An Induction-based Compaction of Sets of Association Rules among Web Concepts

Mauricio Minuto Espil
ARBA, Provincia de Buenos Aires, Argentina

Juan M. Ale
Universidad Austral, Argentina

Abstract

Association rules are nowadays regarded as robust vehicles for creating Web recommendations. We present an induction-based technique for creating a compact representation of sets of association rules, particularly intended for publishing the compact representation in the Web. Our technique efficiently induces a defeasible logic program from a set of association rules, in a manner that the complete set of the given associations can be concluded, when integrating the induced compact program within a defeasible logic reasoning framework.

1. Introduction

E-business and e-commerce portals are mainly intended for companies that offer the service of on-line transactions to clients. Since web-services become available to the outside of owner boundaries, many companies have started the encapsulation of their facilities in the form of web services. Companies are progressively adopting public protocols of wide-spread use in order to allow machine-driven transactions, in addition to the human-driven transactions available through their portals. Since business and commerce imply advertising, it has become evident that some form of "semantic" advertising added to service description is also needed for potential "machine" clients to asses the potential of the service provided.

B2B is an area where the knowledge of the state and evolution of the market is central. Association rules have been recently considered as robust vehicles for producing recommendations [27], and mining algorithms has been developed specially for the issue [20]. In B2B, publishing association rules on the history of the transactions of some company may result crucial for gaining markets. That is, *association rules may become data* potentially appreciated in web-service based B2B.

Besides, the reader must notice that an association rule from an itemset I_1 to an itemset I_2 implies the existence of a logical relation among two *conceptual classes*: the class of transactions that involve itemset I_1 and the class of transactions that involve both itemsets I_1 and I_2. More specifically, it *implies* a *subsumption* relation, in the sense that the extension of the last class is contained in the extension of the first one, with extra quantitative information provided on the degree of participation of the subsumed class into the more general one (confidence) and into the world of analysis (support). As subsumption relations thence, association rules can be easily encoded in OWL, RDF and RDFS [32, 33, 34]). Moreover, the implication is important regarding the fact that the assembling and filtering of association rules through the use of ontologies has shown concrete gain in terms of informative power

```
<owl:Class rdf:ID="I1"/>
<owl:Class rdf:ID="I2"/>
<owl:Class rdf:ID="I1andI2">
    <owl:intersectionOf >
        <owl:Class rdf:resource="#I1"/>
        <owl:Class rdf:resource="#I2" />
    </owl:intersectionOf>
</owl:Class>
<rdf:Description rdf:about="#I1andI2">
    <rdfs:subclassOf rdf:resource="#I1"
                     def-ar:support="5.100"
                     def-ar:confidence="93.100"/>
</rdf:Description>
```

[23]. It turns out therefore extremely desirable to incorporate association rules to web service description documents.

Example 1: The association rule from class I1 to class I2 can be encoded as follows:

In the fragment above, only a new namespace has been added (*def-ar*), and two attributes are added to the property rdfs:subclassOf: *def-ar:support*, informing the observed support threshold, and *def-ar:confidence*, informing the observed confidence threshold.

1.1. Recommendation and Association Rules Mining

Algorithms for association discovery that scale well on large amounts of transaction data have been developed and are well-known, as A-Priori [1], DIC [9], FPgrowth [15], and more recently [5]. Nevertheless, those algorithms are not particularly intended to be used for creating recommendation in machine-oriented transactions. The algorithms available for discovering association rules were devised with a purpose in mind: *analysis*. According to the models available for defining analytical rule *interest* – statistically or empirically – [16, 18], *interesting* association rules are those that exhibit certain *signifying* measures (called here generically *parameters*), such as *high confidence* and *good support*. The actual discovered values of parameters are thence of extreme importance for analysts. Analysts, however, do not need to attend to *all* the associations that are present in the data, since some of the mined rules may show no analytical importance. The analysts would rather provide *thresholds* to parameter values to filter the potential result according to interest criteria, and normally, the thresholds provided are high and the number of rules obtained small. Thresholds thus serve as pruning devices for analysis.

Recommendation is a somewhat different matter. Informing clients of the applied thresholds instead of the actual parameter values of each rule seems sufficient for advertising. Moreover, although the number of exhibited rules should not be very large (as it was the case of analysis) - low time response in web interactions is central -, filtering through analytical signifying thresholds does not seem desirable in this context. It could imply a drastic reduction in recommendations that clients would appreciate. Other schemes of reduction are thus needed. Moreover, whenever possible, all rules should be shown, albeit in compact form.

1.2. Rule Set Compaction

The compaction problem has received attention from the data mining community since the very beginning [6, 21]. Several pre, in and post-processing reduction techniques have been proposed [7, 8, 11, 23, 30]. In those approaches; the rules exhibited are restricted only to those that match given patterns; non-matching rules are not shown. Because no reasoning framework is provided, the user cannot deduce all the pruned rules from the exhibited set.

From a different perspective, different notions of closures and minimal covers have also been employed for reducing the number of rules to show [10, 24, 25, 31]. Those approaches, based on formal concept analysis, prune *redundant* rules and allow the inference of all pruned rules. Nonetheless, they fail in discovering patterns characterizing the singularities present within each particular dataset when considering *redundancy*; the patterns employed and the deduction mechanism (closure computation) are fixed in the algorithm code.

