Adaptation knowledge discovery using positive and negative cases

Jean Lieber, Emmanuel Nauer

Adaptation knowledge discovery using positive and negative cases

Jean Lieber and Emmanuel Nauer

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Abstract. Case-based reasoning usually exploits positive source cases, each of them consisting in a problem and a correct solution to this problem. Now, the general issue of exploiting also negative cases—i.e., problem-solution pairs where the solution answers incorrectly the problem—can be raised. Indeed, such cases are "naturally" generated by a CBR system as long as it sometimes proposes incorrect solutions. This paper aims at addressing this issue for adaptation knowledge (AK) discovery: how positive and negative cases can be used for this purpose. The idea is that positive cases are used to propose adaptation rules and that negative cases are used to filter out some of these rules. In a preliminary work, this kind of AK discovery has been applied using frequent closed itemset (FCI) extraction on variations within the case base and tested on a toy Boolean use case, with promising first results. This paper resumes this study and evaluates it on 4 benchmarks, which confirms the benefit of exploiting negative cases for AK discovery. This involves some adjustments in the data preparation and in adaptation rule filtering, in particular because FCI extraction works only with Boolean features, hence some methodology lessons learned for AK discovery with positive and negative cases.

Keywords: adaptation knowledge discovery, negative cases, closed itemset extraction, case-based reasoning

1 Introduction

Case-based reasoning (CBR) [13] aims at solving a new problem—the *target problem*—thanks to a set of cases (the *case base*), where a case is a pair consisting of a problem and a solution to this problem. A *source case* is a case from the case base, consisting of a *source problem* and one of its solutions. The classical approach to CBR consists in selecting source cases *similar* to the target problem and adapting them to solve it. The adaptation step may use different approaches, one of them is the use of adaptation knowledge (AK) which can be acquired automatically using AK discovery processes. For this purpose, CBR usually exploits positive source cases [1–3, 5]. However, CBR systems sometimes produce also incorrect solutions, generating negative cases, i.e. problem-solution pairs where the solution answers incorrectly the problem. These negative cases may play an important role, especially in the knowledge discovery process, by improving the quality of the AK being extracted and, so, by improving the results of the CBR system itself [8].

The objective of this paper is to study how the exploitation of negative cases (in addition to positive cases), which was only evaluated on a toy use case, can be used

on different applications. Several existing machine learning datasets have been used for this purpose. The initial process of [8], which requires the representation of the problem and of the solution with Boolean attributes requires two major adaptations. First, the cases which are represented by other types of attributes in particular by nominal values, has to be transformed to fit the Boolean representation. Second, adaptation rules extracted by the AK process have to be filtered: this is an application-dependent task. The idea, beside this filtering, is to use the most efficient adaptation rules to solve a new problem.

The paper is organized as follows. Section 2 introduces the motivations and the preliminaries for this work. A reminder about frequent closed itemset (FCI) extraction and CBR are presented. Section 3 presents the specific approach for exploiting positive and negative cases in an AK discovery process. Section 4 introduces a dataset selection coming from a machine learning test database, and presents the evaluation of the AK discovery approach based on positive and negative cases on 4 selected benchmarks. Section 5 concludes and highlights the methodology lessons learned for AK discovery and then points out lines for future research.

2 Motivations and preliminaries

Many types of CBR applications are concerned by the use of AK and especially adaptation rules to solve new problems. For example, it has been shown that using AK can benefit to TAAABLE, a cooking system [5] which adapts cooking recipes. In [8], a preliminary study on automatically generated datasets has shown that exploiting negative cases in addition to positive ones for AK discovery improves the quality of the adaptation rules being extracted, which, in turn, improves the result of the CBR system. The approach is based on a Boolean representation of the cases. The AK discovery process uses variations between pairs of cases and FCI extraction to compute the adaptation rules.

This section first introduces some assumptions and notations about CBR. The Boolean setting which is at the basis of this work is then explained, and in particular, it is explained how the cases are represented in this Boolean framework, how the variations between cases are represented and computed, and, finally, how FCI extraction can be applied to compute adaptation rules that can be used to solve a new problem.

2.1 Assumptions and notations about CBR

Let $\mathcal{P}$ and $\mathcal{S}$ be two sets. A *problem* (resp., a *solution*) is an element of $\mathcal{P}$ (resp., of $\mathcal{S}$). The existence of a binary relation with the semantics "has for solution" is assumed, though it is not completely known to the CBR system. Moreover, it is assumed that this relation is functional (every problem has exactly one correct solution). Let $\mathtt{f}$ be the function from $\mathcal{P}$ to $\mathcal{S}$ such that $\mathtt{y} = \mathtt{f}(\mathtt{x})$ if $\mathtt{y}$ is the solution of $\mathtt{x}$. A *case* is a pair $(\mathtt{x}, \mathtt{y}) \in \mathcal{P} \times \mathcal{S}$ where $\mathtt{y} = \mathtt{f}(\mathtt{x})$. The fact that "has for solution" is functional is useful for making the evaluation process automatic: $\mathtt{y}$ is a *correct* solution to $\mathtt{x}$ iff $\mathtt{y} = \mathtt{f}(\mathtt{x})$, hence if $(\mathtt{x}, \mathtt{y})$ belongs to the test set and $\widehat{\mathtt{y}}$ is a solution returned by the CBR system, this solution is correct iff $\widehat{\mathtt{y}} = \mathtt{y}$.

A CBR system on $(\mathcal{P}, \mathcal{S}, \mathtt{f})$ is built with a knowledge base $\mathtt{KB} = (\mathtt{CB}, \mathtt{DK}, \mathtt{RK}, \mathtt{AK})$ where CB, the case base, is a finite set of cases, DK is the domain knowledge, RK is the retrieval knowledge (in this work, $\mathtt{RK} = \mathtt{dist}$, a distance function on $\mathcal{P}$), and AK is the adaptation knowledge that takes the form of adaptation rules.

A CBR system on $(\mathcal{P}, \mathcal{S}, \mathtt{f})$ aims at associating to a query problem $\mathtt{x}^{\mathtt{tgt}}$ a $\mathtt{y}^{\mathtt{tgt}} \in \mathcal{S}$, denoted by $\mathtt{y}^{\mathtt{tgt}} = f_{\mathrm{CBR}}(\mathtt{x}^{\mathtt{tgt}})$. The function f_{CBR} is intended to be an approximation of f. It is built thanks to the following functions:

- the retrieval function, with the profile $\mathtt{retrieval} : \mathtt{x}^{\mathtt{tgt}} \mapsto (\mathtt{x}^s, \mathtt{y}^s) \in \mathtt{CB}$;
- the adaptation function, with the profile $\mathtt{adaptation} : ((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}}) \mapsto \mathtt{y}^{\mathtt{tgt}} \in \mathcal{S}$; it is usually based on DK and AK. $((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}})$ is an *adaptation problem*.

