Adaptation knowledge discovery for cooking using closed itemset extraction

Emmanuelle Gaillard, Jean Lieber, and Emmanuel Nauer

LORIA (UMR 7503—CNRS, INRIA, Nancy University)
BP 239, 54506 Vandœuvre-lès-Nancy, France,
First-Name.Last-Name@loria.fr

Abstract. This paper is about the adaptation knowledge (AK) discovery for the Taaable system, a case-based reasoning system that adapts cooking recipes to user constraints. The AK comes from the interpretation of closed itemsets (CIs) whose items correspond to the ingredients that have to be removed, kept, or added. An original approach is proposed for building the context on which CI extraction is performed. This approach focuses on a restrictive selection of objects and on a specific ranking based on the form of the CIs. Several experimentations are proposed in order to improve the quality of the AK being extracted and to decrease the computation time. This chain of experiments can be seen as an iterative knowledge discovery process: the analysis following each experiment leads to a more sophisticated experiment until some concrete and useful results are obtained.

Keywords: adaptation knowledge discovery, closed itemset, data preprocessing, case-based reasoning, cooking.

1 Introduction

This paper addresses the adaptation challenge proposed by the *Computer Cooking Contest* (http://computercookingcontest.net/) which consists in adapting a given cooking recipe to specific constraints. For example, the user wants to adapt a strawberry pie recipe, because she has no strawberry. The underlying question is: which ingredient(s) will the strawberries be replaced with?

Adapting a recipe by substituting some ingredients by others requires cooking knowledge and adaptation knowledge in particular. Taaable, a case-based reasoning (CBR) system, addresses this problem using an ingredient ontology. This ontology is used for searching which is/are the closest ingredient(s) to the one that has to be replaced. In this approach the notion of "being close to" is given by the distance between ingredients in the ontology. In the previous example, Taaable proposes to replace the strawberries by other berries (e.g. raspberries, blueberries, etc.). However, this approach is limited because two ingredients which are close in the ontology are not necessarily interchangeable and because introducing a new ingredient in a recipe may be incompatible with some other ingredient(s) of the recipe or may required to add other ingredients.

© Amedeo Napoli, Vilem Vychodil (Eds.): CLA 2011, pp. 87–99,
ISBN 978–2–905267–78–8, INRIA Nancy – Grand Est and LORIA, 2011.

This paper extends the approach proposed in [2] for extracting this kind of adaptation knowledge (AK). The approach is based on closed itemset (CI) extraction, in which items are the ingredients that have to be removed, kept, or added for adapting the recipe. This paper introduces two originalities. The first one concerns the way the binary context, on which the CI extraction is performed, is built, by focusing on a restrictive selection of objects according to the objectives of the knowledge discovery process. The second one concerns the way the CIs are filtered and ranked, according to their form. The paper is organised as follows: Section 2 specifies the problem in its whole context and introduces Taaable which will integrate the discovered AK in its reasoning process. Section 3 gives preliminaries for this work, introducing CI extraction, case-based reasoning, and related work. Section 4 explains our approach; several experiments and evaluations are described and discussed.

2 Context and motivations

2.1 Taaable

The *Computer Cooking Contest* is an international contest that aims at comparing systems that make inferences about cooking. A candidate system has to use the recipe base given by the contest to propose a recipe matching the user query. This query is a set of constraints such as inclusion or rejection of ingredients, the type or the origin of the dish, and the compatibility with some diets (vegetarian, nut-free, etc.).

Taaable [1] is a system that has been originally designed as a candidate of the *Computer Cooking Contest*. It is also used as a brain teaser for research in knowledge based systems, including knowledge discovery, ontology engineering, and CBR. Like many CBR systems, Taaable uses an ontology to retrieve recipes that are the most similar to the query. Taaable retrieves and creates cooking recipes by adaptation. If there exist recipes exactly matching the query, they are returned to the user; otherwise the system is able to retrieve similar recipes (i.e. recipes that partially match the target query) and adapts these recipes, creating new ones. Searching similar recipes is guided by several ontologies, i.e. hierarchies of classes (ingredient hierarchy, dish type hierarchy and dish origin hierarchy), in order to relax constraints by generalising the user query. The goal is to find the most specific generalisation of the query (the one with the minimal cost) for which recipes exist in the recipe base. Adaptation consists in substituting some ingredients of the retrieved recipes by the ones required by the query.

Taaable retrieves recipes using query generalisation, then adapts them by substitution. This section gives a simplified description of the Taaable system. For more details about the Taaable inference engine, see e.g. [1]. For example, for adapting the *"My Strawberry Pie"* recipe to the `no Strawberry` constraint, the system first generalises `Strawberry` into `Berry`, then specialises `Berry` into, say, `Raspberry`.