More general approaches have also been attempted. A series of induction mechanisms were introduced in [12, 17, 28], showing techniques for the induction of "queries" on itemsets that mine associations from frequent itemsets; the queries acting as a compact representation of the rules. Those approaches, however, fail in real compaction. The presence of *all* itemsets with their respective frequencies is necessary for reconstructing the rules from the induced queries.

In this paper, we present a different approach. Since the relation among defeasible logic [3] and semantic web applications is closer, from the descriptive side [26], and from the normative side (particularly when e-commerce is involved [13]), we believe that the addition of association rules to those settings is worth exploring. This is the subject of this work.

We present therefore an algorithm that produces a compact representation of the given set of association rules through a *defeasible* logic program [3, 4], a triplet formed upon:

a) a set of *associations*, atomic formulae formed upon association predicates on pairs of terms denoting itemset classes, which represent, semantically, association rules that are *present* in the given set;

b) a set of *assumptions*, Horn clauses of associations with itemset variables, that represent particular *inference rules* that characterise the given set;

c) a set of *defeaters*, counter-arguments to associations that can be wrongly implied from associations and assumptions, and semantically represent a set of association rules that are *not present* in the given set.

A linear-time framework for non-monotonically reasoning with programs is defined, in a manner that the set of all derived ground instances (associations) can be computed, and a *PTIME induction algorithm* is then presented for inducing a compaction from the given set of associations, used here as "positive examples", in a machine-learning terminology. All and only all the given associations can be inferred from the induced program; the program showing therefore a *compaction* principle, in the sense that assumptions *entail* implicit associations.

Our approach is closer to the spirit of [14, 19]. The difference relies in scope. While the cited works have the identification of legal (meta-)defeasible rules for reasoning on legal argumentation as a goal, our approach only aims at producing a compaction. The difference is significant, because in the cited

approaches an association is regarded as a defeasible sentence, thus turning the approach more oriented to discover nested defeasible rules [29]. Moreover, an optimal is searched there, thus leading to intra-ctability and the consequent use of heuristics, in that case on the legal domain which is not ours.

1.2. Compaction by Induction

Example 2: Let us have the association rules shown in Table 1, mined from a real set provided by the branch of a major banking institution, with ≥ 0.6 as threshold for confidence and ≥ 0.05 as threshold for support.

1-A⇒B	11-BC⇒G	21-G⇒BC
2-A⇒C	12-BG⇒C	22-H⇒C
3-A⇒I	13-C⇒A	23-H⇒I
4-A⇒CI	14-C⇒I	24-H⇒CI
5-AB⇒H	15-CG⇒B	25-I⇒A
6-AC⇒I	16-CI⇒A	26-I⇒C
7-AI⇒C	17-CI⇒H	27-I⇒AC
8-B⇒C	18-CH⇒I	28-I⇒H
9-B⇒G	19-G⇒B	29-I⇒CH
10-B⇒CG	20-G⇒C	30-IH⇒C

Table 1 Set Example

We observe a frequent pattern to hold on the rule-set that tells us the following: if this is the case that the antecedent (the left side of the association) of some rule *r* (the pattern body to match) is the union of two disjoint itemsets i_1 and i_2, then it is likely to find within the set a rule *r'* (the pattern head to conclude) with its antecedent equal to one of the itemsets i_1 or i_2 and the consequent (the right side of the association) equal to the consequent of rule *r*. We notice that 12 rules in the set can be concluded from the pattern head once the pattern body has matched another rule in the set, and also notice 5 counter-examples. Henceforth the pattern can be used here for safely pruning the rules concluded from the head; provided a deductive mechanism exists that allows the pruned rules to be deduced through the application of the pattern to the appropriate remaining rules, with account of the discovered counter-examples. On this basis, we can safely prune, in the order that follows, rules 3(6), 14(6), 2(7), 26(7), 9(11), 8(12), 20(12), 19(15), 13(16), 25(16), 28(17), 23(18) (the rule number in parenthesis denoting the rule matching the pattern body). On the other hand, we notice that rules: A⇒H, B⇒H, C⇒G, C⇒B and C⇒H are not members of the set of rules and are although deduced from the pattern. The reconstruction mechanism must account for them as counter-examples, in order to avoid deduction inaccuracies.

According to Example 2, if we could induce, with an appropriate induction mechanism, the meta-rule: "*For any* 3^{ary}*-tuple of itemsets* X, Y, *and* Z, *whenever a rule from* X union Y (X and Y disjoint) to Z *holds with a confidence* ≥ 0.6 *and a support* ≥ 0.05, *conclude that a rule from* X *to* Z *also holds with the same confidence and support thresholds*" we could safely prune rules from the set of rules, with no information loss; the pruned rules could always be *inferred*, in the classical sense, from the meta-rule and the associations remaining in the pruned set. Since rules A⇒H, B⇒H, C⇒G, C⇒B and C⇒H would be also classically – and wrongly – inferred, our induction mechanism should produce counter-arguments of the form "*do not conclude rule r with a confidence* ≥ 0.6 *and a support* ≥ 0.05", for each wrongly inferable rule *r*, in order to defeat its classical derivation. The inference mechanism needed should thus produce defeasible conclusions; they must be abandoned whenever a stronger counterargument is present. In addition, the example shows that, if we count non-pruned rules, meta-rules and defeaters as plain rules, the information presented to the client is smaller in number than the whole set of the given associations. We have just produced a *compaction* of the set.

