Extracting Generic Cooking Adaptation Knowledge for the TAAABLE Case-Based Reasoning System

Emmanuelle Gaillard[123] and **Emmanuel Nauer**[123] and **Marie Lefevre**[4] and **Amélie Cordier**[4]

Abstract. This paper addresses the issue of interactive adaptation knowledge acquisition. It shows how the expert's involvement in this process can improve the quality and usefulness of the results. The approach is defended in the context of TAAABLE, a CBR system which adapts recipes to user needs. In TAAABLE, adaptation knowledge takes the form of substitutions. A datamining process allows the discovery of specific substitutions in recipes. A second process, that must be driven by an expert, is needed to generalise these substitutions to make them usable on other recipes. For that, we defend an approach based on closed itemsets (CIs) for extracting generic substitutions starting from specific ones. We focus on a restrictive selection of objects, on a specific filtering on the form of the CIs and on a specific ranking on support and stability of the CIs. Experimentations demonstrate the feasibility of our approach and show some first results.

Keywords: adaptation knowledge discovery, interactive knowledge acquisition, closed itemset, cooking.

1 Introduction

This paper addresses the issue of interactive adaptation knowledge (AK) discovery. It shows how experts can be involved in a datamining process in order to improve the overall results of a system. The approach is implemented within TAAABLE, a case-based reasoning (CBR) system which is a regular contestant of the *Computer Cooking Contest* (`http://computercookingcontest.net/`). This contest proposes to compare systems that are able to adapt cooking recipe to users constraints. For example, the user wants a pie recipe with raspberries. According to the user constraints, TAAABLE searches, in the recipe base (which is a case base), whether some recipes satisfy these constraints. Recipes, if they exist, are returned to the user; otherwise the system is able to retrieve similar recipes (i.e. recipes that match the target query partially) and to adapt these recipes, creating new ones. TAAABLE uses WIKITAAABLE, a semantic wiki in which the knowledge required by TAAABLE is stored.

[1] Université de Lorraine, LORIA, UMR 7503 — Vandœuvre-lès-Nancy, F-54506, France
[2] CNRS, LORIA, UMR 7503 — Vandœuvre-lès-Nancy, F-54506, France
[3] Inria — Villers-lès-Nancy, F-54602, France
[4] Université de Lyon, CNRS - Université Lyon 1, LIRIS, UMR 5205 - F-69622, Lyon, France

Currently, searching similar recipes and adapting them is only guided by the cooking ontology stored in WIKITAAABLE. AK is a type of knowledge a CBR systems may use to improve its results. For adapting recipes, an AK is considered as a substitution of ingredient(s) by other(s), in a given context. Two types of substitution are actually considered depending on the context in which the substitution can be applied:

- if the context of the substitution (e.g. "replace strawberries with raspberries") is a specific recipe (e.g. in the *"My Strawberry Pie"* recipe), then the AK is specific and can be reused for adapting only the specific recipe.
- if the context of the substitution (e.g. "replace butter with margarine") is a set of recipes (e.g. cake recipes), then the AK is generic and could be reused for adapting each of the recipes belonging to the set of recipes defining the context.

The main objective of this work is to define an approach to acquire generic AK. AK is required to produce fine-grained adapted recipes. In the current TAAABLE system, the adaptation process consists of two steps. First, a recipe similar to the query is retrieved. Then, the recipe is adapted to the specific constraints through a process of generalisation/specialisation. This process is arbitrary and does not take into account any specific AK. Therefore, a mid-term objective is to extend the current TAAABLE reasoning process by taking into account AK for computing better adaptation.

An approach for extracting generic ingredient substitutions based on similarity of cooking actions is studied in [17]. Our approach addresses the discover of generic AK from specific AK, using closed itemsets (CIs). This work is the continuity of a previous work on specific AK discovery [9], that has been integrated in WIKITAAABLE for improving the man-machine collaboration for acquiring AK [6].

This paper uses a classical knowledge discovery in database (KDD) process. Its originality lies in the special attention that has been given to the validation step. A dedicate interface has been built in order to interact with cooking experts for validating and adjusting AK proposed by the KDD process.

The paper is organised as follows. Section 2 introduces the context and motivation of this work. Section 3 explains our approach. Section 4 gives some results and evaluation, including a scenario for repairing generic AK. Section 5 describes the interface for acquiring AK and highlights the various features available for the expert. Section 6 discusses related work. Section 7 concludes the paper and discusses ongoing work.

2 Context and motivation

TAAABLE is a CBR system that has been designed for participating to the Computer Cooking Contest[5], an international contest which aims at comparing CBR system results on a common domain: cooking. Several challenges are proposed in this contest. Among them, two challenges (won by TAAABLE in 2010):

- the *main challenge*, asking CBR systems to return recipes satisfying a set of constraints given by the user, such as inclusion or rejection of ingredients, the type or the origin of the dish, the compatibility with some diets (vegetarian, nut-free, etc.). Systems have to search into a set of limited (approximately 1500) recipes for recipes satisfying the constraints, and if there is no recipe satisfying all the constraints, the systems have to adapt existing recipes into new ones.
- the *adaptation challenge*, asking CBR systems to adapt a given recipe to specific constraints. For example, "adapt the *My strawberry pie* recipe because I do not have strawberry".

An illustration of the TAAABLE interface is given in 1.

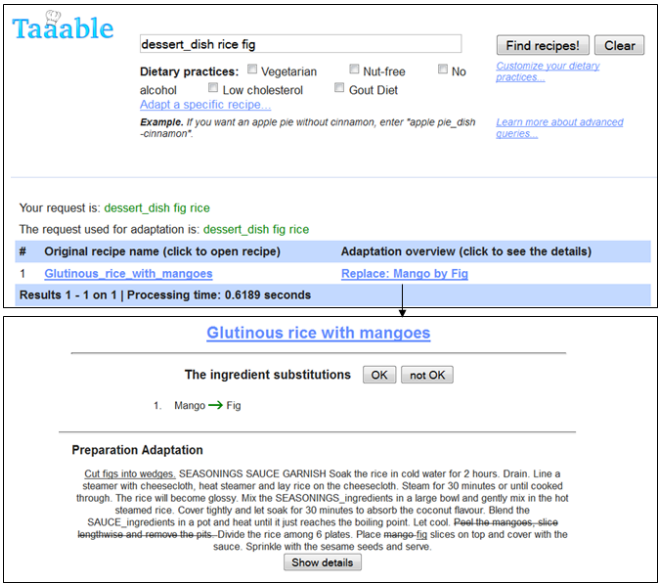

Figure 1. The TAAABLE interface. Queried for a dessert dish, with rice and fig, TAAABLE proposes to replace mango by fig in the "Glutinous rice with mangoes" recipe. After viewing the adapted recipe, the user can give feedback about the substitution ("OK" or "not OK").