2.2 Domain ontology

An ontology $\mathcal{O}$ defines the main classes and relations relevant to cooking. $\mathcal{O}$ is a set of atomic classes organised into several hierarchies (ingredient, dish type, dish origin, etc.). Given two classes `B` and `A` of this ontology, `A` is subsumed by `B`, denoted by $\texttt{B} \sqsupseteq \texttt{A}$, if the set of instances of `A` is included in the set of instances of `B`. For instance, $\texttt{Berry} \sqsupseteq \texttt{Blueberry}$ and $\texttt{Berry} \sqsupseteq \texttt{Raspberry}$.

2.3 Taaable adaptation principle

Let R be a recipe and Q be a query such that R does not exactly match Q (otherwise, no adaptation would be needed). For example, $Q = \texttt{no Strawberry}$ and $R =$ *"My Strawberry Pie"*.The basic ontology-driven adaptation in TAAABLE follows the generalisation/specialisation principle explained hereafter (in a simplified way). First, R is generalised (in a minimal way) into $\Gamma(R)$ that matches Q. For example, Γ may be the substitution $\texttt{Strawberry} \rightsquigarrow \texttt{Berry}$. Second, $\Gamma(R)$ is specialised into $\Sigma(\Gamma(R))$ that still matches Q. For example, Σ is the substitution $\texttt{Berry} \rightsquigarrow \texttt{Raspberry}$ (the class `Berry` is too abstract for a recipe and must be made precise). This adaptation approach has at least two limits. First, the choice of Σ is at random: there is no reason to choose raspberries instead of blueberries, unless additional knowledge is given. Second, when such a substitution of ingredient is made, it may occur that some ingredients should be added or removed from R. These limits point out the usefulness of additional knowledge for adaptation.

3 Preliminaries

3.1 Itemset extraction

Itemset extraction is a set of data-mining methods for extracting regularities into data, by aggregating object items appearing together. Like FCA [8], itemset extraction algorithms start from a *formal context* K, defined by $K = (G, M, r)$, where G is a set of objects, M is a set of items, and r is the relation on $G \times M$ stating that an object is described by an item [8]. Table 1 shows an example of context, in which recipes are described by the ingredients they require: G is a set of 5 objects (recipes R, R_1, R_2, R_3, and R_4), M is a set of 7 items (ingredients `Sugar`, `Water`, `Strawberry`, etc.).

An *itemset* I is a set of items, and the *support* of I, $support(I)$, is the number of objects of the formal context having every item of I. I is frequent, with respect to a threshold σ, whenever $support(I) \geq \sigma$. I is closed if it has no proper superset J ($I \subsetneq J$) with the same support. For example, $\{\texttt{Sugar}, \texttt{Raspberry}\}$ is an itemset and $support(\{\texttt{Sugar}, \texttt{Raspberry}\}) = 2$ because 2 recipes require both `Sugar` and `Raspberry`. However, $\{\texttt{Sugar}, \texttt{Raspberry}\}$ is not a closed itemset, because $\{\texttt{Sugar}, \texttt{PieCrust}, \texttt{Raspberry}\}$ has the same support. Another, equivalent, definition of closed itemsets can be given on the basis of a closure operator $\cdot''$ defined as follows. Let I be an itemset and I' be the set of objects that have all the items

	Sugar	Water	Strawberry	PieCrust	Cornstarch	CoolWhip	Raspberry	Gelatin	Apple	AppleJuice	Cinnamon	PieShell
R	×	×	×	×	×	×						
R_1	×			×	×		×	×				
R_2	×	×		×			×					
R_3	×			×	×				×	×		
R_4	×	×							×		×	×

Table 1. Formal context representing ingredients used in recipes.

of I: $I' = \{x \in G \mid \forall i \in I, x\ r\ i\}$. In a dual way, let X be a set of objects and X' be the set of properties shared by all objects of X: $X' = \{i \in M \mid \forall x \in X, x\ r\ i\}$. This defines two operators: $\cdot' : I \in 2^M \mapsto I' \in 2^G$ and $\cdot' : X \in 2^G \mapsto X' \in 2^M$. These operators can be composed in an operator $\cdot'' : I \in 2^M \mapsto I'' \in 2^M$. An itemset I is said to be closed if it is a fixed point of $\cdot''$, i.e., $I'' = I$.

In the following, "CIs" stands for closed itemsets, and "FCIs" stands for frequent CIs. For $\sigma = 3$, the FCIs of this context are $\{\texttt{Sugar}, \texttt{PieCrust}\}$, $\{\texttt{Sugar}, \texttt{PieCrust}, \texttt{Cornstarch}\}$, $\{\texttt{Sugar}, \texttt{Water}\}$, $\{\texttt{Sugar}\}$, $\{\texttt{Water}\}$, $\{\texttt{PieCrust}\}$, and $\{\texttt{Cornstarch}\}$.