Thus $f_{\mathrm{CBR}}(\mathtt{x}^{\mathtt{tgt}}) = \mathtt{adaptation}(\mathtt{retrieval}(\mathtt{x}^{\mathtt{tgt}}), \mathtt{x}^{\mathtt{tgt}})$.

With no domain knowledge and no adaptation knowledge ($\mathtt{DK} = \emptyset$ and $\mathtt{AK} = \emptyset$), the adaptation consists usually of a mere copy of the solution. This process is called *null adaptation*:

$$\mathtt{null\_adaptation} : ((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}}) \mapsto \mathtt{y}^s$$

Adaptation principle using adaptation rules. Generally speaking, an adaptation rule ar is a function mapping an adaptation problem $((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}}) \in \mathtt{CB} \times \mathcal{P}$ to $\mathtt{y}^{\mathtt{tgt}} \in \mathcal{S} \cup \{\mathtt{failure}\}$. A cases of failure ($\mathtt{y}^{\mathtt{tgt}} = \mathtt{failure}$) occurs when no adaptation rule ar is applicable on this adaptation problem. Else, $\mathtt{y}^{\mathtt{tgt}}$ is a proposed solution to $\mathtt{x}^{\mathtt{tgt}}$, by adaptation of $(\mathtt{x}^s, \mathtt{y}^s)$ according to ar.

A score $\mathtt{supp}(\mathtt{ar}) \geq 0$, called the support of the rule ar, is associated with ar; the higher is $\mathtt{supp}(\mathtt{ar})$, the more ar is preferred.

The adaptation consists in selecting the subset AAR of AK of applicable adaptation rules with maximum support: $\mathtt{ar} \in \mathtt{AAR}$ iff $\mathtt{ar}((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}}) \neq \mathtt{failure}$ and there exists no $\mathtt{ar}' \in \mathtt{AAR}$ such that $\mathtt{supp}(\mathtt{ar}') > \mathtt{supp}(\mathtt{ar})$.

2.2 Boolean setting illustrated with a Boolean function example

Initially, the Boolean setting has been chosen in [8] to validate the AK discovery approach by experiments using automatically generated Boolean functions as f. Let $\mathbb{B} = \{0, 1\}$ be the set of Boolean values. The Boolean operators are denoted by the connector symbols of propositional logic: for $a, b \in \mathbb{B}$, $\neg a = 1 - a$, $a \wedge b = \min(a, b)$, $a \vee b = \max(a, b)$. Let $p \geq 1$. In the examples, an element of $\mathbb{B}^p$ is noted without parentheses and commas: $(0, 0, 1, 1, 0, 0)$ is simply noted by 001100. The Hamming distance dist on $\mathbb{B}^p$ is defined by $\mathtt{dist}(a, b) = \sum_{i=1}^{p} |b_i - a_i|$. For example, with $p = 6$, $\mathtt{dist}(001100, 101110) = 2$.

Let $m, n \in \mathbb{N}^*$, $\mathcal{P} = \mathbb{B}^m$, $\mathcal{S} = \mathbb{B}^n$ and $\mathtt{f} : \mathcal{P} \to \mathcal{S}$, be a Boolean function to be approximated. A CBR system is considered on $(\mathcal{P}, \mathcal{S}, \mathtt{f})$ with $\mathtt{DK} = \emptyset$, $\mathtt{RK} = \mathtt{dist}$, the Hamming distance on $\mathcal{P}$, and AK a set of adaptation rules.

Table 1 introduces 4 cases on $\mathcal{P} = \mathbb{B}^6$, $\mathcal{S} = \mathbb{B}$, with a given f.

	x_1	x_2	x_3	x_4	x_5	x_6	$\mathtt{f}$
$\mathtt{c}^1$	0	0	1	1	0	0	0
$\mathtt{c}^2$	0	1	0	0	1	1	1
$\mathtt{c}^3$	0	1	1	0	0	0	1
$\mathtt{c}^4$	1	0	1	1	1	0	1

Table 1. An example of 4 problems of $\mathbb{B}^6$ with their solution on $\mathbb{B}$ for $\mathtt{f}(x) = x_1 \vee (x_2 \wedge x_3) \vee \neg x_4$.

Adaptation rule language. The adaptation rule language used in this work is based on the notion of variations between Booleans, as described hereafter. Given $\ell, r \in \mathbb{B}$ (ℓ stands for *left*, r for *right*), the variation from ℓ to r is represented by *variation symbols*. Each of the 4 ordered pairs (ℓ, r) is represented by a variation symbol v:

- $(\ell, r) = (\mathtt{1}, \mathtt{0})$ is represented by $v = \mathtt{-}$;
- $(\ell, r) = (\mathtt{0}, \mathtt{1})$ is represented by $v = \mathtt{+}$;
- $(\ell, r) = (\mathtt{0}, \mathtt{0})$ is represented by $v = \mathtt{=0}$;
- $(\ell, r) = (\mathtt{1}, \mathtt{1})$ is represented by $v = \mathtt{=1}$.

Given two cases $\mathtt{c}^1 = (\mathtt{x}^1, \mathtt{y}^1)$ and $\mathtt{c}^2 = (\mathtt{x}^2, \mathtt{y}^2)$, the variation V^{12} from $\mathtt{c}^1$ to $\mathtt{c}^2$ is encoded by the set of the expressions $\mathtt{x}_i^v$ and $\mathtt{y}_j^w$ such that v (resp., w) is a variation symbol from $\mathtt{x}_i^1$ to $\mathtt{x}_i^2$ (resp., from $\mathtt{y}_j^1$ to $\mathtt{y}_j^2$). For example, the variations from $\mathtt{c}^2 = ((\mathtt{010011}), \mathtt{1})$ to $\mathtt{c}^3 = ((\mathtt{011000}), \mathtt{1})$ are $V^{23} = \{\mathtt{x}_1^{=0}, \mathtt{x}_2^{=1}, \mathtt{x}_3^{+}, \mathtt{x}_4^{=0}, \mathtt{x}_5^{-}, \mathtt{x}_6^{-}, \mathtt{y}_1^{=1}\}$.