2.1 TAAABLE principles

Like many CBR systems [16], TAAABLE uses an ontology to retrieve source cases that are the most similar to a target case (i.e. the query). TAAABLE retrieves and creates cooking recipes by adaptation. According to the user constraints, the system looks up, in the recipe base (which is a case base), whether some recipes satisfy these constraints. Recipes, if they exist, are returned to the user; otherwise the system is able to retrieve similar recipes (i.e. recipes that match the target query partially) 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, etc.), in order to relax constraints by generalising the user query. The goal is to find the most specific generalisation (with the minimal cost) for which recipes exist in the case base. Adaptation consists of substituting some ingredients of the source cases by the ones required by the user.

[5] `http://computercookingcontest.net/`

2.2 WikiTAAABLE

WIKITAAABLE is a semantic wiki that uses Semantic MediaWiki [11] as support for encoding knowledge associated to wiki pages. WIKITAAABLE contains the set of resources required by the TAAABLE reasoning system, in particular: an ontology of the domain of cooking, and recipes.

Figure 2. The Berry concept in WIKITAAABLE.

The cooking ontology is composed of 6 hierarchies: a *food* hierarchy (ingredients used in recipes, e.g. `Berry`, `Meat`, etc.), a *dish type* hierarchy (types of recipes, e.g. `PieDish`, `Salad`, etc.), a *dish moment* hierarchy (when to eat a dish, e.g. `Snack`, `Starter`, `Dessert`, etc.), a *location* hierarchy (origins of recipes, e.g. `France`, `Asia`, etc.), a *diet* hierarchy (food allowed or not for a specific diet, e.g `Vegetarian`, `NutFree`, etc.), an *action* hierarchy (cooking actions used for preparing ingredients, `toCut`, `toPeel`, etc.).

The set of recipes contained in WikiTaaable are those provided by the contest, that have been semantically annotated according to the domain ontology. Each recipe is encoded as a wiki page, composed of several sections: a title, which is the name of the recipe, an "*Ingredients*" section containing the list of ingredients used in the recipe, each ingredient being linked to its corresponding *Category* page in the food hierarchy, a "*Textual Preparation*" section describing the preparation process, some possible "*substitutions*" which are adaptation knowledge, and "*other information*" like the dish type, for example.

2.3 Adaptation knowledge

Improving the current results of Taaable could be done by acquiring *adaptation knowledge* (AK). In CBR systems, using AK is a classical approach for producing more fine-grained adaptations [15]. In the Taaable/WikiTaaable context, an AK is a substitution of some ingredients with other ones (e.g. in *"My Strawberry Pie"* recipe, `Strawberry` could be replaced with `Raspberry`). Formally, an adaptation knowledge is a 4-tuple $(context, replace, by, provenance)$, where:

- *context* represents the recipe or the class of recipes on which the substitution can be applied. An AK is specific if its *context* is a single recipe and generic if its *context* is a class of recipes (a specific type of dish, for example).
- *replace* and *with* are respectively the set of ingredients that must be replaced and the set of replacing ingredients.
- *provenance* is the source the AK comes from. Currently, four sources have been identified:

  1. Taaable, when AK results from a proposition of adaptation given by the reasoning process of Taaable.
  2. AK *extractor*, when AK results from a specific knowledge discovery system called "AK Extractor", which implements a KDD process also based on closed itemsets for discovering specific AK [9].
  3. *user*, when AK is given by a user by editing the wiki, as it is usually done in cooking web site, when users add comments about ingredient substitution on a recipe. See, for example, `http://en.wikibooks.org/wiki/Cookbook:substitutions`.
  4. *recipe*, when the AK is directly given by the original recipe when a choice between ingredients is mentioned (e.g. "100g butter or margarine"). This particular substitutions are taken into account by a wiki bot which runs through the wiki for automatically extracting them.

According to this definition, (*"My Strawberry Pie"*, *Strawberry*, *Raspberry*, Taaable), is an AK obtained from Taaable, meaning that strawberries can be replaced by raspberries in the *"My Strawberry Pie"* recipe. In WikiTaaable, each substitution is encoded as a wiki page like the one given in Figure 3.

In order to increase the acquisition of specific AK, a collaborative environment between users and automatic processes supported by machines has been implemented [6].

Figure 3. Example of a substitution page.

2.4 Objectives

The main objective of this work is to go beyond the discovery and the acquisition of specific AK, by discovering more generic AK. Generic AK are rules that could be applied in a larger number of situations, because the context of the AK will be a set of recipes (e.g. for cakes), instead being only one recipe (e.g. for the *"My Strawberry Pie"* recipe).

For that, an automatic KDD process can be used on cooking data and especially recipes. A crucial step of the KDD process [8] is that the results produced by the KDD process must be interpreted and validated by a (human) expert, in order to be considered as knowledge. The expert can also be involved for repairing some results that are not completely relevant. The next section details our KDD approach. In section 5, we show how the expert is involved in our approach.

Increasing the quantity and quality of AK in the wiki will improve the results of the reasoning system. However, this requires a smooth modification of the reasoning process for taking into account the AK for computing adaptation (this is a mid-term work). Besides, to ensure the non-regression of the system, the impact of the continuously new AK produced on the system results must be evaluated by test sets. In this paper, we focus on generic AK acquisition. A way for evaluating how knowledge evolution can be managed for improving the results of a reasoning process is proposed in [19].

3 Methodology for adaptation Knowledge discovery

AK discovery is based on the scheme of KDD [8]. The main steps of the KDD process are data preparation, datamining, and interpretation of the extracted units of information. Data preparation relies on formatting data for being used by datamining tools and on filtering operations for focusing on special subsets of objects and/or properties, according to the

objectives of KDD. Datamining 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 datamining results or of the noise included in these results. All the steps are managed by a computer science analyst guided by an expert of the domain.

The KDD process used in this paper is based on itemset extraction, which is introduced in 3.1. The preparation of data that will be used as entry of the itemset extraction is presented in 3.2 and the way for obtaining generic AK from itemset is detailed in 3.3. Experimentations and results are presented and discussed in 4.

3.1 Itemset extraction

Itemset extraction is a set of datamining methods for extracting regularities into data, by aggregating object items appearing together. Like FCA [10], itemset extraction algorithms start from a *formal context* K, defined by $K = (O, A, R)$, where O is a set of objects, A is a set of attributes, and R is the relation on $O \times A$ stating that an object is described by an attribute [10]. Table 1 shows an example of context, in which recipes are described by the ingredients they require: O is a set of 5 objects (r_1, r_2, r_3, r_4, and r_5) which are recipes, A is a set of 12 attributes (`Sugar`, `Water`, `Strawberry`, etc.) which are ingredients. A *cross* at the intersection of a recipe r (in line) and an ingredient i (in column) means that r requires i.