For the following experiments, the Charm algorithm [12] that efficiently computes the FCIs is used thanks to Coron a software platform implementing a rich set of algorithmic methods for symbolic data mining [11].

3.2 Case-based reasoning

Case-based reasoning (CBR [10]) consists in answering queries with the help of previous experience units called cases. In Taaable, a case is a recipe and a query represents user constraints. In many systems, including Taaable, CBR consists in the retrieval of a case from the case base and in the adaptation of the retrieved case in an adapted case that solves the query. Retrieval in Taaable is performed by minimal generalisation of the query (cf. section 2.3). Adaptation can be a simple substitution (e.g., substitute strawberry with any berry) but it can be improved thanks to the use of some domain specific AK. This motivates the research on AK acquisition.

3.3 Related work

The AK may be acquired in various way. It may be collected from experts [6], it may be acquired using machine learning techniques [9], or be semi-automatic, using data-mining techniques and knowledge discovery principles [3,4].

This paper addresses automatic AK discovery. Previous works, such as the ones proposed by d'Aquin et al. with the Kasimir project in the medical do-

main [5], and by Badra et al. in the context of a previous work on TAAABLE [2], are the foundations of our work.

KASIMIR is a CBR system applied to decision support for breast cancer treatment. In KASIMIR, a case is a treatment used for a given patient. The patient is described by characteristics (age, tumour size and location, etc.) and the treatment consists in applying medical instructions. In order to discover AK, cases that are *similar* to the target case are first selected. Then, FCIs are computed on the variations between the target case and the similar cases. FCIs matching a specific form are interpreted for generating AK [5].

Badra et al. use this approach to make cooking adaptations in TAAABLE [2]. Their work aims at comparing pairs of recipes depending on the ingredients they contain. A recipe R is represented by the set of its ingredients: $\texttt{Ingredients}(R)$. For example, the recipe *"My Strawberry Pie"* is represented by

$$\texttt{Ingredients}(\text{\textit{"My Strawberry Pie"}}) = \{\texttt{Sugar}, \texttt{Water}, \texttt{Strawberry}, \texttt{PieCrust}, \texttt{Cornstarch}, \texttt{CoolWhip}\}$$

Let (R, R') be a pair of recipes which is selected. According to [2], the representation of a pair is denoted by Δ, where Δ represents the variation of ingredients between R and R'. Each ingredient ing is marked by $-$, $=$, or $+$:

- $ing^- \in \Delta$ if $ing \in \texttt{Ingredients}(R)$ and $ing \notin \texttt{Ingredients}(R')$, meaning that ing appears in R but not in R'.
- $ing^+ \in \Delta$ if $ing \notin \texttt{Ingredients}(R)$ and $ing \in \texttt{Ingredients}(R')$, meaning that ing appears in R' but not in R.
- $ing^= \in \Delta$ if $ing \in \texttt{Ingredients}(R)$ and $ing \in \texttt{Ingredients}(R')$, meaning that ing appears both in R in R'.

Building a formal context about ingredient variations in cooking recipes. Suppose we want to compare the recipe R with the four recipes (R_1, R_2, R_3, R_4) given in Table 1. Variations between $R =$ *"My Strawberry Pie"* and a recipe R_i have the form $\bigwedge_j ing_{i,j}^{mark}$. For example:

$$\Delta_{R,R_1} = \texttt{Sugar}^= \wedge \texttt{Water}^- \wedge \texttt{Strawberry}^- \wedge \texttt{PieCrust}^= \wedge \texttt{Cornstarch}^= \wedge \texttt{CoolWhip}^- \wedge \texttt{Raspberry}^+ \wedge \texttt{Gelatin}^+ \quad (1)$$

According to these variations, a formal context $K = (G, M, I)$ can be built (cf. Table 2, for the running example):

- $G = \{\Delta_{R,R_i}\}_i$
- M is the set of ingredient variations: $M = \{ing_{i,j}^{mark}\}_{i,j}$. In particular, M contains all the conjuncts of Δ_{R,R_1} ($\texttt{Strawberry}^-$, etc., cf.(1)).
- $(g, m) \in I$, if $g \in G$, $m \in M$, and m is a conjunct of g, for example $(\Delta_{R,R_1}, \texttt{Strawberry}^-) \in I$.

	Sugar$^{=}$	Water$^{=}$	Water^{-}	Strawberry^{-}	PieCrust$^{=}$	PieCrust^{-}	Cornstarch$^{=}$	Cornstarch^{-}	CoolWhip^{-}	Raspberry^{+}	Gelatin^{+}	Apple^{+}	AppleJuice^{+}	Cinnamon^{+}	PieShell^{+}
Δ_{R,R_1}	×		×	×	×		×		×	×	×				
Δ_{R,R_2}	×	×		×	×			×	×	×					
Δ_{R,R_3}	×		×	×	×		×		×			×	×		
Δ_{R,R_4}	×	×		×		×		×	×			×		×	×

Table 2. Formal context for ingredient variations in pairs of recipes (R, R_j).