Example 3: The meta-rule encountered in Example 2 can be encoded in an rdf-style, and added to a descriptive document as follows:

```
<def-ar:metaClass rdf:ID="X"/>
<def-ar:metaClass rdf:ID="Y"/>
< def-ar:metaClass rdf:ID="Z">
< def-ar:metaClass rdf:ID="XandY">
    < def-ar:intersectionOf >
        < def-ar:metaClass rdf:resource="#X"/>
        < def-ar:metaClass rdf:resource="#Y"/>
    </def-ar:intersectionOf>
</def-ar:metaClass>
< def-ar:metaRule  def-ar:support="5.100"
                   def-ar:confidence="60.100">
   < def-ar:antecedentRule>
      <rdf:Description rdf:about="#XandY">
          <def-ar:subclassOf rdf:resource="#Z" >
      </rdf:Description>
   </def-ar:antecedentRule>
   < def-ar:consequentRule>
      <rdf:Description rdf:about="#X">
          <def-ar:subclassOf rdf:resource="#Z" >
      </rdf:Description>
   </def-ar:consequentRule>
< def-ar:metaRule>
```

Example 4: The first counter-example can also be added as follows:

```
<owl:Class rdf:ID="A"/>
<owl:Class rdf:ID="H"/>
<def-ar:Defeats  def-ar:support="5.100"
                 def-ar:confidence="60.100">
    <rdf:Description rdf:about="#H">
        <def-ar:subclassOf rdf:resource="#A"/ >
    </rdf:Description>
</def-ar:Defeats>
```

Meta-classes in Example 3 are class variables, names denoting classes generically. Intersection of meta-

classes corresponds to intersection among the classes substituting the variables. The counter-example in Example 4 is encoded as *defeating* a potential subsumption with appropriate values for support and confidence. It is rather a general claim than a particular one.

The fragment shown in Example 3 together with all counter-examples found on Example 2 encoded as in Example 4, plus the non-pruned rules from Example 2 coded as shown in Example 1 constitute a document that results a compaction of the set of all rules exhibited in Table 1, providing that there exists a closure notion from a reasoning device capable of reconstruct the entire given set if needed.

If the induction mechanism is sufficiently aware in detecting non-straightforward meta-rules, as the meta-rule encoded in Example 3, the pruning mechanism could be applied as a complement of the reduction mechanism based on cover computation defined in [24, 25], and the reduction mechanism of redundant rules in the sense of [31], producing more reduction. Our reduction mechanism is able to identify general inference rules (as those of [24, 25, 31]) and prune all general redundant rules in consequence, and may also identify patterns present only within the given particular set, as the meta-rule identified in Example 3, not considered in any of the reduction schemes from [24, 25, 31], showing therefore a stronger compaction power.

1.4. Paper Organization

The rest of the paper proceeds as follows. In section 2, we present formally the logic for reasoning on association rules. In Section 3, we present an algorithm for inducing a program - in the presented logic - from a set of association rules mined from data. In Section 4, we show experimental results that assess the effectiveness of our framework with respect to the compaction goal. In Section 5, we discuss our approach from an implementation perspective, and in Section 6 we conclude.

2. Formal Framework

We have explained in the introduction that our approach relies on inducing a theory in some logic of formulae with an *interpretation* on association rules. For a formal definition of the semantics of association rules, the reader is referred to [1].

A family of non-monotonic logic formalisms for defeasible reasoning on incomplete knowledge with a well defined sceptical reasoning process has been defined [3]. A defeasible logic theory is a collection of rules, formed upon a set of atoms as a body and an atom as a head, that allows the reasoning on sets of given facts. In defeasible logic, the rules constituting a theory represent assertions whose truth is indisputable, and assertions whose truth is problematic. As a consequence, two sorts of conclusions are obtained from the reasoning process: indisputable or defeasible.

More formally, a defeasible logic theory is composed of a set of strict rules (rules that are indisputably true), defeasible rules (rules whose application is considered problematic), defeaters (counter-arguments to defeasible conclusions), and a superiority relation among rules (as a disambiguation mechanism).

It was shown that the problem of deciding if an atom is a member of the extension of a defeasible theory can be efficiently implemented since it demands linear time and space [22]. Besides, it has been shown that the absence of a superiority relation does not compromise the expressive power of defeasible logic [4]. Within our approach, thus, we are interested in defeasible rules and defeaters only, and, since our targets for reasoning are association rules, we incorporate a notion of *threshold covering* to the reasoning process; if an association rule is concluded with some threshold values for support and confidence, the same association is concluded for any smaller value down to 0, provided there is no defeater for the rule with a value in-between. In the example above, the association rule 3 ($A \Rightarrow I$) is concluded upon the association rule 6 ($AC \Rightarrow I$) with ≥ 0.6 as confidence and ≥ 0.05 as support, according to the "defeasible rule" pattern encountered in the example. Thus, $A \Rightarrow I$ is also (implicitly) concluded with ≥ 0.5 as confidence and ≥ 0.03 as support. However, if a defeater for rule $A \Rightarrow I$ is simultaneously asserted with ≥ 0.04 the $A \Rightarrow I$ would not be concluded with ≥ 0.5 as confidence and ≥ 0.03 as support.

This choice is important for better understanding of the theories obtained. Within our approach, we consider defeasible rules that allow us to conclude that an association rule defeasibly holds, with independence of the conformance with given support and confidence thresholds, provided that other association rules also hold *conforming* the thresholds. Defeaters are included here to prevent the erroneous conclusion of an association *not conforming* the given thresholds.