	Sugar	Water	Strawberry	PieCrust	Cornstarch	CoolWhip	Raspberry	Gelatin	Apple	AppleJuice	Cinnamon	PieShell
r_1	×	×	×	×	×	×						
r_2	×			×	×		×	×				
r_3	×	×		×			×					
r_4	×			×	×				×	×		
r_5	×	×							×		×	×

Table 1. Example of formal context representing ingredients used in recipes.

An *itemset* I is a set of attributes, and the *support* of I, $supp(I)$, is the number of objects of the formal context having every item of I. I is *frequent*, with respect to a threshold σ, whenever $supp(I) \geq \sigma$. I is *closed* if it has no proper superset J ($I \subsetneq J$) with the same support. For example, {`Sugar`, `Raspberry`} is an itemset and $supp(\{\texttt{Sugar}, \texttt{Raspberry}\}) = 2$ because 2 recipes require both `Sugar` and `Raspberry`. However, {`Sugar`, `Raspberry`} is not a closed itemset, because {`Sugar`, `PieCrust`, `Raspberry`} has the same support. In the following, "CIs" stands for closed itemsets.For $\sigma = 3$, the frequent CIs of this context are {`Sugar`, `PieCrust`, `Cornstarch`}, {`Sugar`, `PieCrust`}, {`Sugar`, `Water`}, {`Sugar`} {`Water`} {`PieCrust`}, and {`Cornstarch`}.

The *stability* of I, *stab(I)*, is the probability that I endures despite the absence of an object of the formal context having every item of I [12]. In other words, a stable CI does not result from the existence of some particular objects. Formally, let X be a set of objects and X' the maximal set of properties describing X, with $X' = \{a \in A | \forall o \in X, oRa\}$. Let n be the cardinal of X. Let $\langle C \rangle_j$ be the equivalence class of X corresponding to the set of objects of size j included in X and having the same maximal set of properties that X.

Kuznetsov define the stability as [12]:

$$stab(X') = \frac{|\bigcup_{j=2}^{n-1} \langle C \rangle_j|}{2^n - n - 2}$$

Example. Let $X_1 = \{r_1, r_2, r_4\}$, $X_1' = \{$`Sugar`, `PieCrust`, `Cornstarch`$\}$ and $n = 3$. $\langle X_1 \rangle_2 = (\{r_1, r_2\}, \{r_1, r_4\}, \{r_2, r_4\})$

$$stab(X_1') = \frac{|\bigcup_{j=2}^{n-1} \langle X_1' \rangle_j|}{2^n - n - 2}$$

$$stab(X_1') = \frac{3}{2^3 - 3 - 2}$$

$$stab(X_1') = 1$$

Let $X_2 = \{r_1, r_2, r_3, r_4\}$, $X_2' = \{$`Sugar`, `PieCrust`$\}$ and $n = 4$. $\langle X_2 \rangle_2 = (\{r_3, r_4\})$ $\langle X_2 \rangle_3 = (\{r_1, r_2, r_3\}, \{r_1, r_3, r_4\}, \{r_2, r_3, r_4\})$

$$stab(X_2') = \frac{|\bigcup_{j=2}^{n-1} \langle C \rangle_j|}{2^n - n - 2}$$

$$stab(X_2') = \frac{1 + 3}{2^4 - 4 - 2}$$

$$stab(X_2') = 0.4$$

As $stab(\{\texttt{Sugar}, \texttt{PieCrust}, \texttt{Cornstarch}\}) = 1$ and $stab(\{\texttt{Sugar}, \texttt{PieCrust}\}) = 0.4$, {`Sugar`, `PieCrust`, `Cornstarch`} is more stable than {`Sugar`, `PieCrust`}. This means that the existence of {`Sugar`, `PieCrust`} is more dependant of its objects than {`Sugar`, `PieCrust`, `Cornstarch`}. If one of the objects r_1, r_2, r_4 is removed from the formal context, X_1 CI will always exist. This is not the case for X_2. If r_3 is removed from the formal context, X_2 will not continue to be a CI.

A CI I is said *stable*, with respect to a threshold γ, whenever $stab(I) \geq \gamma$. In the previous example, with $\gamma = 0.75$, {`Sugar`, `PieCrust`, `Cornstarch`} is stable, whereas {`Sugar`, `PieCrust`} is not stable.

For the following experiments, the CHARM algorithm [21] that efficiently computes the CIs is used. CHARM is implemented in CORON, a software platform implementing a rich set of algorithmic methods for symbolic data mining [20].

3.2 Data preparation for generic AK discovering

The first step when using a closed itemsets mining algorithm is building the formal context. In order to extract generic AK, specific AK, such as the ones stored in WIKITAAABLE, will be used.

Building the formal context is guided by the set of recipes for which generic AK want to be extracted (e.g. for cakes). The characterisation of the set of recipes is the context parameter of a substitution, representing in which case the substitution can be applied. In the following and for simplification, we choose to characterise a set of recipes only by a dish type T belonging to the hierarchy of dish types stored in WIKITAAABLE. But the KDD process described in this section

could also be run on a set of recipes characterised more precisely (e.g. for cakes containing apples).

Each specific AK, as the one presented in Figure 3 will be used for producing an object in the formal context. The set of properties will be composed of the ingredients that are replaced and the ingredients they are replaced with, extended by their most generic concepts in the food hierarchy of WikiTaaable. In the formal context, replaced ingredient(s) and their generics are prefixed by `R` and replacing ingredient(s) and their generics are prefixed by `W`. Moreover, the replaced (respectively replacing) ingredient(s) of a substitution is/are duplicated and prefixed by `R_ING` (respectively `W_ING`) for distinguish it/them from their generic concepts. The idea, when prefixing the properties of the formal context like this, is to facilitate the interpretation of the CIs (see hereafter).

For example, let S_1, S_2 and S_3 be 3 specific AK substitutions for cakes, with S_1 and S_2 consisting in replacing `Strawberry` with `Apple`, and S_3 consisting in replacing `Raspberry` with `Pear`. According to the food hierarchy which states that the generic concepts of `Strawberry` are `Fruit` and `Berry`, that the generic concepts of `Apple` are `Fruit` and `PomeFruit`, that the generic concepts of `Raspberry` are `Fruit` and `Berry`, and that the generic concepts of `Pear` are `Fruit` and `PomeFruit`, the formal context presented in Table 2 is built.

	R_ING_Strawberry	W_ING_Apple	R_Strawberry	W_Apple	R_Berry	R_Fruit	W_PomeFruit	W_Fruit	R_ING_Raspberry	W_ING_Pear	R_Raspberry	W_Pear
S_1	×	×	×	×	×	×	×	×				
S_2	×	×	×	×	×	×	×	×				
S_3					×	×	×	×	×	×	×	×

Table 2. Formal context for S_1, S_2 and S_3 substitutions.