Interpretation. In the formal context, an ingredient marked with + (resp. −) is an ingredient that has to be added (resp. removed). An ingredient marked with = is an ingredient common to R and R_i.

4 Adaptation Knowledge discovery

AK discovery is based on the same scheme as knowledge discovery in databases (KDD [7]). The main steps of the KDD process are data preparation, data-mining, and interpretation of the extracted units of information. Data preparation relies on formatting data for being used by data-mining tools and on filtering operations for focusing on special subsets of objects and/or items, according to the objectives of KDD. Data-mining tools are applied for extracting regularities into the data. These regularities have then to be interpreted; filtering operations may also be performed on this step because of the (often) huge size of the data-mining results or of the noise included in these results. All the steps are guided by an analyst.

The objective of our work is the extraction of some AK useful for adapting a given recipe to a query. The work presented in the following focuses on filtering operations, in order to extract from a formal context encoding ingredient variations between pairs of recipes, the cooking adaptations. The database used as entry point of the process is the Recipe Source database (`http://www.recipesource.com/`) which contains 73795 cooking recipes. For the sake of simplicity, we consider in the following, the problem of adapting R by substituting one or several ingredient(s) with one or several ingredient(s) (but the approach can be generalised for removing more ingredients, and also be used for adding ingredient(s) in a recipe). Three experiments are presented; they address the same adaptation problem: adapting the $R =$ *"My Strawberry Pie"* recipe, with `Ingredients`(*"My Strawberry Pie"*) = {`Sugar`, `Water`, `Strawberry`, `PieCrust`, `Cornstarch`, `CoolWhip`}, to the query `no Strawberry`. In each experiment, a formal context about ingredient variations in recipes is built. Then, FCIs are extracted and filtered for proposing cooking adaptation. The two first

experiments focus on object filtering, selecting recipes which are more and more similar to the *"My Strawberry Pie"* recipe: the first experiment uses recipe from the same type (i.e. pie dish) as *"My Strawberry Pie"* instead of choosing recipes of any type; the second experiment focuses on a more precise filtering based on similarity between the *"My Strawberry Pie"* recipe and recipes used for generating the formal context on ingredient variations.

4.1 A first approach with closed itemsets

As introduced in [2], a formal context is defined, where objects are ordered pairs of recipes (R, R') and properties are ingredients marked with $+$, $=$, $-$ for representing the ingredient variations from R to R'. The formal context which is build is similar to the example given in Table 2. In each pair of recipes, the first element is the recipe $R =$*"My Strawberry Pie"* that must be adapted; the second element is a recipe of the same dish type as R which, moreover, does not contain the ingredient which has to be removed. In our example, it corresponds to *pie dish* recipes which do not contain *strawberry*. This formal context allows to build CIs which have to be interpreted in order to acquire adaptation rules.

Experiment. 3653 pie dish recipes that do not contain strawberry are found in the Recipe Source database. The formal context, with 3653 objects $\times$ 1355 items produces 107,837 CIs (no minimal support is used).

Analysis. Some interesting CIs can be found. For example, $\{\texttt{PieCrust}^-, \texttt{Strawberry}^-, \texttt{Cornstarch}^-, \texttt{CoolWhip}^-, \texttt{Water}^-, \texttt{Sugar}^-\}$ with support of 1657, contains all the ingredients of R with a $-$ mark, meaning that there are 1657 recipes which have no common ingredients with the R recipe. In the same way, $\{\texttt{PieCrust}^-, \texttt{Strawberry}^-, \texttt{Cornstarch}^-, \texttt{CoolWhip}^-, \texttt{Water}^-\}$ with support 2590, means that 2590 recipes share only the `Sugar` ingredient with R because the sugar is the sole ingredient of R which is not included in this CI. The same analysis can be done for $\{\texttt{PieCrust}^-, \texttt{Strawberry}^-, \texttt{Cornstarch}^-, \texttt{CoolWhip}^-, \texttt{Sugar}^-\}$ (support of 1900), for water, etc.

Conclusion. The CIs are too numerous for being presented to the analyst. Only 1996 of the 3653 pie dish without strawberry recipes share at least one ingredient with R. There are too many recipes without anything in common. A first filter can be used to limit the size of the formal context in number of objects.

4.2 Filtering recipes with at least one common ingredient

Experiment. The formal context, with 1996 objects $\times$ 813 items, produces 22,408 CIs (no minimal support is used), ranked by decreasing support.