An *adaptation rule* $\mathtt{ar}$ is a set of expressions $\mathtt{x}_i^v$ and $\mathtt{y}_j^w$. It is applicable on an adaptation problem $((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}})$ if there exists $\mathtt{y}^{\mathtt{tgt}} \in \mathbb{B}^n$ such that $V^{st} \supseteq \mathtt{ar}$ (where V^{st} represents the variation from $(\mathtt{x}^s, \mathtt{y}^s)$ to $(\mathtt{x}^{\mathtt{tgt}}, \mathtt{y}^{\mathtt{tgt}})$). If it is applicable, then its application consists in choosing such a $\mathtt{y}^{\mathtt{tgt}}$. If several $\mathtt{y}^{\mathtt{tgt}}$'s exist, the chosen one is the closest to $\mathtt{y}^s$ according to the Hamming distance on $\mathcal{S} = \mathbb{B}^n$, meaning that if $\mathtt{ar}$ gives no constraint on some $\mathtt{y}_j^{\mathtt{tgt}}$ then $\mathtt{y}_j^{\mathtt{tgt}} = \mathtt{y}_j^s$. For example:

if $\mathtt{ar} = \{\mathtt{x}_1^{=0}, \mathtt{x}_2^{-}, \mathtt{x}_3^{+}, \mathtt{y}_1^{+}\}$, $(\mathtt{x}^s, \mathtt{y}^s) = (\mathtt{010}, \mathtt{00})$ and $\mathtt{x}^{\mathtt{tgt}} = \mathtt{001}$
then $\mathtt{ar}$ is applicable on $((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}})$ and $\mathtt{ar}((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}}) = \mathtt{y}^{\mathtt{tgt}} = \mathtt{10}$

2.3 Itemset extraction

Itemset extraction is a collection of data-mining methods for extracting regularities into data, by aggregating object items appearing together. Like formal concept analysis [7], itemset extraction algorithms start from a *formal context* K, defined by $K = (G, M, I)$, where G is a set of objects, M is a set of items, and I is the relation on $G \times M$ stating that an object is described by an item [7]. Table 2 shows an example of context, in which the 4 cases of Table 1 have been used to generate 12 case variations (variation combinations between all the possible pairs of distinct cases $\mathtt{c}^1$, $\mathtt{c}^2$, $\mathtt{c}^3$, $\mathtt{c}^4$) are described by 28 properties (x_i^v and y_j^w for $i \in [1, 6]$, $j \in \{1\}$ and $v, w \in \{\mathtt{-}, \mathtt{+}, \mathtt{=0}, \mathtt{=1}\}$). G is a set of 12 objects (V^{12}, V^{13}, ..., V^{43}), M is a set of 28 variation items.

	x_1=1	x_1=0	x_1-	x_1+	x_2=1	x_2=0	x_2-	x_2+	x_3=1	x_3=0	x_3-	x_3+	x_4=1	x_4=0	x_4-	x_4+	x_5=1	x_5=0	x_5-	x_5+	x_6=1	x_6=0	x_6-	x_6+	y_1=1	y_1=0	y_1-	y_1+
V^{12}		×						×			×				×					×				×				×
V^{13}		×						×	×						×			×				×						×
V^{14}				×		×			×				×							×		×							×
V^{21}		×					×					×				×			×				×				×		
V^{23}		×			×							×		×					×				×		×				
V^{24}				×			×					×				×	×						×		×				
V^{31}		×					×		×							×		×				×					×		
V^{32}		×			×						×			×						×				×	×				
V^{34}				×			×		×							×				×		×			×				
V^{41}			×			×			×				×						×			×					×		
V^{42}			×					×			×				×		×							×	×				
V^{43}			×					×	×						×				×			×			×				

Table 2. An example of formal context representing 12 case variations described by 28 properties: 24 properties describing the variations on the problem and 4 properties describing the variations on the solution.

An *itemset* I is a set of items, and the *support* of I, $\texttt{supp}(I)$, is the number of objects of the formal context having every items of I. I is frequent, with respect to a threshold $\tau_{\texttt{supp}}$, whenever $\texttt{supp}(I) \geq \tau_{\texttt{supp}}$. I is closed if it has no proper superset J ($I \subsetneq J$) with the same support. For example, $\{x_2^{=1}\}$ is an itemset and $\texttt{supp}(\{x_2^{=1}\}) = 2$ because exactly 2 cases have the property $x_2^{=1}$. However, $\{x_2^{=1}\}$ is not a closed itemset, because $\{x_2^{=1}, y_1^{=1}\}$ has the same support. For $\tau_{\texttt{supp}} = 6$, the frequent closed itemsets (FCIs) of this context are $\{x_1^{=0}\}$, $\{y_1^{=1}\}$, and $\{x_3^{=1}, x_6^{=0}\}$.

The experiments use CORON, a software platform which implements efficient algorithms for symbolic data mining and especially FCI computation [14].

3 Exploiting case variations for adaptation knowledge discovery with positive and negative cases

Exploiting case variations is not a new idea. [9] introduces this approach of AK learning based on pairwise comparisons of cases. This approach, also called *Case Difference Heuristic* in [10], has been applied in various domains such as medicine [3] or cooking [1, 5].

So, for ordered pairs of cases $(\texttt{c}^\ell, \texttt{c}^r)$ associated to their variations $V^{\ell r}$ forming a formal context, as the one presented in Table 2, an AK discovery process based on FCI can be run. Each extracted FCI produces an adaptation rule $\texttt{ar}$ with a support $\texttt{supp}(\texttt{ar})$, i.e. the number of $V^{\ell r}$ containing $\texttt{ar}$. For example, for $\tau_{\texttt{supp}} = 2$, $\{x_1^{=0}, x_2^{+}, x_4^{-}, \texttt{y}_1^{+}\}$ is an FCI which produces an adaptation rule. For a source case $(\texttt{00·1··}, \texttt{0}) \in \mathbb{B}^6 \times \mathbb{B}$, where each · represents 0 or 1, This adaptation rule can be used for solving any target problem $01{\cdot}0{\cdot\cdot}$. The proposed solution is 1, obtained by applying a + variation on the source solution 0.

The adaptation rule set extracted using this FCI approach depends on how the formal context is built, and in particular whether it is built only using positive source

cases, or using also negative source cases. For a case $(\mathtt{x},\mathtt{y})^s = (\mathtt{x}^s,\mathtt{y}^s) \in \mathtt{CB}$, the case $(\mathtt{x},\mathtt{y})^s$ is said *positive* if $\mathtt{y}^s$ is a correct solution for $\mathtt{x}^s$ ($\mathtt{y}^s = \mathtt{f}(\mathtt{x}^s)$) and *negative* otherwise. $\mathtt{CB}^+$ (resp. $\mathtt{CB}^-$) denotes the set of positive (resp. negative) cases of CB, with $\mathtt{CB} = \mathtt{CB}^+ \cup \mathtt{CB}^-$ and $\mathtt{CB}^+ \cap \mathtt{CB}^- = \emptyset$.