For extracting knowledge for a given dish type T, the formal context $K = (O, A, R)$ will be composed as follow:

- O is the set of substitutions in recipes producing a dish of type T;
- A is the set of replaced ingredient(s), the replacing ingredient(s) and their generic concepts, prefixed with the role (`R`, `W`, `R_ING`, `W_ING`) they play in the substitution;
- R is the relation stating that the ingredient is involved in the substitution.

For obtaining concrete results, a large number of specific AK is required for the datamining process in order to extract generic AK. So, specific AK have been taken from the *Recipe Source* database (`http://www.recipesource.com`) from ingredient lines containing a choice between ingredients. For example, a recipe containing an ingredient line such as *"500g of strawberry or apple"* is equivalent to S_1, meaning that strawberry can be replaced by apple. However, a choice $\texttt{Ing}_1$ or $\texttt{Ing}_2$ represents in fact two substitutions: $\texttt{Ing}_1$ can be replaced by $\texttt{Ing}_2$ and conversely. That is why, a choice of ingredients will produce two lines in the formal context, one for each possible substitutions.

3.3 From closed itemsets to generic AK

The formal context allows to build CIs which have to be transformed and interpreted in order to acquire generic AK. A formal context as the one given in example in Table 2 will generate CIs with prefixed ingredients, grouping together ingredients or category of ingredients depending on their regularities of co-occurrence. Figure 4 shows the first lines of the raw results resulting from the datamining process applied on specific AK that can be applied in cake dishes, ranked by their support, and organised hierarchically according to their itemset inclusion (as it is done in concept lattices).

Two kinds of CIs can be distinguished depending if the CI contains ingredient prefixed by `R_ING`, `W_ING` or not:

- a CI composed only from ingredients which are not prefixed by `R_ING` or `W_ING` is the expression of a generalised substitution rules. The ingredients of the specific AK used for building the formal context have been generalised to some more generic class of ingredient. For example, the CI {`R_Berry`, `R_Fruit`, `W_PomeFruit`, `W_Fruit`} will be produced from Table 2. It can be interpreted as follow: berry can be replaced by pome fruit in cake dishes (starting from strawberry and raspberry replaced respectively by apple and pear). In the following, this kind of substitution will be referred as a *generalised* AK.
- a CI composed with ingredients prefixed by `R_ING` or `W_ING` is the expression of a part of substitution that really exists in specific AK. For example, the CI {`R_ING_Strawberry`, `W_ING_Apple`, `R_Strawberry`, `W_Apple`, `R_Berry`, `R_Fruit`, `W_PomeFruit`, `W_Fruit`} will be produced from Table 2. When keeping only the most specific `R_ING` and `W_ING`, it can be interpreted as follows: strawberry can be replaced by apple in cake dishes. In the following this kind of substitution will be referred as *instantiated* AK.

3.3.1 CI *interpretation*

For extracting substitution AK, a CI must contain a replaced part (prefixed by `R_`) and a replacing part (prefixed by `W_`). So CIs with only ingredients prefixed by `R_` or only ingredients prefixed by `W_` will not be interpreted.

Some other simplifications are required to transform a CI into an AK. First, only the most specific ingredients prefixed by `R_` and the most specific ingredients prefixed by `W_` are kept. As illustrated previously, {`R_Berry`, `R_Fruit`, `W_PomeFruit`, `W_Fruit`} will be simplified in {`R_Berry`, `W_PomeFruit`}, from which a substitution AK can be generated. Second, if a CI contains only two items related to the same ingredient, one prefixed by `R_` and one prefixed by `W_`, then the CI will not produce a useful AK. For example, {`R_Fruit`, `W_Fruit`} will generate an AK stating that fruit can be replaced by fruit, which is not really interesting.

Depending from how the ingredient are prefixed in a CI, different kinds of AK will be generated:

- `R_ING_x`, `W_ING_y` has to be interpreted like *"Replace the ingredient x by the ingredient y"*;
- `R_ING_x`, `W_y` has to be interpreted like *"Replace the ingredient x by ingredients of the class y"*;
- `R_x`, `W_ING_y` has to be interpreted like *"Replace ingredients of the class x by the ingredient y"*;

```
(1) R_Fat, W_Fat ; Stab :1; Supp : 1510
(2)    R_Fat, R_Butter, W_Fat ; Stab :0.99; Supp : 757
(3)    R_Fat, R_Butter, R_ING_butter, W_Fat ; Stab :0.99; Supp : 740
(4)     R_fat, R_Butter, R_ING_Butter, W_Fat, W_Margarine, W_ING_Margarine ; Stab :1; Supp : 699
(5)        R_fat, R_Butter, R_ING_Butter, W_Fat, W_Shortening ; Stab :0.49; Supp : 28
(6)            R_fat, R_Butter, R_ING_Butter, W_Fat, W_Shortening, W_ING_Shortening ; Stab :0.99; Supp : 27
(7)            R_fat, R_Butter, R_ING_Butter, W_Fat, W_Shortening, W_Crisco, W_ING_Crisco ; Stab :0.5; Supp : 1
...
(17)          R_Fat, W_Fat, W_Butter ; Stab :0.99; Supp : 757
(18)              R_Fat, W_Fat, W_Butter, W_ING_Butter ; Stab :0.99; Supp : 740
(19)                  R_Fat, W_Fat, W_Butter, W_ING_Butter, R_Margarine, R_ING_Margarine ; Stab :1; Supp : 699
...
```

Figure 4. First lines of the raw output resulting from the datamining process.

- R_x, W_y has to be interpreted like *"Replace ingredients of the class x by ingredients of the class y"*.

Each CI will be presented in the interface with replaced ingredients first (ingredients prefixed by R_) followed by replacing ingredients (ingredients prefixed by W_). Moreover, if two CIs I_1 and I_2 exist, such as $I_1 = \{\texttt{R\_x}, \texttt{W\_y}\}$ and $I_2 = \{\texttt{R\_y}, \texttt{W\_x}\}$, only the first one will be kept for proposing an AK involving x and y in the interface. A symmetric arrow between x and y will represent that x can be replaced with y, and conversely, y can be replaced with x.

After simplification, only lines (2), (3), (4), (6), (7) of Figure 4 are kept, the other lines being removed by one of the simplification rules given before.

3.3.2 CI *filtering and ranking*

As the number of CIs generated by the datamining process is (often) huge, some rules are required in order to limit the number of CIs that will be presented to the expert for evaluation. Usually, CIs are filtered thanks to their support and stability: CIs with support lower than a threshold σ or with a stability lower than a threshold γ could be removed. Support and stability are also generally used for ranking the CIs. In our approach, the expert sets support and stability values through the interface (which is presented in section 5).

