Plausible Reasoning in an Algorithm for Generation of Good Classification Tests

Xenia Naidenova[1[0000-0003-2377-7093]], Vladimir Parkhomenko[2[0000-0001-7757-377X]], Tatyana Martirova[1[0000-0003-0000-6608]], and Alexander Schukin[2[0000-0002-9534-824X]]

[1] Military medical academy, Saint Petersburg, Russia
ksennaid@gmail.com, martta462@yandex.ru
[2] Peter the Great St. Petersburg Polytechnic University, Russia
{vladimir.parkhomenko,alexander.schukin}@spbstu.ru

Abstract. The paper is devoted to the application of the plausible reasoning principles to symbolic machine learning. It seems for us that the applications are essential and necessary to improve the efficiency of ML algorithms. Many such algorithms produce and use rules in the form of implication. The generation of these rules with respect to the object classes is discussed. Our classification rules are specific. Their premise part, called good closed tests (GCTs), should cover as many objects as possible. One of the algorithms of GCTs generation called NIAGARA is presented. The algorithm is revisited and new procedures, based on plausible reasoning, are proposed. Their correctness is proved in propositions. We use the following rules: implication, interdiction, inductive rules of extending current sets of goal-oriented objects, rules of pruning the domain of searching solution. They allow to rise the effectiveness of algorithms.

Keywords: Plausible reasoning · Closed itemsets · Good diagnostic test · Good test analysis · Symbolic machine learning.

1 Introduction

In the paper, we propose a new, more efficient version of the algorithm NIAGARA that has been proposed in [15] for generating maximally redundant good tests (GMRTs) introduced in [14]. The increase in the efficiency of the algorithm is based on the introduction of several new implicative rules of plausible reasoning, which make it possible to use directly the known properties of the target objects generated in the algorithm. The rules of plausible reasoning play a huge role in the design of data mining algorithms. In turn, the rules of plausible reasoning are generated using machine learning methods.

The rules of plausible reasoning are productive in solving the following problems: forming contexts for the problem being solved, pruning the search space for solutions, forming descriptions of objects, identifying essential elements in data processing, extracting relationships between elements in the field of finding solutions, interpreting the results obtained, and more. From this point of view, it is legitimate to consider the rules of plausible reasoning as a system element in the tasks of constructing data mining algorithms.

The paper is organized as follows. Section 2 gives a short introduction to plausible reasoning. We discuss only such rules, which are applied in our new version of NIAGARA. Section 3 gives definitions of GMRTs, Section 4 discusses applying the plausible reasoning rules in the algorithm NIAGARA-2 for GMRTs generation and gives a running example. Finally, at the end of the paper, we discuss some related works.

2 Plausible reasoning Rules

In the paper, we consider the plausible reasoning rules presented as if-then logical assertions. We divide these rules into the following categories:

- INSTANCES or relationships between objects or facts really observed;
- RULES OF THE 1ST TYPE or logical assertions based on the regular relationships between objects (or/and their properties);
- RULES OF THE 2ND TYPE or plausible reasoning rules with the help of which rules of the first type are applied, modified, and mined from data.

A description of these rules and the information of theirs application are given in [14].

2.1 Rules of the 1st type used in the algorithm NIAGARA-2

Implication The implication rule is a classical logical rule. It has a form like $x, y, z \to w$, the left and right parts of which are called an antecedent (premise) and a consequent (conclusion) respectively. If all values in a premise are true, then the conclusion is true.

Interdiction or forbidden rule The interdiction is a type of implication rule. It can be regarded as an equation like $x, y, z \to$ false (never). One can present the interdiction via a number of implication, e.g. $x, y \to$ not z; $x, z \to$ not y; $y, z \to$ not x.

2.2 Rules of the 2nd type used in the algorithm NIAGARA-2

Suppose, that y is a set of attribute values, which are observed simultaneously. Let `p`, `antecedent(p)` and `consequent(p)` be an implication, its antecedent and consequent, respectively. Then we can see the following applications of implication and interdiction.