2.1. Logic for Associations

We want to represent the set of all given association rules among itemsets through a defeasible theory. Thus, the domain on which formulae in our logic are built is founded structurally on the set of all itemsets formed upon the set of items involved, with

exception of the null itemset. This way, *terms* in our logic (constants and variables) represent itemsets with a certain number of items.

Definition 1 (Itemset Term): An *itemset term* is a construct of any of the forms:

- $i_1 \ldots i_n$, *a ground* itemset term, where $i_1 \ldots i_n$ is a non-empty list of items.

- $V_{m,M}$, *a variable* itemset term, where $V_{m,M}$ is an itemset variable, $0 \leq m \leq M$. The pair m, M indicates the class of itemsets involved – with size between m and M –.When the pair is absent, the pair $(0, \omega)$ *is assumed*.

- $t_1 \cup \ldots \cup t_n$, a itemset *union* term, where $t_1, \ldots, t_m$ is a non-empty list of itemset terms, and $\cup$ is an itemset infix function name with set union as fixed interpretation. $t_1 \cup \ldots \cup t_n$ implies that all t_i, i=1..n, are mutually disjoint.

Definition 2 (Association): Within our logic, an *association a* is an atom of the form:

$$a\text{: } \mathbf{S} \Rightarrow_{(\sigma,\delta.)} \mathbf{T},$$

where (_ $\Rightarrow_{(\sigma,\delta.)}$ _) is an *association predicate* on two itemset terms (that fill the _ positions): **S,** or alternatively Ant(*a*), the *antecedent* of *a*, and **T,** or alternatively Cons(*a*), the *consequent* of *a*.

Association predicates are parametric. The pair (σ,δ), which is a part of the predicate signature, is a pair of *parameters*: σ, the *support threshold*, and δ, the *confidence threshold*. Both parameters must be rational numbers. This way, there would be as many association predicates (countable infinite) as pairs of (σ,δ) of parameters could be formed in the logic. An association $\mathbf{S} \Rightarrow_{(\sigma,\delta.)} \mathbf{T}$ in our logic always implies that the atom $\mathbf{S} \cap \mathbf{T} = \varnothing$ holds.

Finally, we call a *schema* an association with at least one itemset variable.

Definition 3 (Assumption): An *assumption* α is a clause of the form:

$$\alpha\text{: } B(\alpha) \vdash H(\alpha) \quad \text{where:}$$

- $B(\alpha)$ (the *body* of assumption α) is a non-empty list of association schemas with no arithmetic operators used in thresholds.
- $H(\alpha)$ (the *head* of the assumption α) is a non empty list of association schema, such that every variable appearing in $H(\alpha)$ also appears in $B(\alpha)$.

An assumption is *head-relevant* if each atom in the body shares at least one non-ground term with the head or has a ground itemset term with a non-empty intersection with a ground itemset term in the head.

Example 5: The following assumption α_1 corresponds to the meta-rule induced in Example 2:

$$\alpha_1\text{: } X_{1,1} \cup Y_{1,1} \Rightarrow_{(\mathbf{0.05,\ 0.6})} Z_{1,1} \vdash X_{1,1} \Rightarrow_{(\mathbf{0.05,\ 0.6})} Z_{1,1}$$

where $X_{1,1}$, $Y_{1,1}$and $Z_{1,1}$are itemset variables of item-size = 1.

Assumption α_1 is head-relevant.

Definition 4 (Defeater): A *defeater d* is a construct of the form:

$$d\text{: } \mathbf{X} \not\Rightarrow_{(\mathbf{s,\ p})} \mathbf{Y}$$

where $\mathbf{X} \Rightarrow_{(\mathbf{s,\ p})} \mathbf{Y}$ is an association. A defeater asserts that the association $\mathbf{X} \Rightarrow_{(\mathbf{s,\ p})} \mathbf{Y}$ cannot be concluded in the logic. In the context of a proof system, a defeater has priority over conclusions obtained from the application of assumptions.

Example 6: The defeater d_1: $\mathbf{A} \not\Rightarrow_{(\mathbf{0.05,0.6})} \mathbf{H}$ corresponds to the first counter-argument introduced in relation with Example 2.

2.2 The Reasoning Framework

In order to reason appropriately with programs made of associations, assumptions and defeaters, a non-monotone inference mechanism is presented, and theories are defined on it. Programs in this framework are inspired from [3], and can be translated in linear-time on the number of their ground instances into definite programs of clausal logic [4], with a linear-time ground inference procedure [15] on the same basis.

Definition 5 (Compaction Program): A *compaction program* ρ is a 3^{ary}-tuple (*AR*, *Das*, *Dft*), where *AR* is a set of associations, *Das* is a set of assumptions, and *Dft is* a set of defeaters.