4 Experimentations and results

In this work, some experimentations have been realised for showing first results and for discussing about the best way of validating AK coming from CIs. In all the tables of results presented in this section, the AK propositions are symmetric.

A first experiment shows, on an example, how many AK are generated starting from a set of specific AK. For *cake* dishes, the formal context contains 2556 objects described by 978 properties, and produces 1599 CIs. Among them, 321 CIs do not contain at the same time ingredients prefixed by R_ and ingredients prefixed by W_, and can be removed. After merging two symmetric AK into one, 571 CIs still remain. 52 CIs composed from the same ingredient prefixed once by R_ and then by W_ are then removed. With a minimal support $\sigma = 3$ and a minimal stability $\gamma = 0.5$, only 131 CIs are kept, for $\gamma = 0.6$, only 113, for $\gamma = 0.7$, only 97, for $\gamma = 0.8$, only 81, for $\gamma = 0.9$, only 51, for $\gamma = 0.95$, only 37. So, increasing the stability (as well as the minimal support) is a way to limit the number of CIs that will be proposed to the expert. Unfortunately, there is no good heuristic to define automatically these thresholds. That is why these parameters may be modified in the interface. Moreover, ranking these AK by support or by stability provides results that are different, as shown in Table 3 and Table 4. Deciding between the support and the stability, which one could facilitate the choice of *good* CIs, is not easy. However, using the stability for ranking produces more instantiated AK (i.e. AK. involving ingredients prefixed by R_ING and W_ING).

Replaced	With	Supp	Stab
Butter	Fat	758	1.00
ING_Butter	Fat	741	1.00
Butter	ING_Margarine	716	1.00
ING_Butter	ING_Margarine	699	1.00
Cultured milk product	Dairy	92	1.00
ING_Pecan	Nut	55	1.00
Nut	Walnut	49	0.72
ING_Butter milk	Dairy	47	0.98
Nut	ING_Walnut	47	0.87
ING_Pecan	ING_Walnut	44	1.00
ING_Butter milk	Cultured milk product	41	0.97
Fruit	Citrus fruit	40	1.00
ING_Butter milk	ING_Sour milk	36	1.00
Coffee	Liquid	36	1.00
ING_Butter	ING_Shortening	27	1.00
Fruit	Orange	23	0.98
Fat	Dairy	21	0.99
Fruit	Berry	20	0.98
Dairy	Milk	20	0.98
Citrus fruit	Orange	17	1.00

Table 3. The 20 first AK propositions for cake dishes, ranked by decreasing support, with $\gamma = 0.7$.

Table 3 gives the 20 first AK propositions for cake dishes, ranked by decreasing support, with $\gamma = 0.7$. The support fosters the emergence of generalised AK propositions. Some of the generalised AK propositions are too generic, e.g. Fruit/Citrus fruit, Fat/Dairy, Coffee/Liquid; they will not produce a relevant AK. For example, Fat/Dairy will not produce a relevant AK because not all the subconcepts of Fat can be replaced with anyone of the subconcepts of Dairy (for example, Oil cannot be replaced with Mozzarella). However, some symmetric generic AK propositions could be relevant, but not in a symmetric way. For example, for Fat → Butter seems a relevant AK (Oil, Shortening can be replaced with Butter), but the opposite is not relevant (Butter cannot be replaced in general with Oil, when melted with sugar, for example). Instantiated AK seems more relevant regarding a dish type, e.g. ING_Butter/ING_Margarine, ING_Walnut/ING_Pecan. However some of instantiated AK could not be applied to the type of dish for which the

Replaced	With	Supp	Stab
Butter	Fat	758	1.00
ING_Butter	Fat	741	1.00
Butter	ING_Margarine	716	1.00
ING_Butter	ING_Margarine	699	1.00
Cultured milk product	Dairy	92	1.00
ING_Pecan	Nut	55	1.00
ING_Pecan	ING_Walnut	44	1.00
Fruit	Citrus fruit	40	1.00
ING_Butter milk	ING_Sour milk	36	1.00
Coffee	Liquid	36	1.00
ING_Butter	ING_Shortening	27	1.00
Citrus fruit	Orange	17	1.00
ING_Margarine	ING_Unsalted butter	15	1.00
ING_Milk	ING_Water	9	1.00
ING_Egg	ING_Egg substitute	8	1.00
ING_Coffee	ING_Water	8	1.00
Fat	Dairy	21	0.99
ING_Espresso	Coffee	14	0.99
Fruit	Stone fruit	13	0.99
ING_Butter milk	Dairy	47	0.98

Table 4. The 20 first AK propositions for cake dishes with best stability.

KDD process has been run. For example, `ING_Sour milk` and `ING_Butter` are exchangeable in `Cheesecake` but not in all type of `cakes`.

Table 4 gives the 20 first AK propositions for cake dishes with the best stability. The same kind of analysis can be done for Table 3. The major difference is the number of instantiated AK propositions which is the double with a stability ranking, comparing to the support ranking. As instantiated AK are, in proportion, more relevant than generalised AK, a way to obtain a good number of generic AK is to filter instantiated AK, as presented in Table 5. In this table, CIs are filtered for only producing instantiated AK.

Replaced	With	Supp	Stab
ING_Butter	ING_Margarine	699	1.00
ING_Pecan	ING_Walnut	44	1.00
ING_Butter milk	ING_Sour milk	36	1.00
ING_Butter	ING_Shortening	27	1.00
ING_Margarine	ING_Unsalted butter	15	1.00
ING_Milk	ING_Water	9	1.00
ING_Egg	ING_Egg substitute	8	1.00
ING_Coffee	ING_Water	8	1.00
ING_Brandy	ING_Cognac	6	0.98
ING_Espresso	ING_Strong coffee	6	0.98
ING_Butter milk	ING_Milk	5	0.97
ING_Armagnac	ING_Cognac	5	0.97
ING_Coffee	ING_Espresso	5	0.97
ING_Butter	ING_Flour	5	0.97
ING_Cake mix	ING_Chocolate	5	0.97
ING_Brandy	ING_Rum	5	0.97
ING_Butter	ING_Sour milk	5	0.97
ING_Cream	ING_Evaporated milk	4	0.94
ING_Honey	ING_Maple syrup	4	0.94
ING_Apple	ING_Orange juice	4	0.94
ING_Cream	ING_Milk	4	0.94
ING_Almond	ING_Pecan	4	0.94
ING_Lemon rind	ING_Orange	4	0.94
ING_Cream cheese	ING_Mascarpone	4	0.94

Table 5. All instantiated AK propositions ranked by decreasing support with $sigma = 4$ and $\gamma = 0.7$.