Implication application If `antecedent(p)` $\subseteq y$, then y is extendable by `consequent(p)`: $y \leftarrow y \cup$ `consequent(p)`. This application uses modus ponens: if X, then Y; X; hence Y.

Interdiction application Assume, that `p` is $z \to$ not k. If `antecedent(p)` $\subseteq y$, then k is the forbidden value for all extensions of y. This application uses modus ponendo tollens: either X or Y; X; hence not Y; either X or Y; Y; hence not X. X and Y are called alternatives.

Later we use these rules to extend elements of initial input set to obtain GMRTs.

3 Good test analysis

Let us recall main definitions of Good tests analysis [13,15,14].

Suppose, that R and S are a multi-valued table [7] and a set of object indices, respectively. Then $R(k)$ and $S(k)$ are called as the set of k-object descriptions and the set of k-object indices, respectively, where $k \in K$ is a class of objects, e.g. "positive" $(+)$ or "negative" (-), in a particular case.

Let FM be $R \setminus R(k)$, i.e. the set of object descriptions different from a k class. Denote by U and T the set of attributes and the set of attribute values (or just "values", for simplicity), respectively. Each value appears at least in one object description (object, for short) from R. Denote by n and $dom(Atr)$ the total number of object indices and the domain of an attribute $Atr \in U$.

A Galois connection [16] from attribute values to object indices is given with a function $s(\cdot)$, which takes t, a subset of the set T of disjoint attribute values, and returns a subset of object indices. We assume, that attribute values are from a nominal scale [7].

We call $t \subseteq T$, $s(t) \neq \emptyset$, a *diagnostic test* for $R(k)$ iff $t \not\subset d$, $\forall d \in FM$. To be a diagnostic test t means, that the condition $s(t) \subseteq S(k)$ and an implication rule `p` "if t, then k", are satisfied. Obviously, t is an `antecedent(p)`, where p is a rule of the 1st type.

Denote by $DT(k)$ a set of all diagnostic tests $\{t \mid \forall t \text{ for} R(k)\}$. If there are $t, d \in DT(k)$, one and only one of the following conditions is satisfied: $s(t) \subset s(d)$, $s(d) \supset s(t)$ and $s(t) \sim s(d)$, where *sim* stands for the incompatibility relation.

Then a non-empty test t is called a *good test* for a $R(k)$ iff $s(t) \subseteq S(k)$ and simultaneously $(\forall g)\{s(t) \cup g\}$, $g \in S(k) \setminus s(t)$ is not a test for $R(k)$.

A set $t \subseteq T$ of values is called *maximally redundant* if for any implication rule $Y \to z$ in R we have $(Y \subseteq t) \to (z \in t)$.

A Galois connection $S \to T$ is given as $s(B) = \{g \mid g \in S, B \subseteq t_i$, where t_i is an object description$\}$. Another Galois connection $T \to S$ is given as $t(s) = \{$intersection of all $t_i \mid t_i \subseteq T, i \in s\}$.

There are two closure operators [7] $generalization\_of(t) = t'' = t(s(t))$ and $generalization\_of(s) = s'' = s(t(s))$. A set t is *closed* if $t(s(t)) = t$ and s is closed if $s(t(s) = s$.

The interconnection between good tests analysis and FCA has been elucidated in [15]. Moreover, in the paper it is shown that maximally redundant good test are very popular classifiers, e.g. please, see interconnections with other symbolic classifiers like JSM-hyposeses [12,5].