Definition 6 (Closure): We say that an association *a*: $\mathbf{S} \Rightarrow_{(\sigma,\delta.)} \mathbf{T}$ is *derivable from* a compaction program ρ : (*AR*, *Das*, *Dft*), if and only if there exists a sequence of ground associations π, recursively satisfying the following:

For all $0 \leq i \leq k - 1$
  if $\pi[i+1] = \mathbf{X} \Rightarrow_{(\sigma,\delta.)} \mathbf{Y}$ **then**
    1) $\mathbf{X} \Rightarrow_{(\sigma',\delta'.)} \mathbf{Y} \in AR$, for some $\sigma' \geq \sigma$, $\delta' \geq \delta$
      and $\nexists\ \mathbf{X} \not\Rightarrow_{(s,p)} \mathbf{Y} \in Dft$,
        for any p, s | $1 - s \leq \sigma$, $1 - p \leq \delta$; **or**
    2) $\exists$ some ground instance β of $\alpha \in Das$
        . | $H(\alpha) = \mathbf{X} \Rightarrow_{(\sigma',\delta')} \mathbf{Y}$,
          for some $\sigma' \geq \sigma$, $\delta' \geq \delta$
      and $\nexists\ \mathbf{X} \not\Rightarrow_{(s,p)} \mathbf{Y} \in Dft$,
        for any p, s | $1 - s \leq \sigma$, $1 - p \leq \delta$
      and for each $\mathbf{V} \Rightarrow_{(\sigma'',\delta''')} \mathbf{W}$ in $B(\alpha)$,
        $\exists j,\ 1 \leq j \leq i$, $\pi[j] = \mathbf{V} \Rightarrow_{(\sigma'',\delta''')} \mathbf{W}$,
        for some $\sigma' \geq \sigma''$, $\delta' \geq \delta''$. (1)

and an index $k \geq 1$, such that $\pi[k] = \mathbf{S} \Rightarrow_{(\sigma,\delta.)} \mathbf{T}$. A *closure Cl*($\rho$) for a compaction program ρ : (*AR*, *Das*, *Dft*) is a set AR_+, where AR_+ is the set of all ground associations derivable from program ρ.

For $1 \le i \le k$
  Derive forests from all head-relevant ground assumptions with body size not > k, forming a dependency graph from body rules into head rules of each assumption:
  Find all clases of isomorphic forests *generalising isomorphic forests into classes of candidate assumptions Das*, generating a fresh variable per leaf in each forest class and a substitution per leaf in each instance of the forest class;
Loop
  Search for *a set Dftmin of* ***defeaters*** *for assumptions* ***in*** *Das*, attaching all substitutions and candidate – *conflicting* – assumptions used for inferring each defeater;
Loop
  adjusting the classes by variable sizes and confidence and support, reducing the number of defeaters in *Dftmin*;
Choose a maximal elimination order for the rule dependency graph;
Prune rules in the order produced;
If the compaction criterion is fulfilled
  exit the algorithm returning *Das*, *ARmin* and *Dftmin*;
If *there is no conflicting assumptions*
  exit the algorithm returning failure*;*
Choose a conflicting assumption to prune from and delete it from *Das*;

Fig. 1: Induction Algorithm

3. Inductive Defeasible Compaction

In this section we present the main result of our paper: we present the notion of *inductive defeasible compaction* of a set of association rules and an algorithm for finding such compaction of a given set of *discovered* associations. The input is assumed a *complete* set of associations, with maximum values for confidence and support thresholds; no holding association rule should miss to be interpreted by an association atom in the input set, and, for all association atom in the input set, no association rule holds with support and confidence greater that the thresholds given in the atom.

Definition 7 (Inductive Defeasible Compaction): Given a constant k > 0 and a set of ground associations AR, complete for a given database **D**, a tuple $(\sigma, \delta,...)$ of parameters, an *inductive defeasible compaction* of the set *AR* is a program ρ: (AR_{min}, *Das*, Dft_{min}), with a set *Das* of head-relevant assumptions, with no more than k atoms in the bodies, that satisfies that:

i) $Cl(\rho) = AR$;
ii) $\#AR_{min} + \#Dft_{min} + \#\{\text{atoms}(\alpha) \mid \alpha \in Das\} < \#AR$; *and*
iii) *there not exists a program* ρ': (AR', *Das*, *Dft'*) *such that* $Cl(\rho') = AR$, *and* $AR' \subset AR$ *or* $Dft' \subset Dft$. (2)

3.1 The Induction Algorithm.

A *PTIME* algorithm that computes a compaction of a complete set of ground associations by inducing a set *Das*, and producing appropriate sets AR_{min} and Dft_{min} for the induced set *Das* is presented in Figure 1. We discuss here the underlying ideas, and details related with its correctness and time complexity.

The algorithm begins with a procedure, detailed in Figure 2, that greedily tries to produce all ground head-relevant assumptions, increasing the possible body size, variable *j* in the algorithm, from 1 to k. Each association in *AR* is considered there the head of a potential ground assumption, and all groups with a body size that equals *j* are considered as potentially bodies, provided the union of the itemsets that appear in the antecedent and the consequent of all members of the group covers the union of the itemsets of the antecedent and the consequent of the selected head rule.