Many experiments, on various dish types, produce interesting results. In many dish types, butter can be replaced with margarine (and vice versa), which is the most frequent substitution in cooking. Some other interesting AK can be mentioned for illustration, like, for example:

- in `Beverage`: `ING_Brandy` → `ING_Liquor`, `ING_Honey` → `ING_Sugar`, `ING_Lemon juice` → `ING_Lime juice`;
- in `Dinner pie`: `ING_Cream` → `ING_Milk`, `ING_Plain yogurt` → `ING_Sour cream`, `ING_Basil` → `ING_Thyme`;
- in `Salad`: `ING_Mayonnaise` → `ING_Salad dressing`, `ING_Lemon juice` → `ING_Vinegar`, `ING_Soy sauce` → `ING_Tamari`
- in `Rice dish`: `ING_Olive oil` → `ING_Vegetable oil`, `ING_Chicken` → `ING_Ham`,`ING_Cheddar` → `ING_Monterey jack`
- in `Soup`: `ING_Leak` → `ING_Onion`, `ING_Chicken stock` → `ING_Vegetable stock`

5 Dialogue with the expert to transform CI to generic AK

The process of CI extraction, filtering and ranking allows generating AK propositions. The number of AK propositions is huge. Besides, AK have different forms (general or specific AK).

To be exploited in the recipe adaptation process, AK propositions must be validated by an expert. However, the AK result from the extraction process can not be proposed to the expert as obtained at the output of the process (see Figure 4). To facilitate the work of the expert, we will offer an interface. This interface, presented in Figure 5, allows experts to validate or to correct AK propositions. It is composed of three parts.

The first part, on the top left hand side, allows choosing parameters for filtering AK. Parameters are:

- *Type* which corresponds to the type of dish for which AK propositions are extracted. This parameter also determines the context in which the AK could be applied;
- *Min supp* which is minimal support required for a CI;
- *Min stab* which is the minimal stability required for a CI;
- Maximal *Number* of AK propositions displayed;
- Ranking method: by decreasing stability or decreasing support;
- Type of AK: generalised and/or instantiated AK.

These parameters can be chosen by experts to define the scope of the rules they want to work on.

The second part, on the bottom left hand side, displays AK propositions satisfying the filtering parameters. Each line is of the form ‘‘`Food-A Symbol Food-B`’’, where `Food-A` and `Food-B` are lists of ingredients, and `Symbol` can be a right-facing arrow (→) indicating that `Food-A` can be replaced by `Food-B`, a left-facing arrow (←) to reverse the proposed AK and thus replace `Food-B` by `Food-A`, and finally, a symmetric arrow (↔) indicating that the rule is symmetric (`Food-A` can be replaced by `Food-B` and `Food-B` can be replaced by `Food-A`). In the illustration given in Figure 5, all AK are symmetric.

On the right of each AK, three actions are possible:

- validate the AK which will be stored in WikiTaaable;
- reject the proposed AK;
- display more details about the AK.

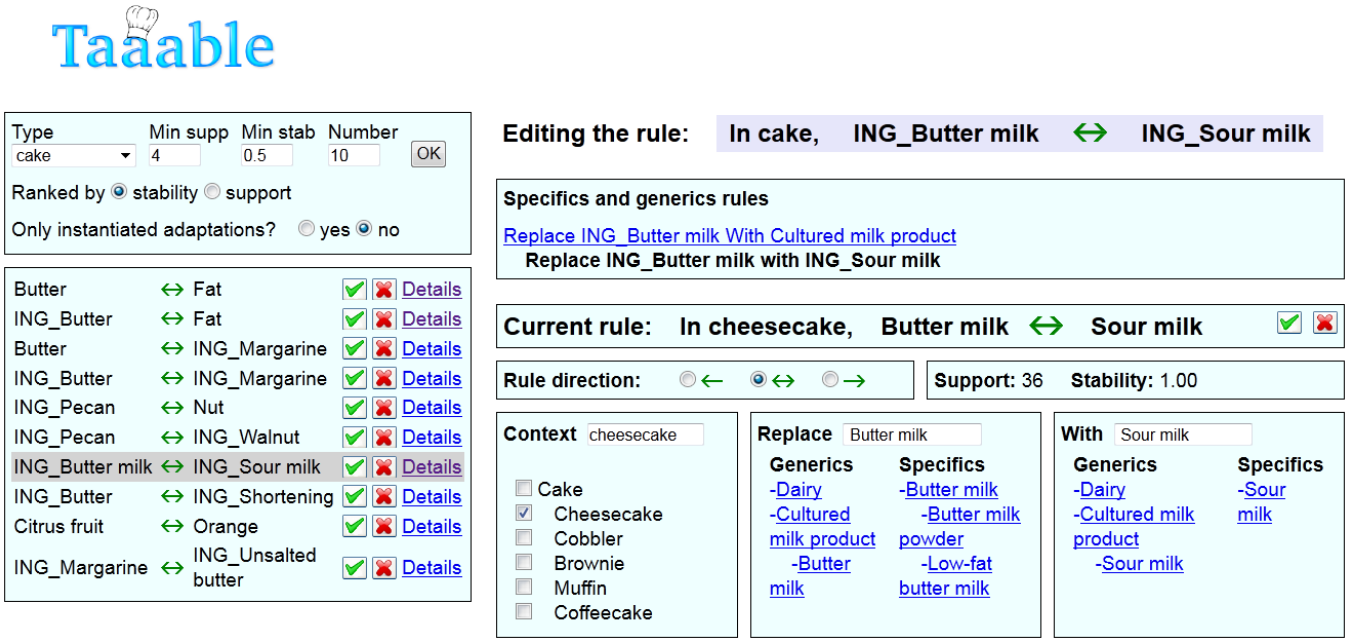

Figure 5. Expert validation interface.

The details of a AK are displayed on the third part of the interface, on the right-hand side. On this frame, the expert can make modifications in order to adapt the proposed AK.

The first line shows the proposed AK. The first frame allows the expert seeing the *"closed rules"* of the proposed AK and may navigate in the direct generics and direct specifics AK. If the expert select another AK in this list, the selected AK becomes the AK to validate and the interface is refreshed.

The second frame shows the AK which evolves according to the actions of the expert on the following frames. The *"Rule direction"* frame allows expert to modify the direction of the AK. The frame at its right indicates the support and the stability of the AK.

The *"Context"* frame allows changing the selected dish type by one or several more specific dish types in the ontology. This modification is necessary when the rule is too general and does not apply to an entire category of dish.

For example, we saw that the AK `ING_Sour milk` ↔ `ING_Butter milk` proposed for any cake cannot be validated as such because, actually, it cannot be applied to all types of cake. The expert can specify more specific categories of dishes for which the proposed AK is true, in this example, for `Cheesecake`.

The *"Replace"* and *"With"* frames allow changing the ingredients involved in the AK. This functionality is required in the case of a too general rule for involving more specific ingredients instead of too general category, or conversely. The expert can choose one of the generic or specific ingredients in the ontology.