4 Algorithm NIAGARA-2 for good tests generation with plausible reasoning

4.1 An idea of the algorithm

NIAGaRa-2 is a batch algorithm for inferring all GMRTs for a set of positive (or negative) objects. It is a new variant of a NIAGARA algorithm described in [14]. For this goal the sequence $S_0 \subseteq \ldots \subseteq S_q \subseteq S_{q+1} \subseteq \ldots \subseteq S_{q+m}$, where S_q is the set of all subsets of $S(+)$ of cardinality equal q, is inferred. The generalization rule is applied to each element $(s_q, t(s_q))$, beginning with two initial sets $R(+) = \{t_1, t_2, \ldots, t_i, \ldots, t_{nt}\}$ and $S(+) = \{1, 2, \ldots, i, \ldots, nt\}$, where nt is the number of positive objects.

The procedure DEBUT (Figure 2-a) forms the extensions of elements of the initial set $S(+) = \{1, 2, \ldots, i, j, \ldots, nt\}$ and returns the set $\{s_{12}, s_{13}, \ldots, s_{ij}, \ldots\}$, where $s_{ij} = \{i, j\}$, $1 < i < j < nt$.

If $s_{ij} = i, j$, such that $(s_{ij}, t(s_{ij}))$ is not a test for $R(+)$, then it is saved in the set Q of forbidden pairs of objects. If $s_{ij} = i, j$, such that $(s_{ij}, t(s_{ij}))$ is a test for $R(+)$, then it is generalized (closed) and the result $s = generalization_{of}(s_{ij})$ is inserted in S(test).

When DEBUT is over, it is necessary to check whether an element s of S(test) corresponds to a GMRT for $R(+)$ or not. For this goal, the following rule is used: if some object j, for $j = 1, \ldots, nt$, belongs to one and only one element s of $S(test)$, then s cannot be extended and, consequently, s corresponds to a GMRT and it is deleted from S(test) and inserted into STGOOD.

S(test) is the partially ordered set containing all $s = \{i_1, i_2, \ldots, iq\}$, $q = 1, 2, \ldots, nt$, satisfying the condition that $(s, t(s))$ is a test for $R(+)$ but not a good one. STGOOD is the partially ordered set containing all $s = \{i_1, i_2, \ldots, iq\}$, $q = 1, 2, \ldots, nt$, satisfying the condition that $(s, t(s))$ is a GMRT for a given set of positive examples. For every s in S(test), the set $ext(s)$ of all possible extensions of s, which correspond to tests for $R(+)$, is formed.

The procedure SELECT(s) (Figure 3-a) returns the set select(s) of objects that are admissible to produce the extensions of s corresponding to tests.

The sets S(test) and STGOOD are lexicographically ordered. Context(s) is defined as the set of object indices which can be currently used for extending $s \subseteq S(+)$: context(s) = $\{\{\cup s*\} |$ Prefix$(s*) =$ Prefix$(s), s \prec s*\}$, where Prefix(s) is the first index in s and $\prec$ is the symbol of the lexicographical order.

Introducing the concept of context(s) allows the decomposition of the algorithm into independent processes. The set $V(s)$ is determined as the set of object indices, which must be deleted from context(s) to avoid duplicate generation. CAND(s) = context(s)$\setminus V(s)$, where $V(s) = \{\cup s*, s \subseteq s*, s* \in \{S(test) \setminus s \cup STGOOD\}\}$.

The set $V(s)$ is the union of all the set in $\{S(test) \setminus s \cup STGOOD\}$ containing s, hence, s is in the intersection of these sets. If we want an extension of s not to be included in any element of $\{S(test) \setminus s \cup STGOOD\}$, we must use, for extending s, the object indices not appearing simultaneously with s in the set $V(s)$.

```
Algorithm
  Input: R(+), R(−), nt, S(+) = {1, . . . , nt}.
  Output: TGOOD
  DEBUT;
  while S(test) ≠ ∅ do
     SELECT(s);
     EXTENSION(s);
     ANALYSIS_OF_EXTENSION(s);
  while STGOOD do
     TGOOD ← {t(s)| s ∈ STGOOD};
     STGOOD\s;
```

Fig. 1. Main Algorithm of NIAGARA-2

Select(s) is determined as the set of object indices admissible for extending s (for this goal the set Q of forbidden pairs of indices is used): select(s) $= \{i, i \in CAND(s) : (\forall j)(j \in s), i, j \notin \{STGOOD \vee Q\}\}$.