Form a graph *G* with one vertex for each rule *a* in *AR* and
  an edge (*a*, *b*) for each pair of vertices *a* and *b* of *G*, such that itemsets(*a*) intersects itemsets(*b*).
Form an ordered list *L* of all items in vertices of *G*;
For each vertex *a* of *G*
  Create a pair of indices pointing to the first and the last items of vertex *a* in list *L*;
For j=1..k
  For each vertex *a* of *G*
    For each group *g* of j vertices of *G* adjacent to *a*
      s.t. there exist a sequence of vertices $b_1,\ldots,b_j$
        s.t. last(b_i, *L*) ≥ first(b_{i+1}, *L*), for $1 \le i \le j-1$, and
          first(b_1, *L*) ≤ first(*a*, *L*) and
          last(*a*, *L*) ≤ last(*bj*, *L*)
        Form a disjoint partition *P*(*a*, *g*, j) of the union of all itemsets in *a* and all itemsets in each rule *b* in *g*
        Form a forest *head*(*a*, g, j) with two trees,
          headAnt(*a*, *g*, j) and *headCons*(*a*, *g*, j),
          with itemsets Ant(*a*) and Cons(*a*) as roots
          and each subset of itemsets Ant(*a*) and Cons(*a*)
            in *P*(*a*, *g*, j) as their respective sons
        For each vertex *b* in *g*,
        form j forests *body*(*a*, *g*, i), 1≤ i ≤j, with two trees,
          s.t. *bodyAnt*(*a*, *g*, i) and *bodyCons*(*a*, *g*, i),
          with itemsets Ant(*b*) and Cons(*b*) as roots
          and each subset of itemsets Ant(*b*) and Cons(*b*)
            in *P*(*a*, *g*, j) as their respective sons
        Assign to each leaf *l* of trees *bodyAnt*(*a*, *g*, i) and
          bodyCons(*a*, *g*, i), 1≤ i ≤j,
          a fresh variable $V_{m,M}$, m, M = size(itemset(*l*)).
        Assign to each leaf *l* of tree *headAnt*(*a*, *g*, j)
          the variable assigned to itemset *l*
          in some leaf of some tree *bodyCons*(*a*, *g*, i),
          labelled before.

Fig. 2 Forests Derivation

Next, the procedure proceeds to build, for each ground assumption, a set of forests of trees of itemset terms, with 1) one forest for the head, with one tree for the antecedent and one tree for the consequent; and 2) one forest for each potential body, with a tree for the antecedent and consequent of each atom in the body. The leaves of the trees in the forests contain the subsets that are produced from the complete intersection of the prospective head of the

assumption and the prospective bodies, considering the antecedent and the consequent of each rule separately.

Then, fresh variables are assigned to leaves; the forest becoming a structural representation of an assumption, candidate for the set *Das*.

Note that the time complexity of the procedure detailed in Figure 3 is then $O(n^k)$, n the number of the given rules.

The next step consists in finding isomorphisms among the forests, with the linear time algorithm of [2], leaving one assumption per isomorphic class in the set *Das*; the step demanding $O(m^2)$ tests, m the number of forests. We preserve all ground instances of each class on a list attached to the class, in a manner that the substitutions applied to each variable can be deduced from them easily.

The algorithm proceeds next to find defeaters, the set Dft_{min}, from the set of assumptions – forest classes -, applying them greedily to the given associations, according to the condition in (1). The assumptions used in inferring an atom to be defeated are then attached to the defeater with the substitutions applied. The method employed works satisfying the property iii) in (2) for set Dft_{min} because a minimal set of defeaters can always be obtained using the assumptions and the *complete* given set *AR*; the defeaters inferred from *Das* is independent of the level of pruning applied to *AR*, provided no information loss occurs in that pruning. This step takes $O(nk)$, n the number of associations. An optimisation is attempted for set *Das*, with the reduction of the number of defeaters in Dft_{min} as a goal. The algorithm loops, trying to find the better set of assumptions, by successive adjustments. The substitutions applied to form ground assumptions are contrasted with the substitutions applied to form defeaters. Adjustments in variables' sizes – hitherto unlimited – and in thresholds of the assumptions are operated, producing if possible: 1) intervals of sizes of variables that exclude the size of variables of defeaters; 2) a maximum of confidence and support thresholds for bodies in the assumptions that prevent the formation of one or more defeaters. It is easy to see that the step is polynomial in the size of set *AR*.

The pruning of *AR* is then accomplished. A candidate for pruning results any association that appears as head of a ground instance of an assumption in *Das* (recall they were attached to assumptions in step one). Since the rule dependency graph may have cycles, cycles are identified and broken by eliminating one node of each cycle. An elimination order is therefore produced, by a systematic remove of ears in the graph, considering only nodes candidate for pruning. Associations are then pruned from *AR* in the elimination order, forming the set AR_{min}. Note that this technique ensures *properties i and iii to hold for set* AR_{min}. Finding cycles in a graph, the hard part of this step is known to have a polynomial time complexity in the number of nodes, so it is this step in the number of given associations, since the nodes in the graph are the associations themselves.

Finally, the test: $\#AR_{min} + \#Dft_{min} + \#\{\text{atoms}(\alpha) \mid \alpha \in Das\} < \#AR$ is made. If the answer is positive the algorithm ends successfully – the program returned *satisfying property ii* (in addition to *i* and *iii*). If the answer is negative, an assumption among those contributing in producing a defeater is chosen for elimination (the most employed when producing defeaters is chosen first), and the process loops finding defeaters for the new set *Das*. If there are no assumptions to prune, the algorithm ends with failure. This step has constant time, provided the sizes has been stored, and the number of loops, if they resulted necessary, cannot exceed $O(n_k)$ iterations.

4. Experimental Results

Our approach has been experimented on three different highly correlated transaction database cases: case 1: (PtC), case 2: (DSP) and case 3: (Arry), each from a different domain of e-commerce companies respectively, with a total of 2.9, 3.2 and 0.22 millions of records each, a number of 10502, 4135, and 1550 items.