Starting from the two sets of cases $\mathtt{CB}^+$ and $\mathtt{CB}^-$, ordered pairs of cases $(\mathtt{c}^1,\mathtt{c}^2)$ are formed, with $\mathtt{c}^1 = (\mathtt{x}^1,\mathtt{y}^1) \in \mathtt{CB}^+$, $\mathtt{c}^2 = (\mathtt{x}^2,\mathtt{y}^2) \in \mathtt{CB}$ and $\mathtt{x}^1 \neq \mathtt{x}^2$. Each such pair is encoded by a set V^{12} of the variations from $\mathtt{x}^1_i$ to $\mathtt{x}^2_i$ and from $\mathtt{y}^1_j$ to $\mathtt{y}^2_j$, as presented before. When $\mathtt{c}^2 \in \mathtt{CB}^+$, the variation from $\mathtt{c}^1$ to $\mathtt{c}^2$ can be considered as a positive example of adaptation rule (i.e. the application of the rule produces a correct answer). When $\mathtt{c}^2 \in \mathtt{CB}^-$, the variation from $\mathtt{c}^1$ to $\mathtt{c}^2$ can be considered as a negative example of adaptation rule (i.e. the application of the produces an incorrect answer).

The AK learning process based on FCI extraction takes as input a set of $V^{\ell r}$ which is used to build the formal context. In [8], two approaches have been used to build the formal context. The first one consists in using each $V^{\ell r}$ as an object with variations between pairs of cases of $\mathtt{CB}^+$ as properties. This approach is denoted by AK^+ in the following. A limit of AK^+ is that it may produce too general adaptation rules (this issue has been discussed in [4]). For example, the formal context presented in Table 2 produces 15 rules containing a variation on y_1 for $\tau_{\mathtt{supp}} = 2$, e.g. $\{x_5^{=1}, y_1^{=1}\}$. This rule expresses the fact that only by knowing the variation =1 on x_5, the outcome is $\mathtt{y}^1 = 1$.

As the application of such a general rule is likely to give an incorrect answer, a second approach exploiting negative cases to filter too general adaptation rules is also considered. This idea is inspired from machine learning approaches based on version spaces [12] or its link with formal concept analysis [6, 11]: generating adaptation rules covering positive examples without covering negative ones. This approach exploiting both positive and negative cases is denoted by $AK^\pm$ in the following. Suppose that $\mathtt{c}^5 = (001111, 1) \in \mathtt{CB}^-$ (i.e. 1 is an incorrect solution for 001111 w.r.t. f). Using this negative case in the AK process eliminates some rules, and in particular the rule $\{x_5^{=1}, y_1^{=1}\}$, because $\mathtt{c}^5$ is a counterexample of the application of the rule. Applying this rule on 001111, the problem part of $\mathtt{c}^5$, produces 1, the solution of $\mathtt{c}^5$, which is an incorrect result.

To illustrate the benefit of $AK^\pm$ w.r.t. AK^+ but also w.r.t. the classical nearest neighbor approach, consider the cases of Table 1 and let $\mathtt{x}^{\mathtt{tgt}} = 000111$. For the nearest neighbor approach, according to dist, if considering the closest case of the case base w.r.t 000111 which is $(010011, 1)$ with $\mathtt{dist}(000111, 010011) = 2$, the result will be 1 (by null_adaptation). For the AK^+ solving approach, considering the same closest case than for the nearest neighbor approach and the rules computed on variations only between positive cases, the higher support rule $\{x_5^{=1}, y_1^{=1}\}$ can be applied with $c^2 = (010011, 1)$ and its application gives 1 as result. Considering now the same closest case than before but using rules computed on variations between positive cases but taking into account the negative examples. The previous rule $\{x_5^{=1}, y_1^{=1}\}$ is removed because of $c^5 = (001111, 0)$ which is a negative example for this rule. Another rule $\{x_1^{=0}, x_2^-, x_4^+, y_1^-\}$ can then be applied because of the variations $(x_1^{=0}, x_2^-, x_3^{=0}, x_4^+, x_5^{=1}, x_6^{=1}\}$ between $\mathtt{x}^s = 010011$ and $\mathtt{x}^{\mathtt{tgt}} = 000111$. The variation y_1^- can be applied on $\mathtt{y}^s = 1$, producing 0 as (correct) result.

4 Experiments on benchmarks

To test our approach on real applications, we have chosen to use benchmarks describing problems with their solutions. For this, the UCI Machine Learning Repository[1] and the Open ML website[2] have been used. These resources contain a great number of benchmarks. However, our Boolean representation formalism constrains to choose datasets where the problem and its solution are described by Boolean features or by features which can be transformed easily into Boolean ones (e.g. if an attribute is nominal, it can be transformed in several Boolean attributes, the value being encoded by only one true attribute among all these Boolean attributes).

For the experiments presented in this paper, we focused on datasets:

- where the solution is Boolean or nominal, typically benchmarks addressing a classification problem;
- containing enough data (i.e. at least 100 cases) because the experiments require enough data to build a learning dataset to extract adaptation rules and to have a testing dataset in order to evaluate the different problem solving approaches (nearest neighbors, AK^+ and $AK^\pm$).

The following sections not only present the execution on different benchmarks but also the methodological work of data preparation.

4.1 Experiment setting and evaluation methodology

The objective of the evaluation is to study, on various benchmarks, how the Boolean approach exploiting negative cases in addition to positive ones improves the results of the CBR system.

For each dataset, the raw data describing the problem and its solution are transformed into a Boolean encoding. The result of this encoding is the case base $\mathtt{CB}$. The size of $\mathtt{CB}$, denoted by $|\mathtt{CB}|$, depends on the dataset. For each run, $\mathtt{CB}$ is split in 2 subsets of the same size by a random selection of cases: $\mathtt{CB} = \mathtt{CB}_L \cup \mathtt{CB}_T$, with $\mathtt{CB}_L$, a set of cases used for learning AK, and $\mathtt{CB}_T$, a set of cases to test the different approaches. $\mathtt{CB}_L$ is then split in 2 subsets of the same size, also by a random selection of cases: $\mathtt{CB}_L = \mathtt{CB}^+ \cup \mathtt{CB}^-$. $\mathtt{CB}^+$ is the set of positive cases. To build $\mathtt{CB}^-$, the set of negative cases, each negative source case $(\mathtt{x}, \mathtt{y}^-)$ is generated from a positive source case $(\mathtt{x}, \mathtt{y})$ by a modification of the solution part. For $\mathcal{S} = \mathbb{B}^n$, with $n = 1$, the modification of the solution part consists in taking the negation of $\mathtt{y}$: $\mathtt{y}^- = \neg\mathtt{y}$. This is the case for the applications described in Sections 4.2, 4.3 and 4.4. When $n \geq 2$, the modification of the solution part depends on the application. Such modification is detailed further, in the context of the application described in Section 4.5.