Note that the following plausible rules are applied in NIAGARA-2:

– forbidden rules;
– rule of extending sets with cutting the searching spaces not containing any solution;
– the implicative rules (based on the known properties of good tests as formal concepts).

4.2 Pseudo-code of NIAGARA-2 algorithm

In the section, the pseudocode of NIAGARA-2 is given. In comparison with the previous algorithm version [14] there are the following improvements:

– in Fig. 3-a, the new function "determining context(s)" substitutes the previous function determining context; context(s) has been defined above;
– in Fig. 2-b, the operation of deleting from S(text) not-extendable sets and transferring them to STGOOD is proposed, see, please, line numbers 7–12.

Main improvements are presented in corollary 3 in the next subsection. The positive impact of the improvements is measured in the subsection with the example discussion.

Under formation of STGOOD, a set s of indices is stored in STGOOD if and only if it is not included in any collection of this set. In a pseudocode we write $gnrOf$ instead of $generalization\_of$ for short.

4.3 Main features of NIAGARA and NIAGARA-2 algorithms

Our algorithm is based on generation only closed sets. Let a set X be not a closed one and $c(X) = s(t(X))$ be its closure. Consider two possibilities: $c(X) = X$ and $X \subset c(X)$. In the first case, X is closed and it is to be extended. In the second case, X is not closed.

Procedure `DEBUT()`

```
Input: R(+), R(−), nt, S(+) = {1, ..., nt}.
Output: S(test)— the set of collections of objects to be extended; Q — the set of forbidden pairs of objects, STGOOD.
STGOOD, Q, S(test) ← ∅;
for i ∈ {1, ..., nt} do
    sum(i) ← 0;
for i ∈ {S[1], ..., S[nt]} do
    for j ∈ {S[i + 1], ..., S[nt]} do
        if to_be_test(t{i, j}) = false then
            Q ← Q ∪ {i, j}
        else
            s'' ← gnrOf({i, j});
            foreach i in s'' do
                sum(i) ← sum(i) + 1;
            insert s'' into S(test) under lexic. order;
foreach i in 1,nt do        (1,nt overlined)
    if sum(i) = 1 then
        find s : i ∈ s, s ∈ S(test);
        insert s into STGOOD under lexic. order;
        delete s from S(test);
nts ← {∪s | s ∈ S(test)};
```

(a) DEBUT procedure

Procedure `ANALYSIS_OF_EXTENSION()`

```
Input: ext(s), S(test), STGOOD.
Output: the modified sets S(test) and STGOOD.
if ext(s) = ∅ then
    if s ⊂ s∗, s∗ ∈ S(test) then
        transfer s∗ from S(test) to STGOOD under lexic. order;
        delete s;
    else
        transfer s from S(test) to STGOOD under lexic. order;
if ||ext(s)|| = 1 then
    s = s_new, s_new ∈ ext(s);
    if s ⊄ s∗, s∗ ∈ S(test) then
        transfer s from S(test) to STGOOD under lexic. order;
    else
        transfer s∗ from S(test) to STGOOD under lexic. order;
else
    foreach s_new in ext(s) do
        insert s_new into S(test) under lexic. order;
        delete s;
```

(b) Analysis of extension

Fig. 2. Procedures DEBUT and Analysis of extension of NIAGARA-2

Proposition 1. *If* $X \subset c(X)$*, then* $t(c(X)) \subset t(X)$ *but* $t(X) \subset t(c(X)) \rightarrow t(X) = t(c(X))$.