For example, in the `Biscuit` category, the rule `ING_Margarine` → `Fat` cannot be validated because in the ingredients ontology of Taaable, `Fat` is more generic than `Bacon grease`, `Butter`, `Dripping`, `Duck fat`, `Lard`, `Margarine`, etc. Therefore, this AK indicates that margarine may be substituted by any of these ingredients or category of ingredients, which is not relevant.

After modifications, the expert can validate the adaptation rule or reject the proposed rule.

On Figure 5, the expert has selected the applicative context `cake`, *Min supp* $\sigma = 4$, *Min stab* $\gamma = 0.7$, and 10 rules to display and a ranking by stability. Then, the expert has clicked on *Details* for the rule `ING_Butter milk` ↔ `ING_Sour milk`. On the right, he has changed the applicative context `Cake` by `Cheesecake`. He can now validate or reject this rule proposition.

6 Related work

Previous researches deal with man-machine collaboration where knowledge is obtained by a knowledge extraction process which is guided or/and validated by human. In this section, we present several systems based on KDD or on CBR or even both that demonstrate this collaboration.

6.1 KDD systems

The KDD process requires to be supervised by an expert, who can interact at various levels. One of the most usual problems in a KDD is to control the over-abundance of results generated by the KDD process. The expert could, for example, interact for better selecting the data that will be mined as it is described in [3] which proposes an approach for optimising the formulation of the problem to solve. Another approach consists in filtering and ranking of numerous results obtained by the datamining algorithms. For example,

[18] proposes subjective measures of interestingness for evaluating the datamining results. These measures depend on the user profile and a result is considered as interesting for a user according to two major reasons: unexpectedness (if the user is "surprised" by the results) and actionability (if the user can exploit the results). The paper focuses on unexpected results which are results in contradiction with beliefs of the user. So, a result which may revise beliefs of a user is relevant.

Another usual problem of the KDD process is the selection of relevant information among the large set of information produced, for transforming them into knowledge. Many approaches for taking into account this step of the KDD process have been proposed. For example, [2] presents a methodology for KDD in the context of building a semi-automatic ontology from heterogeneous textual resources. [2] uses formal concept analysis (FCA) [10] for classifying objects according to their properties which are extracted from various textual resources (e.g. thesaurus, full texts, dictionaries, etc.). The results of the FCA process is translated in description logics for representing the ontology concepts. Experts are involved at each step of the KDD process. For example, when the properties describing the objects are produced by an automatic extraction process, experts have to validate and filter the most representative properties, i.e. that described the best the objects. During the last step of the process, experts have to validate the formal concepts that have been produced, by selecting those which makes sense in their domain.

The integration of the AK Extractor follows the same principle. AK Extractor implements a KDD process which produces a set of substitution propositions. These substitutions have to be validated in order to be stored as AK.

6.2 CBR systems

CBR [16] is a method for solving new problems thanks to adaptation of previously solved problems. However, the step of adaptation may fail.

In this case, an expert must take part in the repair process. In the following, we present systems in which the expert is involved in an opportunistic way to repair solutions. Most of these systems take advantage of this opportunity to acquire new adaptation knowledge.

DIAL [13] focuses on an interactive acquisition of AK in the domain of disaster response planning. When the system returns a solution that is inconsistent, the response planning is returned with a description of the elements that need to be adjusted in the planning. For example, a response planning for an earthquake in Los Angeles indicates that National Guard must be called. When this plan is used for an earthquake in Indonesia, a problem arises because there is not National Guard in Indonesia. So, the response plan must be adapted. DIAL is composed of three kinds of adaptation process. Case-based adaptation and rule-based adaptation reuse knowledge already available in the system. However, manual adaptation involves the expert. In this third type of adaptation, the expert selects a generic transformation to apply and navigate in the knowledge base to search relevant knowledge for instantiating the generic transformation. It is an opportunistically triggered man-machine collaboration for AK acquisition.

In DIAL, adaptation is performed manually transformed if automatic adaptations fail. Conversely, WebAdapt [14] proposes two independent adaptations modes: one automatic and one manual. Thus, WebAdapt allows users to choose the way they want their solutions to be adapted depending on their own needs. This is useful when users know the results they want to achieve. WebAdapt is a system for enhancing user experience when navigating on websites. It is used in the domain of sightseeing and itinerary planning. Depending on user's goals and preferences, the system builds adapted itineraries. If the user wants a highly customised itinerary, he can choose the manual adaptation that allows him to interact directly with the system and thus to refine the process of adaptation.

The FRAKAS system [4] is also an opportunistic system. During the reasoning process, FRAKAS triggers interactions with an expert, on the fly, for acquiring missing domain knowledge. Knowledge is said to be missing when an inconsistency appears in the proposed solution. On a dedicated interface, the expert can highlight inconsistent knowledge. A reasoning mechanism processes this inconsistency and proposes possible ways of solving them to the expert. Depending on the expert answer, new knowledge is acquired by the system.

Like [5], [1] uses an opportunistic approach for AK acquisition. In the second version of Taaable, which implements the approach proposed in [1], users may give some feedback on substitutions for adapting recipes. If a proposition of substitution is judged irrelevant by the user, an interface allows the user to guide the system for repairing the adaptation. The user may indicate that some ingredient(s) is/are missing when adding an ingredient, or that some ingredients of the recipe are not compatible with an ingredient that must be added. In the case where is ingredient(s) missing, a system of AK acquisition, called Cabamaka [7] based on KDD, is triggered. This last step allows to repair the bad adaptation and memorise the AK. The AK is a rule composed of ingredient(s) that have to be removed and ingredient(s) that have to be added, similar to a *substitution* AK used in WikiTaaable.

In each of the previous systems, knowledge acquisition is triggered when an adaptation fails. The originality of our approach is that AK can be triggered in parallel of the adaptation process and that this knowledge can be acquired at any time in a semantic wiki collaborative space.

7 Conclusion and ongoing work

In this paper, we described an approach for interactive acquisition of AK. We implemented this approach in Taaable, a case-based reasoning tool for adapting cooking recipes. Our approach is based on the discovery of closed itemsets (CIs), a datamining technique. First, we define a formal context in which the CIs are mined. Then, we transform these CIs in substitutions. Because a huge number of CIs is produced, we use support and stability to filter and rank the CIs before presenting them to the expert.

We have conducted several experiments to measure the influence of support and stability thresholds on the nature of the AK presented to the expert. We found that instantiated AK are, in proportion, more relevant than generalised AK. A way to obtain a good number of generic AK is to filter instantiated AK, and to adjust support and stability consequently. However, depending on the expert expectations, values for support and stability may vary. This is why we let the expert set these values himself within the interface. The interface al-

lows the expert not only to define the context in which CIs will be discovered, but also to validate and/or modify AK found by the automatic process. Therefore, this interface allows the expert to refine the quality of knowledge.