Results. The top five FCIs are:

- $\{\texttt{Strawberry}^-\}$ with support of 1996;
- $\{\texttt{Strawberry}^-, \texttt{CoolWhip}^-\}$ with support of 1916;
- $\{\texttt{Strawberry}^-, \texttt{PieCrust}^-\}$ with support of 1757;
- $\{\texttt{Strawberry}^-, \texttt{PieCrust}^-, \texttt{CoolWhip}^-\}$ with support of 1679;
- $\{\texttt{Strawberry}^-, \texttt{Cornstarch}^-\}$ with support of 1631.

Analysis. Several observations can be made. The first FCI containing an ingredient marked by + ($\{\texttt{Strawberry}^-, \texttt{Egg}^+\}$, with support of 849) appears only at the 46th position. Moreover, there are 45 FCIs with one ingredient marked by + in the first 100 FCIs, and no FCI with more than one ingredient marked by +. A substituting ingredient *ing* can only be found in CIs containing ing^+ meaning that there exists a recipe containing *ing*, which is not in R. So, FCIs that do not contain the + mark cannot be used for finding a substitution proposition, and they are numerous in the first 100 ones, based on a support ranking (recall that it has been chosen not to consider adaptation by simply removing ingredient).

In the first 100 FCIs, there is only 15 FCIs containing both an ingredient marked by + and an ingredient marked by =. In a FCI I, the = mark on a ingredient *ing* means that *ing* is common to R and to recipe(s) involved by the creation of I. So, an ingredient marked by = guarantees a certain similarity (based on ingredients that are used) between the recipes R and R' compared by $\Delta_{R,R'}$. If a FCI I contains a potential substituting ingredient, marked by +, but does not contain any =, the risk for proposing a cooking adaptation from I is very high, because there is no common ingredient with R in the recipe the potential substituting ingredient comes from.

In the first 100 recipes, the only potential substituting ingredients (so, the ingredients marked by +) are egg, salt, and butter, which are not satisfactory from a cooking viewpoint for substituting the strawberries.

We have conducted similar experiments with other R and queries, and the same observations as above can be made.

Conclusion. From these observations, it can be concluded that the sole ranking based on support is not efficient to find relevant cooking adaptation rules, because the most frequent CIs do no contain potential substituting ingredients and, moreover, have no common ingredient with R.

4.3 Filtering and ranking CIs according to their forms

To extract realistic adaptation, CIs with a maximum of ingredients marked by = are searched. We consider that a substitution is acceptable, if 50% of ingredients of R are preserved and if the adaptation does not introduce too many ingredients; we also limit the number of ingredients introduced to 50% of the initial number of ingredients in R. For the experiment with the R = *"My Strawberry Pie"*, containing initially 6 ingredients, it means that at least 3 ingredients must be preserved and at most 3 ingredients can be added. In term of CIs, it corresponds to CIs containing at least 3 ingredients marked with = and at most 3 ingredients marked with +.

Experiment. Using this filter on CIs produced by the previous experiment reduces the number of CIs to 505. However, because some CIs are more relevant than others, they must be ranked according to several criteria. We use the following rules, given by priority order:

1. A CI must have a + in order to find a potential substituting ingredient.
2. A CI which has more = than another one is more relevant. This criterion promotes the pairs which have a largest set of common ingredient.
3. A CI which has less − than another one is more relevant. This criterion promotes adaptations which remove less ingredients.
4. A CI which has less + than another one is more relevant. This criterion promotes adaptations which add less ingredients.
5. If two CIs cannot be ranked according to the 4 criteria above, the CI the more frequent is considered to be the more relevant.

Results. The 5 first CIs ranked according to the previous criteria are:

- $\{\texttt{Water}^{=}, \texttt{Sugar}^{=}, \texttt{Strawberry}^{-}, \texttt{CoolWhip}^{-}, \texttt{Cornstarch}^{=}, \texttt{PieCrust}^{=}, \texttt{Salt}^{+}\}$ with support of 5;
- $\{\texttt{Water}^{=}, \texttt{Sugar}^{=}, \texttt{Strawberry}^{-}, \texttt{CoolWhip}^{-}, \texttt{Cornstarch}^{=}, \texttt{PieCrust}^{=}, \texttt{LemonJuice}^{+}\}$ with support of 4;
- $\{\texttt{Water}^{=}, \texttt{Sugar}^{=}, \texttt{Strawberry}^{-}, \texttt{CoolWhip}^{-}, \texttt{Cornstarch}^{=}, \texttt{PieCrust}^{=}, \texttt{LemonJuice}^{+}, \texttt{CreamCheese}^{+}\}$ with support of 2;
- $\{\texttt{Water}^{=}, \texttt{Sugar}^{=}, \texttt{Strawberry}^{-}, \texttt{CoolWhip}^{-}, \texttt{Cornstarch}^{=}, \texttt{PieCrust}^{=}, \texttt{LemonJuice}^{+}, \texttt{WhippingCream}^{+}\}$ with support of 2;
- $\{\texttt{Water}^{=}, \texttt{Sugar}^{=}, \texttt{Strawberry}^{-}, \texttt{CoolWhip}^{-}, \texttt{Cornstarch}^{=}, \texttt{PieCrust}^{=}, \texttt{LemonJuice}^{+}, \texttt{LemonPeel}^{+}\}$ with support of 2.