Conf.	#rules	#pruned	#dftrs
	PtC		
0.5	6604	2985	1114
0.6	2697	2081	25
0.75	1867	1606	10
0.8	1266	1176	0
0.95	892	866	1
0.98	705	699	1
	DSP		
0.5	2473	1168	268
0.6	1696	869	64
0.75	1509	844	89
0.8	1290	1030	29
0.95	1032	889	15
0.98	759	723	1
	Arry		
0.5	770	492	82
0.6	520	353	60
0.75	472	327	39
0.8	408	287	22
0.95	361	255	25
0.98	314	243	30

Table 2: Experimental Data

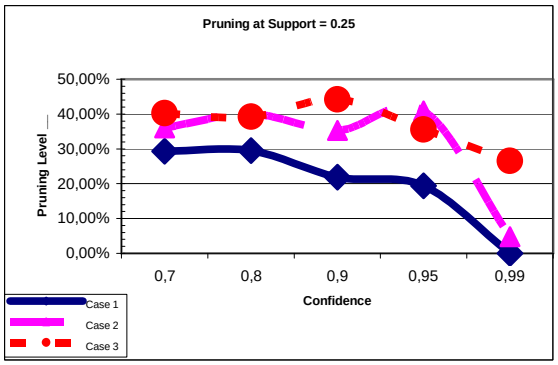

Fig. 3 Pruning experiences at support 0.25

The experiments were developed running the algorithm A-Priori on each of the sets, varying the support down from 0.25 to 0.1, and confidence down from 0.7 to 0.99. Our induction algorithm was then launched for each combination of thresholds. We show in Table 2 the results produced for k=3, support 0.5 and confidences d between 0.5 and 0.7, and the effectiveness of our method when applied to low confidences.

Finally, we note that our scheme eliminates all redundant rules in the sense of [25, 31], that is those association rules that are not in the covers. All the meta-rule deductive schemes implicitly included in [25] and [31] are induced by our method. The percentage of pruning, thus, outperforms [25], and is shown in Figure 3. Notice that the percentage of pruning achieved diminishes as the confidence is superior to 0.8. Nevertheless, the pruning is effective with confidence of 0.99 in the majority of cases. The pruning performed by algorithm A-priori is improved here.

5. Discussion and Challenges

It is important to discuss the technique presented here with focus on the purpose the technique pursues: to produce semantic recommendation.

The reader should have noticed that the algorithm presented relies strongly on "choice". For instance, the algorithm chooses ears in the graph to form an order for elimination, and the choice is arbitrary. This strategy is essential to maintain low complexity (polynomial), and to turn our approach feasible and practical. Nevertheless, a warned reader may conclude that this arbitrary choice implies that there are many compactions to produce and therefore the approach as a whole does not show to produce an optimal solution. And the reader is right in this conclusion. To complete the whole view, we describe how web service descriptions are complemented with the association rules as recommendations. In effect, under our scheme, the document describing the web service is augmented with a set of OWL/RDF/S triples that *only* incorporate the non-pruned rules with the format of Example 1, that is, the set AR_{min} of the compaction program obtained by our algorithm, together with the thresholds applied to the mining process and a registered URI of a registered description service. The assumptions and defeaters are not added to the web service description. If the associations encoded in the triples are not sufficient for the client (a search engine, for instance), the client may request a widening of the response to the description service identified by the given URI, and then the assumptions and defeaters are produced. The reasoning task to derive all the implicitly published rules is the client responsibility. Under this scheme, the rules that appear as members of the set AR_{min} is irrelevant, the only important issue is the size of this set. The developed scheme also supports an extension of the algorithm that admits the assignment of priorities to rules and to itemsets, in order to allow the user to produce a more controlled program as output. Nonetheless, the importance of the extension has not been already tested, and therefore it is beyond the subject of the present paper. It would be also interesting to design a scheme that supports queries where the client provides an itemset class and values for support and confidence and the engine produces a maximal class of inferred associated itemsets as a response. This scheme is under development, so we have not discussed this aspect here.

6. Conclusion

In this paper, we have presented a defeasible logic framework for managing associations that helps in reducing the number of rules found in a set of discovered associations. We have presented an induction algorithm for inducing programs in our logic, made of assumption schemas, a reduced set of association rules and a set of counter-arguments to conclusions called defeaters, guaranteeing that every pruned rule can be effectively inferred from the output. Our approach outperform those of [17], because all reduction compactions presented there can be expressed and induced in our framework, and several other patterns, particular to the given datasets, can also be found. In addition, since a set of definite clauses can be obtained from the induced programs, the knowledge obtained can be modularly inserted in a richer inference engine. Abduction can be also attempted, asking for justifications that explain the presence of certain association in the dataset.

The framework presented can be extended in several ways:

- Admitting defeaters to appear in the head of assumption, to define user interest.

- Admitting arithmetic expressions within assumeptions, for adjustment in pruning.
- Admitting set formation patterns as itemset constants.
- Extending the scope, to cover temporal association rules.

7. References

[1] R. Agrawal, and R. Srikant: Fast algorithms for mining association rules. In *Proc. Int'l Conf. Very Large Databases.* (1994).

[2] A. V. Aho, J. E. Hopcroft, J. Ullman. The design and analysis of computer algorithms, Addison-Wesley, 1974.

[3] G. Antoniou, D. Billington, G. Governatori, M. J. Maher, A. Rock: A Family of Defeasible Reasoning Logics and its Implementation. ECAI 2000: 459-463.