Corollary 1. *If* $t(X) = t(c(X))$*, then* $c(X)$ *can substitute* $X \in S(test)$ *and* X *can be deleted from* $S(test)$ *without loss of any solution.*

Proposition 2. *If an object index is included in one and only one set in* $S(test)$ *then this set cannot be extended.*

Corollary 2. *If an object index is included in one and only one set in* $S(test)$ *then this set is a good test.*

```
Algorithm SELECT(s)
  Input: s, nts, Q, S(test), STGOOD.
  Output: the set select(s) of
          objects for possible
          extension s.
  Determining context(s);
  if context(s) = ∅ then
  | select(s) ← ∅
  else
  | V(s) = {∪s', s ⊆ s', s' ∈
  |   {S(test) \ s ∪ STGOOD}};
  | if V(s) = ∅ then
  | | CAND(s) ←
  | |   context(s);
  | else
  | | CAND(s) ←
  | |   context(s) \ V(s);
  | | if CAND(s) = ∅ then
  | | | select(s) ← ∅;
  | | else
  | | | select(s) = {i ∈
  | | |   CAND(s) |
  | | |   (∀j)(j ∈ s), {i, j} ∉
  | | |   {STGOOD ∨ Q}};
```

(a) SELECT procedure

```
Procedure EXTENSION(s)
  Input: s, select(s), S(test), STGOOD.
  Output: ext(s) — the set of all
          extensions s' of s such
          that t(s') is a test.
  ext(s) = ∅;
  while select(s) ≠ ∅ do
  | foreach j in select(s) do
  | | s_new ← s ∪ j
  | if
  |  to_be_test(t(s_new))=false
  |  then
  | | eliminate s_new;
  | else
  | | s_new ← gnrOf(s_new);
  | | insert s_new into ext(s)
  | |   under lexic. order;
```

(b) EXTENSION procedure

Fig. 3. Procedures SELECT and EXTENSION of NIAGARA-2

The following proposition underpins a new reasoning rule increasing the effectiveness of NIAGARA-2 algorithm in comparison with the previous version.

Proposition 3. *If a closed set X of object indices contains a non-extendable subset Y, then X is also non-extendable.*

Corollary 3. *If a set X of object indices is a closed one and a test and it contains a non-extendable subset, then X is a good test.*

We call corollaries 1, 2 and 3 as Reasoning rules 1,2 and 3, respectively. Although the first and second propositions (and corollaries) are new, the appropriate rules have been already used in the NIAGARA.

The 3rd rule makes it possible to avoid extending sets containing a non-extendable subset. This rule, with logical point of view, is a new forbidden rule. With pattern lattice view, this rule means that if a pattern is non-extendable then all elements of its principal filter in the pattern lattice are also non-extendable (forbidden).

Fig. 4 gives an example of working Corollary 1. Green color and solid lines means the closed sets and links between them (see line diagram in [9]). Black color and dashed lines means the non-closed sets and links to theirs closures. These sets must to be deleted with substituting them by their closures.

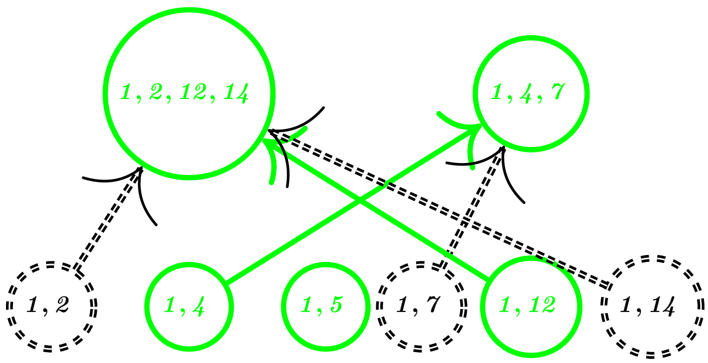

Fig. 4. An illustration of using Proposition 1 and Corollary 1

We can delete from consideration sets $\{1, 2\}$, $\{1, 7\}$, and $\{1, 14\}$ because $t(\{1, 2\}) = t(\{1, 2, 12, 14\}, t(\{1, 7\}) = t(\{1, 4, 7\})$, and $t(\{1, 14\}) = t(\{1, 2, 12, 14\})$ and, consequently, extending not-closed sets will give the same result as extending their closed supersets. Table 1 illustrates the preference of applying Boolean representing the sets of object indices. Data in Table 1 is in accordance with Fig. 4.