A future work to improve user interaction is to further integrate our generic AK acquisition interface in TAAABLE. By doing so, we want to allow experts to use at any time (e.g. to acquire additional AK while adapting a specific recipe). We believe that a better integration in the interface, and a connection with the TAAABLE inference engine will help experts by enabling them to perform AK acquisition in "context". For example, we could show to expert the consequences of some AK on actual recipes by providing him actual examples. By asking him "Are you sure you want to replace butter with bacon grease in this sweet cake recipe?", we help him to become aware of the applicability of the acquired rule. Therefore, this will help the expert to judge more easily the relevance of a rule.

Another future work concerns the extension of the interface by some functionalities that will help the expert to better determine the context of application of a generic AK. Datamining techniques like FCA can help again for discovering regularities in recipes linked to the set of specific AK that produce a generic AK. Indeed, the properties of these recipes, e.g. the ingredients they use, the dish types they produce, their origin, can be used as entry of a new datamining process. On the example presented in Figure 5 about the substitution of `Butter milk` with `Sour milk` with a support of 36, such a process will show that for 32 of these 36 recipes are `Cheese cake` recipes, the 4 others being `Cake` recipes. Expected results are also about some more precise context of application of generic AK, according to ingredients used in recipe, e.g. in `Salad` containing `Fish`, `Vinegar` can be replaced with `Lemon juice`.

8 Acknowledgements

This work is supported by French National Agency for Research (ANR), program Contint 2011 (ANR-10-CONTINT-025). More information about Kolflow is available on the project website: `http://kolflow.univ-nantes.fr/`.

REFERENCES

[1] F. Badra, A. Cordier, and J. Lieber, 'Opportunistic Adaptation Knowledge Discovery', in *8th International Conference on Case-Based Reasoning - ICCBR 2009*, eds., Lorraine McGinty and David C. Wilson, volume 5650 of *Lecture Notes in Computer Science*, pp. 60–74, Seattle, United States, (July 2009). Springer.

[2] R. Bendaoud, A. Napoli, and Y. Toussaint, 'Formal Concept Analysis: A unified framework for building and refining ontologies', in *16th International Conference on Knowledge Engineering and Knowledge Management - EKAW 2008*, eds., A. Gangemi and J. Euzenat, volume 5268 of *Lecture Notes in Artificial Intelligence*, pp. 156–171, Acitrezza, Catania, Italie, (2008). Springer Berlin / Heidelberg.

[3] Z. Chen and Q. Zhu, 'Query construction for user-guided knowledge discovery in databases', *Inf. Sci.*, **109**, 49–64, (August 1998).

[4] A. Cordier, *Interactive and Opportunistic Knowledge Acquisition in Case-Based Reasoning*, Thèse de doctorat en informatique, Université Lyon 1, November 2008.

[5] A. Cordier, B. Fuchs, J. Lieber, and A. Mille, 'Interactive Knowledge Acquisition in Case Based Reasoning', in *Workshop on Knowledge Discovery and Similarity, a workshop of the seventh International Conference on Case-Based Reasoning (ICCBR-07)*, eds., D. Wilson and D. Khemani, Belfast, United Kingdom, (August 2007).

[6] A. Cordier, E. Gaillard, and E. Nauer, 'Man-Machine Collaboration to Acquire Cooking Adaptation Knowledge for the TAAABLE Case-Based Reasoning System', *WWW 2012 - SWCS'12 Workshop*, 1113–1120, (April 2012).

[7] M. d'Aquin, F. Badra, S. Lafrogne, J. Lieber, A. Napoli, and L. Szathmary, 'Adaptation Knowledge Discovery from a Case Base', in *17th European Conference on Artificial Intelligence ECAI06*, ed., Traverso, Trento, Italy, (August 2006). IOS Press.

[8] *Advances in Knowledge Discovery and Data Mining*, eds., U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, AAAI/MIT Press, Menlo Park, CA, USA, 1996.

[9] E. Gaillard, J. Lieber, and E. Nauer, 'Adaptation knowledge discovery for cooking using closed itemset extraction', in *The Eighth International Conference on Concept Lattices and their Applications - CLA 2011*, pp. 87–99, Nancy, France, (October 2011).

[10] B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*, Springer, 1999.

[11] M. Krötzsch, D. Vrandecic, and M. Völkel, 'Semantic mediawiki', in *International Semantic Web Conference*, eds., I. F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, volume 4273 of *Lecture Notes in Computer Science*, pp. 935–942. Springer, (2006).

[12] S. O. Kuznetsov, 'On stability of a formal concept', *Annals of Mathematics and Artificial Intelligence*, **49**(1-4), 101–115, (April 2007).

[13] D. Leake, A. Kinley, and D. C. Wilson, 'Acquiring Case Adaptation Knowledge: A Hybrid Approach', in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 684–689, Belfast,Portland, Oregon, United States, (August 1996). AAAI Press.

[14] D. Leake and J. Powell, 'Mining Large-Scale Knowledge Sources for Case Adaptation Knowledge', in *Proceedings of the Seventh International Conference on Case-Based Reasoning*, eds., R Weber and M Richter, pp. 209–223, Belfast, United-Kingdom, (August 2007). Springer Verlag.

[15] R. Lopez De Mantaras, D. McSherry, D. Bridge, D Leake, B. Smyth, S. Craw, B. Faltings, M. L. Maher, M. T. Cox, K. Forbus, M. Keane, A. Aamodt, and I. Watson, 'Retrieval, reuse, revision and retention in case-based reasoning', *Knowl. Eng. Rev.*, **20**, 215–240, (September 2005).

[16] C. K. Riesbeck and R. C. Schank, *Inside Case-Based Reasoning*, Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey, 1989.

[17] Y. Shidochi, T. Takahashi, I. Ide, and H. Murase, 'Finding replaceable materials in cooking recipe texts considering characteristic cooking actions', in *ACM multimedia 2009 workshop on Multimedia for cooking and eating activities*, pp. 9–14, (2009).

[18] A. Silberschatz and A. Tuzhilin, 'On subjective measures of interestingness in knowledge discovery.', in *In Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pp. 275–281, Montreal, Quebec, Canada, (August 1995).

[19] H. Skaf-Molli, P. Molli, E. Desmontils, E. Nauer, Y. Toussaint, G. canals, A. Cordier, and M. Lefevre, 'Knowledge Continuous Integration Process (K-CIP)', in *WWW2012 Workshop on Semantic Web Collaborative Spaces (SWCS2012)*, (2012).

[20] 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*, 110–113, (2005).

[21] M. J. Zaki and C.-J. Hsiao, 'CHARM: An efficient algorithm for closed itemset mining', in *SIAM International Conference on Data Mining SDM'02*, pp. 33–43, (2002).