Analysis. One can observe that potential substituting ingredients take part of the first 5 CIs and each CIs preserve 4 (of 6) ingredients. The low supports of these CIs confirm that searching frequent CIs is not compatible with our need, which is to extract CIs with a specific form.

Conclusion. Ranking the CIs according to our particular criteria is more efficient than using a support based ranking. This kind of ranking can also be seen as a filter on CIs. However, this approach requires to compute all CIs because the support of interesting CIs is low.

4.4 More restrictive formal context building according to the form of interesting CIs

The computation time can be improved by applying a more restrictive selection of recipe pairs at the formal context building step, decreasing drastically the size of the formal context. Indeed, as the expected form of CIs is known, recipe pairs that cannot produce CIs of the expected form can be removed. This can also be seen as a selection of recipes that are similar enough to R. R' is considered

as enough similar to R if R' has a minimal threshold $\sigma^= = 50\%$ of ingredients in common with R (cf. (2)) and if R' has a maximal threshold $\sigma^+ = 50\%$ of ingredients that are not used in R (cf. (3)). These two conditions expresses for $\Delta_{R,R'}$ the same similarities conditions considered in section 4.3 on CIs.

$$\frac{|\texttt{Ingredients}(R) \cap \texttt{Ingredients}(R')|}{|\texttt{Ingredients}(R)|} \geq \sigma^= \tag{2}$$

$$\frac{|\texttt{Ingredients}(R') \setminus \texttt{Ingredients}(R)|}{|\texttt{Ingredients}(R)|} \geq \sigma^+ \tag{3}$$

Experiment. Among the 1996 pie dish recipes not containing `Strawberry`, only 20 recipes satisfy the two conditions. The formal context, with 20 objects $\times$ 40 items, produces only 21 CIs (no minimal support is used).

Results. The 5 first CIs, satisfying the form introduced in the previous section and ranked by decreasing support are:

- $\{\texttt{Water}^=, \texttt{Sugar}^=, \texttt{Cornstarch}^=, \texttt{PieCrust}^=, \texttt{Strawberry}^-, \texttt{CoolWhip}^-, \texttt{RedFoodColoring}^+, \texttt{Cherry}^+\}$ with support of 1;
- $\{\texttt{Water}^=, \texttt{Sugar}^=, \texttt{Cornstarch}^=, \texttt{PieCrust}^-, \texttt{Strawberry}^-, \texttt{CoolWhip}^-, \texttt{PieShell}+\}$ with support of 6;
- $\{\texttt{Water}^=, \texttt{Sugar}^=, \texttt{Cornstarch}^=, \texttt{PieCrust}^-, \texttt{Strawberry}^-, \texttt{CoolWhip}^-, \texttt{Raspberry}^+\}$ with support of 3;
- $\{\texttt{Water}^=, \texttt{Sugar}^-, \texttt{Cornstarch}^=, \texttt{PieCrust}^=, \texttt{Strawberry}^-, \texttt{CoolWhip}^-, \texttt{Apple}^+, \texttt{AppleJuice}^+\}$ with support of 3;
- $\{\texttt{Water}^=, \texttt{Sugar}^=, \texttt{Cornstarch}^=, \texttt{PieCrust}^-, \texttt{Strawberry}^-, \texttt{CoolWhip}^-, \texttt{Peach}^+, \texttt{PieShell}+\}$ with support of 2.

Analysis. According to these CIs the first potential substituting ingredients are: `RedFoodColoring`, `Cherry`, `PieShell`, `Raspberry`, `Apple`, and `Peach`. Each CI preserves 3 or 4 (of 6) ingredients to 6 and two CIs add 2 ingredients.

Conclusion. This approach reduces the computation time without reducing the result quality. Moreover, it gives the best potential adaptation in the first CIs.