Table 1. An Example of the Boolean representation of sets

	1	2	4	5	7	12	14	Closed?	To delete?
1	1	1				1	1	1	
2	1	1						0	1
3	1		1		1			1	
4	1			1				1	
6	1				1			0	1
7	1					1		1	
8	1						1	0	1

5 Performance evaluation

5.1 Running example

The data to be processed is given in [14]. Now we give the results of applying the algorithm NIAGaRa-2 on the set of initial data. *Input*: $S = \{1, 2, \ldots, 14\}$; $T = \{A_1, \ldots, A_{26}\}$; $\text{STGOOD} = \emptyset; S(test) = \emptyset; Q = \emptyset$. *Output*: after implementation of the procedure DEBUT we have the same sets $S(test)$, Q, and STGOOD as in Tables 21, 22, and 23 from [14], respectively. Table 2 presents the extensions of the elements of S(test). The difference between this table and the appropriate table from [14] is discussed in the next section. The results from NIAGARA and NIAGARA-2 are the same, see, please Table 26 in [14].

.

Table 2. The Extensions of the Elements of S(test)

S	Context(s)	CAND(s)	Select(s)	Ext(s)	Delete $s \in S(test)$	STGOOD
1,4	5,7,12	5,12	12	∅	1	1,4,7
1,4,7						
1,5	12	∅	∅	∅	1	1,5,12
1,5,12						
1,12	∅	∅	∅	∅	1	
2,3,4	7,8.10	7,8,10	7,8	2,3,4,7	1	2,3,4,7
2,7	8, 10	8	8	2,7,8	1	2,7,8
2,8	10	∅	∅	∅	1	
2,10	∅	∅	∅	∅	1	2,10
3,7	8,10,11,12	8,10,11	8,11	∅	1	3,7,12
3,7,12						
3,8	10,11	10,11	10,11	∅	1	3,8
3,10	11	11	11	∅	1	3,10
3,11	∅	∅		∅	1	3,11
4,6,8,11	12	∅	∅	∅	1	4,6,8,11
4,6,11	12	∅	∅	∅	1	
4,7	8,11,12	8,11,12	8,11,12	4,7,12	1	4,7,12
4,8	11,12	11,12	11	4,8,11	1	
4,11	12	12	∅	∅	1	
4,12	∅	∅	∅	∅	1	
7,8	11,12	11,12	11	7,8,11	1	7,8,11
7,11	12	12	∅	∅	1	
7,12	∅	∅	∅	∅	1	
8,10	11	11	∅	∅	1	8,10
8,11	∅	∅	∅	∅	1	

5.2 Comparison of NIAGARA and NIAGARA-2 performance

We perform the running example with NIAGARA and NIAGARA-2 to discover all the GMRTs. The advantages of NIAGARA-2 are determined by two main optimizations: the context function and the new reasoning rule 3 based on Proposition 3. To compare the algorithms, we calculate the following measures of the whole process of GMRTs generation:

1. the general number of objects involved in the contexts (Conts);
2. the general number of objects included in the sets Cand(s) for all extended collections of objects (SCand);
3. the general number of objects included in the sets Select(s) for all extended collections of objects (SSelect);
4. the number of objects' collections to be extended (SExt).

The result of comparison is presented in Table 3.