Three adaptation approaches are tested: AK^+, $AK^\pm$, and NN, the classical nearest neighbor approach with `null_adaptation` for adaptation function. For NN, retrieval consists in selecting the 3 most similar source cases to the target problem (according to `dist`) and adaptation consists in making a vote among their solution parts.

[1] `https://archive.ics.uci.edu/ml/index.php`

[2] `https://www.openml.org/search?q=&type=data`

For the two approaches based on AK, all source cases for which adaptation rules can be applied participate to the problem solving. A vote on the results computed from the retrieved cases is used to associate a unique answer. Moreover, a vote is also used when using adaptation rules: 3 adaptation rules with the higher supports are used to adapt each of the source cases and the most frequent result wins.

The two AK-based approaches depends on two parameters: the number of rules being extracted by the knowledge discovery process and the specificity of the rules (i.e. the minimal size of the problem part of the rule). The impact of these two parameters on the results is illustrated and analyzed.

All the approaches are evaluated according to two measures: the precision `prec` and the correct answer rate `car`. Let `ntp` be the number of target problems posed to the system ($\texttt{ntp} = |\texttt{CB}_T|$), `na` be the number of (correct or incorrect) answers ($\texttt{ntp} - \texttt{na}$ is the number of target problems for which the system fails to propose a solution), and `nca` be the number of correct answers. So, the precision `prec` is defined as the average of the ratios $\frac{\texttt{nca}}{\texttt{na}}$, and the correct answer rate `car` is defined as the average of the ratios $\frac{\texttt{nca}}{\texttt{ntp}}$. The average is computed on 100 runs for each evaluation, for different number of rules used for adaptation and for different sizes of rules. The way the rules are filtered is now detailed.

Filtering rules. As mentioned above, using too general adaptation rules may produce incorrect results. General rules contain only variations on few variables, as more specific rules contain variations on more variables. In the TAAABLE project, some studies using adaptation rules on cooking recipes have shown that using more specific rules gives better results than using too general ones [5]. According to the idea that, all other things being equal, more a rule contains variables, less it is risky to use it, a given number of variables that must appear in the rules can be set (in particular for the variables x_i linked to the problem description).

4 benchmarks have been chosen, each of them raising different issues. For each benchmark, a description of the dataset and its purpose are first presented, followed by the results of the experiments and the analysis of these results.

4.2 Congressional Voting Records

This dataset[3] contains votes for each of the U.S. House of Representatives Congressmen on 16 key votes (e.g. education budget or duty free export). The value, for each key vote, can be 'yes', 'no', or 'no vote'. These 16 attributes, describing the votes of a given congressman (and which can be considered as the problem description), are linked to his/her political party, i.e. republican or democrat (which is the solution). So, the aim, for this benchmark, is to determine the political party of the voter from his/her votes. In this experiment, $\mathcal{P} = \mathbb{B}^{16}$ and $\mathcal{S} = \mathbb{B}$. The dataset contains 435 records: 232 records with a complete information about the problem ('yes' or 'no' for the 16 votes) and 203 with at least one 'no vote' value. For this first experiment, it has been chosen to remove

[3] `https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records`

nAR	100					200					300					400				
minPS	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
	prec																			
NN	.87	.88	.88	.88	.87	.88	.88	.87	.87	.88	.87	.88	.88	.88	.87	.88	.87	.87	.87	.88
AK^+	.77	.84	.91	.93	.95	.78	.82	.91	.92	.94	.77	.78	.91	.93	.93	.82	.83	.91	.92	.93
$AK^\pm$	**.94**	**.94**	**.95**	**.96**	**.96**	**.94**	**.94**	**.94**	**.95**	**.95**	**.94**	**.94**	**.94**	**.95**	**.95**	**.95**	**.94**	**.94**	**.94**	**.95**
	car																			
NN	.87	.88	.88	**.88**	**.87**	.88	.88	.87	.87	**.88**	.87	.88	.88	.88	**.87**	.88	.87	.87	.87	.88
AK^+	.77	.84	.91	.88	.82	.78	.82	.91	.90	.86	.77	.78	.91	.92	.87	.82	.83	.91	.91	.89
$AK^\pm$	**.91**	**.91**	**.91**	.86	.80	**.94**	**.93**	**.93**	**.91**	.87	**.93**	**.93**	**.94**	**.93**	**.86**	**.94**	**.94**	**.94**	**.93**	**.90**

Table 3. prec (first 3 lines) and car (last 3 lines) of the three approaches for different numbers of rules and different minimal problem sizes for the Congressional Voting benchmark.

nAR	100					200					300					400				
minPS	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
prec	.76	.83	.91	.92	.94	.82	.87	.92	.91	.92	.80	.87	.93	.91	.91	.79	.87	.93	.91	.92
car	.76	.83	.91	.88	.84	.82	.87	.92	.90	.87	.80	.87	.93	.90	.88	.79	.87	.93	.91	.88

Table 4. prec and car for AK^+ approach on the Congressional Voting Records benchmark with the use of the same number of positive cases for AK^+ adaptation rule extraction as for $AK^\pm$.

the 203 records which do not have the complete information, and thus to keep only the 232 complete records. These 232 records produce 152 cases as there are identical records (exactly the same vote on the 16 questions). The distribution for the solution of these 152 cases is 95 democrats and 57 republicans.

Let nAR be the number of adaptation rules being used in the AK-based approaches, and minPS, the minimal problem size, i.e the minimal number of variables linked to the problem that has to appear in the rules. Table 3 presents the results, for $\texttt{nAR} \in \{100, 200, 300, 400\}$ and $1 \leq \texttt{minPS} \leq 5$. The first three lines give the prec score, the last three lines give the car score for 100 runs.

Some important points can be noticed:

1. $AK^\pm$ is *systematically* better in prec than NN and AK^+ with also a better car, independently of the values of the 2 parameters nAR and minPS.
2. For AK^+, if too less rules (100 or 200) and too general rules ($\texttt{minPS} \leq 2$) are used, AK^+ gives not so good results: the prec and car scores are lower than for NN. However, with more specific rules, AK^+ overcomes NN for prec and car.
3. The choice of rules of good quality (i.e. specific rules) has an important impact on the results for AK^+: for a given number of rules, the prec increases when minPS increases. Indeed, this parameter has a real impact in filtering too general rules. For $AK^\pm$, the impact of this parameter is lower because bad adaptation rules are already filtered thanks to the negative cases.
4. The number of rules parameter has a minimal impact on prec compared to the specificity of the rules. For a given minPS, the prec is quite the same, independently of the number of rules. However, the number of rules parameter impacts the car because using more rules improves the car.