4.5 From CIs to adaptation rules

As Taaable must propose a recipe adaptation, CIs containing potentially substituting ingredients must be transformed. Indeed, a CI does not represent a direct cooking adaptation. For example, the third CI of the last experiment contains $\texttt{Raspberry}^+$, simultaneously with $\texttt{CoolWhip}^-$ and $\texttt{PieCrust}^-$. Removing the pie crust (i.e. $\texttt{PieCrust}^-$) can look surprising for a pie dish, but one must keep in mind that a CI does not correspond to a real recipe, but to an abstraction of variations between R and a set of recipes. So, producing a complete adaptation requires to get back to the Δ_{R,R_i} for having all the variations of ingredient that will take part to the adaptation. For example, for

	Water$^{=}$	Sugar$^{=}$	Cornstarch$^{=}$	PieCrust^{-}	Strawberry^{-}	CoolWhip^{-}	Raspberry^{+}	PieShell^{+}	Gelatin^{+}	FoodColor^{+}	GCPieCrust^{+}
Δ_{R,R_1}	×	×	×	×	×	×	×		×		×
Δ_{R,R_2}	×	×	×	×	×	×	×	×		×	
Δ_{R,R_3}	×	×	×	×	×	×	×	×			

Table 3. Formal context for ingredient variations in pairs of recipes (R, R_j).

the CI $\{\texttt{Water}^{=}, \texttt{Sugar}^{=}, \texttt{Cornstarch}^{=}, \texttt{PieCrust}^{-}, \texttt{Strawberry}^{-}, \texttt{CoolWhip}^{-}, \texttt{Raspberry}^{+}\}$, the Δ_{R,R_i} (with $i \in [1;3]$) are the ones given by Table 3.

The adaptation rules extracted from these 3 recipe variations are:

- $\{\texttt{CoolWhip}, \texttt{PieCrust}, \texttt{Strawberry}\} \rightsquigarrow \{\texttt{Gelatin}, \texttt{GCPieCrust}, \texttt{Raspberry}\}$;
- $\{\texttt{CoolWhip}, \texttt{PieCrust}, \texttt{Strawberry}\} \rightsquigarrow \{\texttt{FoodColor}, \texttt{PieShell}, \texttt{Raspberry}\}$;
- $\{\texttt{CoolWhip}, \texttt{PieCrust}, \texttt{Strawberry}\} \rightsquigarrow \{\texttt{PieShell}, \texttt{Raspberry}\}$.

For R_2 and R_3, `PieShell` is added in replacement of `PieCrust`; in R_1, `GCPieCrust` plays the role of `PieCrust`. These three recipe variations propose to replace `Strawberry` by `Raspberry`. For R_1 (resp. R_2), `Gelatin` (resp. `FoodColor`) is also added. Finally, the three recipe variations propose to remove the `CoolWhip`.

Our approach guarantees the ingredient compatibility, with the assumption that the recipe base used for the adaptation rule extraction process contains only *good* recipes, i.e. recipes which do not contain ingredient incompatibility. Indeed, as adaptation rules are extracted from real recipes, the good combination of ingredients is preserved. So, when introducing a new ingredient ing_1 (marked by ing_1^+), removing another ingredient ing_2 (marked by ing_2^-) could be required. The reason is that there is no recipe, entailed in the creation of the CI from which the adaptation rules are extracted, using both ing_1 and ing_2. In the same way, adding a supplementary ingredient ing_3 (marked by ing_3^+) in addition of ing_1, is obtained from recipes which use both ing_1 and ing_3.

Applying FCA on these Δ_{R,R_i} produces the concept lattice presented in Fig. 1 in which the top node is the CI retained. This node can be seen as a generic cooking adaptation, and navigating into the lattice will conduct to more specific adaptation. The KDD loop is closed: after having (1) selected and formatting the data, (2) applying a data-mining CI extraction algorithm, and (3) interpreting the results, a new set of data is selected on which a data-mining –FCA– algorithm could then be applied.

We have chosen to return the adaptation rules generated from the 5 first CIs to the user. So, the system proposes results where `Strawberry` could be replaced (in addition of some other ingredient adding or removing) by "`RedFoodColoring` and `Cherry`", by `Raspberry` with optional `Gelatin` or `FoodColor`, by `Peach`

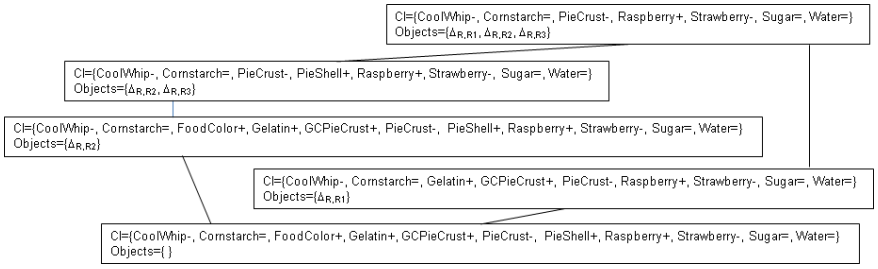

Fig. 1. The lattice computed on the formal context given in Table 3.

with optional `FoodColor` or `LemonJuice`, by "`HeavyCream` and `LemonRind`", or by "`Apple` and `AppleJuice`".

