of disjunctive association rules, because they reach minimum support much easier than conjunctive association rules. We showed that the introduction of the concept of interesting rules avoids the explosion of rules, among those with minimum support.

The experimental results showed that the proposed algorithm performs well under certain restrictions as the limitation of the search spaces of the antecedent and consequent terms. In the experiments with DAR we obtained performance results equivalent to results from other conjunctive association rules.

In at least one type of practical application, the algorithm showed that it is even indispensable: it is the problem of software change impact analysis.

In the future, we plan to extend this work along the following dimensions: (1) extend the model of disjunctive association rules to be more flexible, with the introduction of hybrid terms, i.e., mixing conjunction and disjunction; (2) perform more optimization of the algorithm, an imperative of a model that tends to become more flexible; and (3) apply it in other problem domains: we believe that disjunctive association rules deserves attention, mainly because a database changes with the time (mining over time).
Bibliography