To summarize, the experiments on this dataset show that, except for a low number of rules or with an acceptance of too general rules, AK^+ gives better results than NN. But, the most important fact is that $AK^\pm$ overcomes NN and AK^+ in any situation. Moreover, $AK^\pm$ is very *stable* for the `prec`, and is not really impacted by the number of rules nor the specificity of the rules parameters. This is because the adaptation rules extracted for $AK^\pm$ are, in all situations, of better quality than for AK^+.

Is using both positive and negative cases better than using only positive cases? In all our experiments (the previous one but also the following ones), all approaches use the same case base $\texttt{CB}^+$ as source cases. $\texttt{CB}^+$ is also used to compute the adaptation rules in both AK-based approaches, but $AK^\pm$ uses in addition other cases: the negative cases of $\texttt{CB}^-$. So, there is a kind of disparity between AK^+ and $AK^\pm$ because $AK^\pm$ uses the double number of cases to acquire adaptation rules. This is justified by the fact that, for a CBR application, $\texttt{CB}^+$ is used for both AK approaches, while $\texttt{CB}^-$ can only be used by $AK^\pm$, knowing that negative cases can be acquired with a little effort during the use of the system. However, in order to compare AK^+ and $AK^\pm$ more "fairly", the following question has been raised: what kind of result could AK^+ give when using the same number of cases than $AK^\pm$ to build its rule set? To examine this issue, an experiment has been run focusing on the exploitation of the same number of cases for AK^+ and for $AK^\pm$. Instead of using only $\texttt{CB}^+$ for AK^+, the complete $\texttt{CB}_L$ is now used (without modification of cases to build $\texttt{CB}^-$).

The results, presented in Table 4, show that $AK^\pm$ outperforms AK^+ when using the same number of cases: `prec` and `car` of AK^+ in Table 4 are always lower than the `prec` and `car` of $AK^\pm$ in Table 3. This strenghens the idea that using negative cases in addition to positive cases improves the AK process.

4.3 Tic Tac Toe Endgame

This dataset[4] contains the possible situations of the Tic Tac Toe[5] where the player with the 'x' mark starts. There are 9 attributes describing the problem. Each attribute represents one position of the board, its value can be 'x' (resp, 'o') if an 'x' mark (resp. an 'o' mark) has been played on this position. A third value 'b' indicates that the position is empty (ie. neither 'x', neither 'o' appears). The solution is 1 if the player with the 'x' mark wins, and 0 otherwise. So, the aim of this dataset is, knowing which marks appears on some positions, to infer whether the player with the 'x' mark wins or not.

Data simplification: for this dataset, we have chosen to simplify the initial representation of 3 possible values for each variable, to a representation in which a variable has only 2 possible Boolean values: 1 if an 'x' mark is on the square and 0 otherwise (i.e. the square is marked by 'o' or 'b' in the raw data). In this experiment, $\mathcal{P} = \mathbb{B}^9$ and $\mathcal{S} = \mathbb{B}$. With this transformation, the dataset, containing initially 958 records, produces 272 cases, with a rather balanced solution distribution of 118 'x wins' and 154 'x does not win'.

[4] `https://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe+Endgame`

[5] Tic Tac Toe is a game where 2 players, one playing with an 'x' mark and one playing with an 'o' mark have to align 3 of their marks on a 3×3 board.

nAR	100					200					300					400				
minPS	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
	prec																			
NN	.60	.61	.60	.61	.61	.61	.60	.61	.60	.60	.60	.60	.61	.61	.60	.60	.61	.60	.60	.61
AK^+	.58	.60	.58	.60	.64	.57	.60	.59	.60	.63	.57	.60	.59	.61	.63	.59	.60	.59	.60	.63
$AK^\pm$	**.70**	**.72**	**.71**	**.69**	**.70**	**.71**	**.71**	**.70**	**.68**	**.69**	**.71**	**.71**	**.70**	**.69**	**.70**	**.71**	**.70**	**.70**	**.70**	**.69**
	car																			
NN	.60	.61	.60	**.61**	**.61**	.61	.60	.61	.60	**.60**	.60	.60	.61	.61	**.60**	.60	.61	.60	.60	.61
AK^+	.58	.60	.56	.51	.41	.57	.60	.59	.57	.49	.57	.60	.59	.60	.55	.59	.60	.59	.60	.58
$AK^\pm$	**.61**	**.61**	**.61**	.54	.42	**.66**	**.66**	**.65**	**.62**	.53	**.68**	**.68**	**.67**	**.65**	.58	**.69**	**.68**	**.69**	**.67**	**.62**

Table 5. prec and car of the three approaches for different numbers of rules and different minimal problem sizes for the Tic Tac Toe benchmark.

nAR	100					200					300					400				
minPS	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
	prec																			
NN	.80	.80	.80	.79	.80	.79	.79	.80	.79	.80	.79	.79	.79	.79	.79	.80	.80	.79	.80	.80
AK^+	.85	.85	.85	.84	.83	.85	.85	.85	.84	.83	.85	.86	.85	.84	.83	.85	.85	.84	.84	.83
$AK^\pm$	**.88**	**.87**	**.88**	**.88**	**.86**	**.87**	**.87**	**.87**	**.86**	**.86**	**.86**	**.87**	**.88**	**.86**	**.86**	**.86**	**.86**	**.86**	**.86**	**.86**
	car																			
NN	.80	.80	.80	**.79**	**.80**	.79	.79	.80	.79	**.80**	.79	.79	.79	.79	**.79**	.80	.80	.79	.80	**.80**
AK^+	**.85**	**.85**	**.83**	.78	.72	**.85**	**.85**	**.84**	**.80**	.75	**.85**	**.86**	**.85**	**.81**	.76	**.85**	**.85**	**.84**	**.82**	.78
$AK^\pm$	.81	.80	.76	.71	.64	.82	.82	.81	.77	.70	.83	.84	.82	.79	.73	.84	.84	.82	.80	.75

Table 6. prec and car of the three approaches for different numbers of rules and different minimal problem sizes for the Cardiac diagnosis benchmark.

Table 5 presents the results of the three adaptation approaches. This experiment gives similar results than for the first experiment. The hypothesis about $AK^\pm$ to be the better approach is confirmed, demonstrating once again the benefit of exploiting negative cases.