Table 3. Comparison of NIAGARA and NIAGARA-2

Algorithm	Conts	SCand	SSelect	SExt
NIAGARA	59	55	16	25
NIAGARA-2	29	22	14	20

The Const and SCand values are decreased almost at 50% and 40%, respectively. The SSelect value of new algorithm version is not drastically differ from the previous one. SExt is decreased because of GMRTs, which are transferred from S(test) to STGOOD based on the new reasoning rule (the appropriate extensions are omitted). These GMRTS have the following numbers in Table 26 from [14]: 8, 10, 12.

6 Related works

Our prime interest is the methods by which the main problem of algorithms is solved: how to avoid repetitive generation of the same concept or how to test the uniqueness of the generated concept. With respect to this problem, some methods are summarized in [11,4]. There are surveys of closed sets applications in papers [17].

The significance of using contexts is shown by some refined argument in [6]. Also, we can note that the idea of using decision trees (classification structure) for an organization principle of mining ontology from texts is proposed in [10].

Reasoning rules are widely used in all the algorithms dealing with generating frequent closed itemsets. For example, the rule analogous Rule 1 in NIAGARA-2 is supported by lemma 3 in [18]. However, Reasoning Rules 2 and 3 are original and used only to generate GMRTs.

In many algorithms, an inductive rule is applied based on a level wise manner of extending the attributes' sets, attributes' values sets or objects (indices of objects) such that sets of the size q are built from the sets of the size $q-1$ of the previous level. Each set can be built if and only if, there are all its proper subsets at the previous level. In algorithms, the different properties of sets are checked. If a collection does not possess a required property, then it can be deleted from consideration. It strongly reduces the number of sets of all the following levels to be built.

The Level-wise manner of generating itemsets is used for extracting association rules from data. Algorithm AIS for mining association is introduced in [1]. The new algorithms the Apriori, AprioriTid, and AprioriHybrid are improved versions of the first algorithm for mining association rules.

A lattice-theoretic foundation for the task of mining associations based on the formal concept analysis has been done in [19]. It has been showed that the set of frequent concepts uniquely determines all the frequent itemsets. The lattice of frequent concepts can also be used to obtain a rule of generating sets from which all associations can be derived.

There are two strategies to generate concepts with batch algorithms: descending or top-down and ascending or bottom-up. However, it is important whether the leading process consists directly in generating all subsets of objects (extents of concepts) or all subsets of attributes (intents of concepts). Bordat's algorithm [3] uses a top-down strategy for inferring concepts in "breadth first" manner. The leading process of this algorithm is generating the subsets of objects of a given context with diminishing more and more their power. In Ganter's NextClosure algorithm [8], the leading process is the building of lexically ordered attribute subsets.

We can conclude that each algorithm generating itemsets underlying the formation of logical rules (implication, association rules, functional dependencies, formal concepts and many others) uses the plausible reasoning rules in one manner or another and this fact gives the right to assert that the algorithms in question can be considered as models of human thought processes.

Conclusion

We analyzed the application of plausible reasoning in NIAGARA and NIAGARA-2 for a GMRTs generation. Moreover, some new procedures based on plausible reasoning help us to make the algorithm NIAGARA-2 be more efficient. The algorithm time performance is improved thanks to the following new procedures: extending current sets of goal-oriented objects (with the use of implication, based on the properties of closed good tests, interdictions, rules of extension) and pruning a searching space. Their correctness is presented.

The future works include providing experiments to show the effectiveness of the optimizations proposed.