[4] G. Antoniou, D. Billington, G. Governatori, M. J. Maher: Representation results for defeasible logic. ACM Trans. Comput. Log. 2(2): 255-287 (2001).

[5] A. Basel, A. Mahafzah, M. Al-Badarneh: A new sampling technique for association rule mining, Journal of Information Science, Vol. 35, No. 3, 358-376 (2009).

[6] R. Bayardo and R. Agrawal: Mining the Most Interesting Rules. In *Proc. of the Fifth ACMSIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 145-154, (1999).

[7] R. Bayardo, R. Agrawal, and D. Gunopulos: Constraint-based Rule Mining in Large, Dense Databases. *Data Mining and Knowledge Discovery Journal*, Vol. 4, Numbers 2/3, 217-240. (2000).

[8] A. Berrado, G. Runger: Using metarules to organize and group discovered association rules. Data Mining and Knowledge Discovery. Vol 14, Issue 3. (2007).

[9] S. Brin, R. Motwani, J. Ullman, and S. Tsur: Dynamic itemset counting and implication rules for market basket analysis. In *Proc. ACM-SIGMOD Int'l Conf. Management of Data*. (1997).

[10] L. Cristofor and D.Simovici: Generating an nformative Cover for Association Rules. In *ICDM* 2002, Maebashi City, Japan. (2002).

[11] Y. Fu and J. Han: Meta-rule Guided Mining of association rules in relational databases. In *Proc. Int'l Workshop on Knowledge Discovery and Deductive and Object-Oriented Databases*. (1995).

[12] B. Goethals, E. Hoekx, J. Van den Bussche: Mining tree queries in a graph. *KDD*: 61-69. (2005).

[13] G. Governatori, D. H. Pham, S. Raboczi, A. Newman and S. Takur: On Extending RuleML for Modal Defeasible Logic. RuleML, LNCS 5321, 89-103. (2008).

[14] G. Governatori and A. Stranieri. Towards the application of association rules for defeasible rules discovery In *Legal Knowledge and Information Systems*, JURIX, IOS Press, 63-75. (2001).

[15] J. Han, J. Pei and Y. Yin: Mining frequent patterns without candidate generation. *In Proc. ACM-SIGMOD Int'l Conf. Management of Data*. (2000).

[16] C. Hébert, B. Crémilleux: Optimized Rule Mining Through a Unified Framework for Interestingness Measures. DaWaK: LNCS 4081, 238-247. (2006).

[17] E. Hoekx, J. Van den Bussche: Mining for Tree-Query Associations in a Graph. *ICDM* 2006: 254-264.

[18] R. Huebner: Diversity-Based Interestingness Measures For Association Rule Mining. Proceedings of ASBBS Volume 16 Number 1, (2009).

[19] B. Johnston, Guido Governatori: An algorithm for the induction of defeasible logic theories from databases. Proceedings of the 14th Australasian Database Conference, 75-83. (2003).

[20] P. Kazienko: Mining Indirect Association Rules For Web Recommendation. Int. J. Appl. Math. Comput. Sci., Vol. 19, No. 1, 165–186. (2009).

[21] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. Verkamo: Finding interesting rules from large sets of discovered association rules. In *Proc. 3rd Int'l Conf. on Information and Knowledge Management*. (1994).

[22] M. J. Maher, A. Rock, G. Antoniou, D. Billington, T. Miller: Efficient Defeasible Reasoning Systems. *International Journal on Artificial Intelligence Tools* 10(4): 483-501 (2001).

[23] C. Marinica, F. Guillet, and H. Briand: Post-Processing of Discovered Association Rules Using Ontologies. The Second International Workshop on Domain Driven Data Mining, Pisa, Italy (2008).

[24] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal: Closed sets based discovery of small covers for association rules. In *Proc. BDA'99 Conference*, 361-381 (1999).

[25] N. Pasquier, R. Taouil, I. Bastide, G. Stume, and L. Lakhal: Generating a Condensed Representation for Association Rules. In *Journal of Intelligent Information Systems*, 24:1, 29-60 (2005).

[26] P. Pothipruk, G. Governatori: ALE Defeasible Description Logic. Australian Conference on Artificial Intelligence. 110-119 (2006).

[27] J. Sandvig, B. Mobasher Robustness of collaborative recommendation based on association rule mining, Proceedings of the ACM Conference on Recommender Systems (2007).

[28] W. Shen, K. Ong, B. Mitbander, and C. Zaniolo: Metaqueries for data mining. In Fayaad, U. et al. Eds. Advances in Knowledge Discovery and Data Mining. (1996).

[29] I. Song, G. Governatori: Nested Rules in Defeasible Logic. RuleML, LNCS 3791, 204-208 (2005).

[30] H. Toivonen, M. Klemettinen, P. Ronkainer, K. Hatonen, and H. Mannila: Pruning and grouping discovered association rules. In *ECML Workshop on Statistics, Machine Learning and KDD*. (1995).

[31] M. Zaki: Generating Non-Redundant Association Rules. In *Proc. of the Sixth ACMSIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 34-43, (2000).

[32] w3c. OWL Ontology Web Language Reference. In: http://www.w3.org/TR/2004/REC-owl-ref-20040210.

[33] w3c. RDF/XML Syntax Specification. In: http://www.w3.org/TR/rdf-syntax-grammar//.

[34] w3c. RDF Schema. In: http://www.w3.org/TR/rdf-schema/.