4.4 Cardiac diagnosis

This dataset[6] describes diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images. Each patient of the dataset is described by 22 binary features and is classified into two categories: normal (0) and abnormal (1). In this experiment, $\mathcal{P} = \mathbb{B}^{22}$ and $\mathcal{S} = \mathbb{B}$. The dataset, containing initially 267 records, produces 219 unique cases, with a distribution of 186 abnormal and 33 normal diagnoses. This biased distribution of the solutions may have an impact on the results, but as the 3 approaches which are compared are all concerned by this bias, it has been decided, for this dataset, not to balance this distribution and to keep all the cases to run the experiments. The results, presented in Table 6, show again the best behavior of the $AK^\pm$ approach, even with this bias in the dataset. For prec, the improvement of $AK^\pm$ is, in average, of $+8\%$ comparing to NN.

[6] https://archive.ics.uci.edu/ml/datasets/SPECT+Heart

4.5 Car evaluation

This section addresses Car evaluation.[7] Each model of car is described by 6 features:

- the buying price which values can be $vhigh$ (very high), $high$, med or low;
- the maintenance price: $vhigh$, $high$, med or low;
- the number of doors: 2, 3, 4 or $5more$;
- the capacity in persons to carry: 2, 4 or $more$;
- the luggage boot size: $small$, med or big;
- the estimated safety: low, med or $high$.

According to these features, the car models are classified into 4 categories, from the worst one to the best one: $unacc$ (unacceptable), acc (acceptable), $good$, $vgood$ (very good). There are 1728 car descriptions with the following biased distribution for the solution in the initial dataset: 1210 $unacc$, 384 acc, 69 $good$, and 65 $vgood$. To test now the approaches on an unbiased distribution of the solutions, only the maximum possible number of cases with the same number of cases in each category has been kept, so 65 cases per category, forming a dataset of 260 cases.

In this dataset, none of the features is Boolean. Each feature has exactly 1 value belonging to a set of more than 2 possible values. Moreover, the solution takes also its value in a set of more than 2 values. A specific transformation is required to encode these non Boolean values into Boolean ones, and a specific rule filtering must be applied to ensure that the result is one category value. These two adjustments for applying the Boolean approach are now detailed.

Encoding non Boolean attributes into Boolean ones using one-hot encoding. Let a be an attribute and let v_a be a value for a. In addition to attributes which are already described by Boolean values ($v_a \in \{0, 1\}$), nominal values can also be considered. For an attribute a with nominal values, i.e. $v_a \in \{a_1, a_2, \ldots, a_n\}$, n Boolean attributes can be used to encode each of these n possible values. The value $v_a = a_i$ is encoded with 1 on the i^{th} attribute, all the other of the n attributes being set to 0. For example, for the buying price, the possible values are $vhigh$, $high$, med or low. So, 4 Boolean variables are used to encode the initial value, with only one of these variables set to 1. For example, a low buying price is encoded by 1000, med by 0100, $high$ by 0010, and $vhigh$ by 0001. This encoding process is used on the 6 features describing the problem, as well as for encoding the solution. The illustration below shows an example of the complete problem/solution encoding for the initial record/case $((vhigh, med, 5more, 4, big, high), vgood)$.

$$\underbrace{0\ 0\ 0\ 1}_{1}\ \underbrace{0\ 0\ 1\ 0}_{2}\ \underbrace{0\ 0\ 0\ 1}_{3}\ \underbrace{0\ 1\ 0}_{4}\ \underbrace{0\ 0\ 1}_{5}\ \underbrace{0\ 0\ 1}_{6} \rightarrow \underbrace{0\ 0\ 0\ 1}_{7}$$

with 1, 2, 3, 4, 5 and 6 respectively encoding the buying price ($vhigh$), the maintenance price (med), the number of doors ($5more$), the number of persons to carry (4), the luggage boot size (big), the safety ($high$), and 7 encodes the *solution* category of the car ($vgood$).

[7] `https://archive.ics.uci.edu/ml/datasets/Car+Evaluation`

nAR	100					200					300					400				
minPS	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
	prec																			
NN	.67	.67	.68	.68	.67	.67	.67	.67	.67	.68	.67	.66	.67	.67	.68	.67	.67	.67	.67	.67
AK^+	.71	.71	.71	.72	.72	.70	.72	.71	.73	.72	.72	.72	.72	.72	.71	.73	.72	.72	.72	.72
$AK^\pm$	**.81**	**.81**	**.80**	**.80**	**.81**	**.81**	**.80**	**.80**	**.80**	**.81**	**.79**	**.80**	**.80**	**.80**	**.80**	**.80**	**.80**	**.80**	**.80**	**.81**
	car																			
NN	.67	.67	.68	.68	**.67**	.67	.67	.67	.67	.68	.67	.66	.67	.67	.68	.67	.67	.67	.67	.67
AK^+	.66	.66	.67	.67	.59	.68	.69	.68	.69	.69	.68	.69	.70	.69	.68	.70	.70	.69	.70	.69
$AK^\pm$	**.73**	**.73**	**.73**	**.73**	.64	**.77**	**.75**	**.75**	**.74**	**.77**	**.75**	**.76**	**.77**	**.76**	**.77**	**.77**	**.78**	**.76**	**.76**	**.77**

Table 7. prec and car of the three approaches for different numbers of rules and different minimal problem sizes for the Car benchmark.

Generating a negative case when $n \geq 2$ for $\mathcal{S} = \mathbb{B}^n$. For a correct solution $\mathrm{y} \in \mathbb{B}$, y is transformed in a incorrect solution y' simply by $\mathrm{y}' = \neg\mathrm{y}$. However, when a solution is encoded on more than one variable, (e.g. for this dataset, on 4 variables), it requires a specific approach to transform a positive case into a negative one. For the experiment with $n \geq 2$, but with the assumption that only 1 variable of the solution is set to 1, an incorrect solution is randomly selected among the solutions encoding syntactically correct solutions, i.e. a solution with only 1 variable set to 1 and which is not equal to the correct solution. For example, if the correct solution is 0001, the possible incorrect solutions are 0010, 0100 and 1000. An incorrect solution is randomly chosen among these 3 possibilities.

Filtering out rules that incompletely describe the solution variation. When a solution is encoded on more than one variables, (e.g. for this dataset, on 4 variables), obtaining a result which correspond to 1 of the possible values as solution requires rules containing information on all the solution variables. For example, a rule like $\{x_1^{\bar{1}}, x_2^{+}, \mathrm{y}_1^{+}\}$, containing only 1 solution variable (y_1), applied to a source case solution 0010 produces 1010 which does not correspond to a possible correct answer. So, for this dataset, only rules containing the 4 variables of the solution are kept.