5 Conclusion

This paper shows how adaptation knowledge can be extracted efficiently for addressing a cooking adaptation challenge. Our approach focuses on CIs with a particular form, because the support is not a good ranking measure for this problem. A ranking method based on 5 criteria explicitly specified for this adaptation problem is proposed; the support is used in addition to distinguish CIs which satisfy in the same way the 5 criteria.

Beyond the application domain, this study points out that KD is not only a data-mining issue: the preparation and interpretation steps are also important. Moreover, it highlights the iterative nature of KD: starting from a first experiment with few a priori about the form of the results which are too numerous to be interpreted, it arrives to an experiment with a precise aim that gives results that are easy to interpret as adaptation rules.

It has been argued in the paper that this approach is better than the basic adaptation approach (based on substituting an ingredient by another one, on the basis of the ontology), in that it avoids some ingredient incompatibilities and makes some specialisation choices. However, a careful study remains to be made in order to compare experimentally these approaches.

A short-term future work is to integrate this AK discovery into the online system Taaable, following the principles of opportunistic KD [2].

A mid-term future work consists in using the ontology during the KD process. The idea is to add new items, deduced thanks to the ontology (e.g. the properties $\texttt{Cream}^-$ and $\texttt{Milk}^+$ entail the variation $\texttt{Dairy}^=$). First experiments have already been conducted but they raise interpretation difficulties. Indeed, the extracted CIs contain abstract terms (such as $\texttt{Dairy}^=$ or $\texttt{Flavoring}^+$) that are not easy to interpret.

References

1. F. Badra, R. Bendaoud, R. Bentebitel, P.-A. Champin, J. Cojan, A. Cordier, S. Després, S. Jean-Daubias, J. Lieber, T. Meilender, A. Mille, E. Nauer, A. Napoli, and Y. Toussaint. Taaable: Text Mining, Ontology Engineering, and Hierarchical Classification for Textual Case-Based Cooking. In *ECCBR Workshops, Workshop of the First Computer Cooking Contest*, pages 219–228, 2008.
2. F. Badra, A. Cordier, and J. Lieber. Opportunistic Adaptation Knowledge Discovery. In Lorraine McGinty and David C. Wilson, editors, *8th International Conference on Case-Based Reasoning - ICCBR 2009*, volume 5650 of *Lecture Notes in Computer Science*, pages 60–74, Seattle, États-Unis, July 2009. Springer. The original publication is available at www.springerlink.com.
3. S. Craw, N. Wiratunga, and R. C. Rowe. Learning adaptation knowledge to improve case-based reasoning. *Artificial Intelligence*, 170(16-17):1175–1192, 2006.
4. M. d'Aquin, F. Badra, S. Lafrogne, J. Lieber, A. Napoli, and L. Szathmary. Case base mining for adaptation knowledge acquisition. In *International Joint Conference on Artificial Intelligence, IJCAI'07*, pages 750–756, 2007.
5. M. D'Aquin, S. Brachais, J. Lieber, and A. Napoli. Decision Support and Knowledge Management in Oncology using Hierarchical Classification. In Katherina Kaiser, Silvia Miksch, and Samson W. Tu, editors, *Proceedings of the Symposium on Computerized Guidelines and Protocols - CGP-2004*, volume 101 of *Studies in Health Technology and Informatics*, pages 16–30, Prague, Czech Republic, 2004. Silvia Miksch and Samson W. Tu, IOS Press.
6. M. d'Aquin, J. Lieber, and A. Napoli. Adaptation Knowledge Acquisition: a Case Study for Case-Based Decision Support in Oncology. *Computational Intelligence (an International Journal)*, 22(3/4):161–176, 2006.
7. U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, pages 37–54, 1996.
8. B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, 1999.
9. K. Hanney and M. T. Keane. Learning Adaptation Rules From a Case-Base. In I. Smith and B. Faltings, editors, *Advances in Case-Based Reasoning – Third European Workshop, EWCBR'96*, LNAI 1168, pages 179–192. Springer, 1996.
10. C. K. Riesbeck and R. C. Schank. *Inside Case-Based Reasoning*. Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey, 1989.
11. L. Szathmary and A. Napoli. CORON: A Framework for Levelwise Itemset Mining Algorithms. *Supplementary Proc. of The Third International Conference on Formal Concept Analysis (ICFCA '05), Lens, France*, pages 110–113, 2005.
12. M. J. Zaki and C.-J. Hsiao. CHARM: An efficient algorithm for closed itemset mining. In *SIAM International Conference on Data Mining SDM'02*, pages 33–43, 2002.