References

1. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD international conference on Management of data. pp. 207–216 (1993)
2. Birkhoff, G.: Lattice theory, vol. 25. American Mathematical Soc. (1940)
3. Bordat, J.P.: Calcul pratique du treillis de galois d'une correspondance. Mathématiques et Sciences humaines **96**, 31–47 (1986)
4. Carpineto, C., Romano, G.: Concept Data Analysis: Theory and Applications. John Wiley & Sons, Ltd, Chichester, UK (jul 2004)
5. Finn, V.: On machine-oriented formalization of plausible reasoning in the style of F.Backon–J.S. Mill. Semiotika i Informatika **20**, 35–101 (1983), (in Russian)
6. Galitsky, B., Ilvovsky, D.I., Goncharova, E.: Organizing contexts as a lattice of decision trees for machine reading comprehension. In: Kuznetsov, S.O., Napoli, A., Rudolph, S. (eds.) Proceedings of the 10th International Workshop "What can FCA do for Artificial Intelligence?" co-located with the 31st International Joint Conference on Artificial Intelligence (IJCAI-ECAI 2022), Vienna, Austria, July 23, 2022. CEUR Workshop Proceedings, vol. 3233, pp. 75–87. CEUR-WS.org (2022), `https://ceur-ws.org/Vol-3233/paper8.pdf`
7. Ganter, B., Wille, R.: Formal concept analysis: mathematical foundations. Springer, Berlin (1999)
8. Ganter, B.: Two basic algorithms in concept analysis. In: Kwuida, L., Sertkaya, B. (eds.) Formal Concept Analysis. pp. 312–340. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
9. Ganter, B., Kuznetsov, S.: Hypotheses and version spaces. In: Conceptual Structures for Knowledge Creation and Communication, Lecture Notes in Computer Science, vol. 2746, pp. 83–95 (2003)
10. Goncharova, E., Ilvovsky, D.I., Galitsky, B.: Concept-based chatbot for interactive query refinement in product search. In: Kuznetsov, S.O., Napoli, A., Rudolph, S. (eds.) Proceedings of the 9th International Workshop "What can FCA do for Artificial Intelligence?" co-located with the 30th International Joint Conference on Artificial Intelligence (IJCAI 2021), Montréal, Québec, Canada, August 21, 2021. CEUR Workshop Proceedings, vol. 2972, pp. 51–58. CEUR-WS.org (2021), `https://ceur-ws.org/Vol-2972/paper5.pdf`
11. Kuznetsov, S.O., Obiedkov, S.A.: Comparing performance of algorithms for generating concept lattices. Journal of Experimental & Theoretical Artificial Intelligence **14**(2-3), 189–216 (2002)
12. Kuznetsov, S.: Mathematical aspects of concept analysis. Journal of Mathematical Sciences **80**(2), 1654–1698 (1996)
13. Naidenova, X.A., Polegaeva, J.G.: An Algorithm of Finding the Best Diagnostic Tests. In: Mintz, G., Lorents, E. (eds.) The 4-th All Union Conference "Application of Mathematical Logic Methods". pp. 87 – 92 (1986), (in Russian)
14. Naidenova, X.: An incremental learning algorithm for inferring logical rules from examples in the framework of the common reasoning process. In: Triantaphyllou, E., Felici, G. (eds.) Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques, Massive Comp., vol. 6, pp. 89–147. Springer (2006)
15. Naidenova, X., Buzmakov, A., Parkhomenko, V., Schukin, A.: Notes on relation between symbolic classifiers. In: CEUR Workshop Proceedings, CEUR-WS. vol. 1921, pp. 88–103 (2017)
16. Ore, O.: Galois connections. Trans. Amer. Math. Soc **55**, 494–513 (1944)

17. Poelmans, J., Ignatov, D.I., Kuznetsov, S.O., Dedene, G.: Formal concept analysis in knowledge processing: A survey on applications. Expert Systems with Applications **40**(16), 6538 – 6560 (2013)
18. Wang, J., Han, J., Pei, J.: Closet+: Searching for the best strategies for mining frequent closed itemsets. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 236–245. KDD '03, Association for Computing Machinery, New York, NY, USA (2003). https://doi.org/10.1145/956750.956779, https://doi.org/10.1145/956750.956779
19. Zaki, M.J., Ogihara, M.: Theoretical foundations of association rules. In: 3rd ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (June 1998)