Table 7 presents the results on the Car benchmark with $|\mathtt{CB}^+| = |\mathtt{CB}^-| = 26$, i.e. $\frac{|\mathtt{CB}|}{10}$ instead of $\frac{|\mathtt{CB}|}{4}$ for the other experiments. The idea is to examine, in addition, the behavior of the approach on a smallest learning case base. One more time, $AK^\pm$ improves the results w.r.t. NN, by an average of $+11\%$ in prec. Moreover, with this benchmark, $AK^\pm$ improves also well the car by an average of $+10\%$.

4.6 Results and discussion

The different experiments, on different benchmarks, with various dimensions of the problem space (from 9 for the Tic Tac Toe benchmark to 22 for the cardiac diagnosis

benchmark) has shown the efficiency of using negative examples for acquiring adaptation rules. The results are similar for the 4 benchmarks and, in any situation, $AK^{\pm}$ is better than AK^{+} and NN for precision, and most of the time for the `car` measure.

So, exploiting negative cases improves the CBR system results. For the `prec` measure, approaches based on AK built only on positive examples gives most of the time better results than the NN baseline approach, when the number and the length of rules are sufficient to avoid the use of too general rules. However, when introducing the exploitation of negative cases, the `prec` measure really increases, with various improvement: around $+7\%$ for the Congressional Voting, Tic Tac Toe and Cardiac diagnosis benchmarks, to even around $+14\%$ for the Car Evaluation benchmark. But the most important fact is that the results shows that from the precision point of view, $AK^{\pm}$ has given *always* better results than AK^{+} and NN. For the `car` measure, AK^{+} and $AK^{\pm}$ give results sometimes under the ones of the NN approach. However, these `car` scores have to be considered in regards to the `prec` score because increasing the precision has most of the time a negative impact on the number of answers (i.e. the system answers better but less often), and so, on the `car` measure. For the Congressional Votes, $AK^{\pm}$ `car` measure is almost always better than for NN and, in that case, also with a significant improvement, especially when `minPS` and `nAR` are high. Combining with a significant improvement of the `prec` measure as well, the results can be considered as excellent. For the Cardiac diagnosis, the `car` measure of $AK^{\pm}$ and NN are quite similar and for the Car evaluation, the `car` measure overcomes all the time, with an improvement around $+11\%$, the `car` of NN. In conclusion, the results of the approach based on negative cases must be highlighted, because of the increasing of the precision without having an impact on the decreasing of the `car`, which is most of the time higher than for NN.

5 Conclusion

This paper shows the benefits of exploiting negative cases in addition to positive ones to extract, using frequent closed itemsets, adaptation rules of higher quality which improves the results of a CBR system. A methodology has also been presented to transform a case description originally not encoded by Boolean attributes into a Boolean encoding, and to filter adaptation rules. The rule filtering can be based on the support of the rule (i.e. how many times this rule is found in the learning dataset), on the length of the rule (i.e. the more variables the rule uses, the best it is). Applying this methodology on 4 benchmarks with different characteristics shows that similar results are obtained for these applications, which argue for the interest of exploiting negative cases.

The quality of the results depends on numerous parameters. In this work, the impact of the number of rules as well as the size of the rules has been examined. However, it could be interesting to study the impact of other parameters, like, for example, the size of $\mathtt{CB}_L$, $\mathtt{CB}^{+}$ and $\mathtt{CB}^{-}$.

Finally, another interesting future work is to use more finely the negative cases. In this work, the negative cases are used to filter out adaptation rules, usually with a great support value but which are sometimes too general. However, like in the version space model, instead of simply removing a rule which could be useful in some problem solv-

ing situations, we could imagine to specialize this rule, to avoid its use only in a given context. For example, an approach based of FCI on variations between positive and negative cases could bring out elements which could be exploited to refine adaptation rules built only on variations between positive cases.

References

1. Badra, F., Cordier, A., Lieber, J.: Opportunistic Adaptation Knowledge Discovery. In McGinty, L., Wilson, D.C., eds.: 8th International Conference on Case-Based Reasoning - ICCBR 2009. Volume 5650 of Lecture Notes in Computer Science., Seattle, États-Unis, Springer (2009) 60–74
2. Berasaluce, S., Laurenço, C., Napoli, A., Niel, G.: An Experiment on Mining Chemical Reaction Databases. In Le Thi, H.A., Dinh, T.P., eds.: Modelling, Computation and Optimization in Information Systems and Management Sciences - MCO'04, Metz, France, Hermes Science Publishing, London (2004) 535–542
3. d'Aquin, M., Badra, F., Lafrogne, S., Lieber, J., Napoli, A., Szathmary, L.: Case base mining for adaptation knowledge acquisition. In Veloso, M.M., ed.: Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI'07), Morgan Kaufmann, Inc. (2007) 750–755
4. Dufour-Lussier, V., Lieber, J., Nauer, E., Toussaint, Y.: Improving case retrieval by enrichment of the domain ontology. In: 19th International Conference on Case Based Reasoning - ICCBR'2011, London, United Kingdom (2011)
5. Gaillard, E., Lieber, J., Nauer, E.: Adaptation knowledge discovery for cooking using closed itemset extraction. In: The Eighth International Conference on Concept Lattices and their Applications - CLA 2011, Nancy, France (2011)
6. Ganter, B., Kuznetsov, S.O.: Hypotheses and version spaces. In Ganter, B., de Moor, A., Lex, W., eds.: Conceptual Structures for Knowledge Creation and Communication, Berlin, Heidelberg, Springer Berlin Heidelberg (2003) 83–95
7. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer (1999)
8. Gillard, T., Lieber, J., Nauer, E.: Improving Adaptation Knowledge Discovery by Exploiting Negative Cases: First Experiment in a Boolean Setting. In: ICCBR 2018 - 26th International Conference on Case-Based Reasoning, Stockholm, Sweden (July 2018)
9. Hanney, K., Keane, M.T.: Learning adaptation rules from a case-base. In Smith, I., Faltings, B., eds.: Advances in Case-Based Reasoning – Third Eur. Workshop, EWCBR'96. LNAI 1168, Springer Verlag, Berlin (1996) 179–192
10. Jalali, V., Leake, D.B.: CBR Meets Big Data: A Case Study of Large-Scale Adaptation Rule Generation. In: ICCBR. (2015)
11. Kuznetsov, S.O.: Complexity of learning in concept lattices from positive and negative examples. Discrete Applied Mathematics **142**(1) (2004) 111 – 125
12. Mitchell, T.: Version Space: An Approach to Concept Learning,. PhD thesis, Stanford University (1978)
13. Richter, M.M., Weber, R.O.: Case-based reasoning, a textbook. Springer (2013)
14. Szathmary, L., Napoli, A.: CORON: A Framework for Levelwise Itemset Mining Algorithms. Supplementary Proc. of The Third International Conference on Formal Concept Analysis (ICFCA '05), Lens, France (2005) 110–113