How Case-Based Reasoning on e-Community Knowledge Can Be Improved Thanks to Knowledge Reliability

Emmanuelle Gaillard, Jean Lieber, Emmanuel Nauer, Amélie Cordier

How case-based reasoning on e-community knowledge can be improved thanks to knowledge reliability*

Emmanuelle Gaillard[1], Jean Lieber[1], Emmanuel Nauer[1], and Amélie Cordier[2]

[1] Université de Lorraine, LORIA — 54506 Vandœuvre-lès-Nancy, France
CNRS — 54506 Vandœuvre-lès-Nancy, France
Inria — 54602 Villers-lès-Nancy, France
firstname.lastname@loria.fr
[2] Université de Lyon, CNRS, France
Université Lyon 1, LIRIS, UMR5205, F-69622, France
firstname.lastname@liris.cnrs.fr

Abstract. This paper shows that performing case-based reasoning (CBR) on knowledge coming from an e-community is improved by taking into account knowledge reliability. MKM (meta-knowledge model) is a model for managing reliability of the knowledge units that are used in the reasoning process. For this, MKM uses meta-knowledge such as belief, trust and reputation, about knowledge units and users. MKM is used both to select relevant knowledge to conduct the reasoning process, and to rank results provided by the CBR engine according to the knowledge reliability. An experiment in which users perform a blind evaluation of results provided by two systems (with and without taking into account reliability, i.e. with and without MKM) shows that users are more satisfied with results provided by the system implementing MKM.
Keywords: case-based reasoning, meta-knowledge, evaluation, reliability, filtering, ranking, feedback.

1 Introduction

This paper shows experimentally on a use case that taking into account knowledge reliability in case-based reasoning (CBR) improves user satisfaction about the results provided by the system. For managing knowledge reliability, we use a meta-knowledge model, called MKM, that was introduced in a previous work [1].

By analogy with past experiences, or cases, CBR solves new problems [2]. The reasoning process uses knowledge among which cases, domain knowledge,

* This work is supported by French National Agency for Research (ANR), program Contint 2011 through the Kolflow project. More information about Kolflow is available on the project website: http://kolflow.univ-nantes.fr/. The authors wish also to thank the persons who have participated to the evaluation, and especially students of the DUT Informatique of Université Lyon 1 and students of the Licence Informatique of Université de Lorraine.

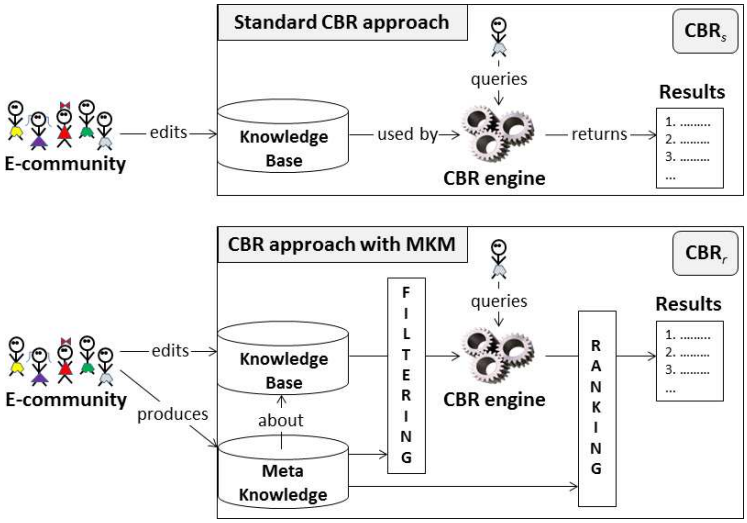

Fig. 1. Extending a standard CBR approach with MKM.

similarity knowledge and adaptation knowledge [3]. Acquiring enough knowledge to perform quality reasoning is cumbersome and tedious. This is the reason why the Web, and more specifically, e-communities, is more and more explored to build knowledge bases. An e-community is a group of people communicating over the Internet to share common ideas, goals, interests, hobbies, etc. Mining an e-community is an efficient way to acquire knowledge on a specific domain (such as video games, programming languages, or cooking). However, due to several factors (e.g. the expertise level of users, their points of view, etc.), the quality of this knowledge is questionable. For example, some people will consider that a tomato is a fruit, but this assertion is not relevant in the cooking domain because adapting a fruit salad recipe by replacing a fruit by a tomato is not really a good (a tasty) idea. Therefore, if we want this knowledge to be usable by a reasoner, while ensuring the quality of the results, we need to estimate the reliability of the knowledge originating from the e-communities. For this, we use MKM, which was designed to capture the reliability of each individual piece of knowledge, denoted KU (knowledge unit) in the following. MKM manages a new knowledge container: the meta-knowledge container, in which each KU is associated to its reliability computed from three other types of meta-knowledge: belief, trust and reputation.

To demonstrate that using MKM on a CBR system using knowledge from the Web improves user satisfaction, two CBR systems are compared. These two CBR systems use the same inference engine and the same knowledge base, built by an e-community. The first system, denoted by CBR_s (s for standard), is the reference system. It implements a standard CBR approach, in which all KUs have the same reliability. The second system extends the reference system with

MKM, where reliability of each KU depends on the community opinion. This second system is denoted by CBR_r (r for reliability).

Fig. 1 shows the common points and differences between CBR_s and CBR_r. The common points are that (1) the two systems are both triggered by queries, (2) the same CBR engine is used to perform reasoning, and (3) the reasoning is based on the same knowledge base coming from an e-community. The main difference between CBR_s and CBR_r is that CBR_r uses an additional container for meta-knowledge. This container is used to filter the KUs exploited by the CBR engine and to rank final CBR results according to the KUs involved in each result.

This paper is organized as follows. Section 2 introduces the use case. Section 3 presents MKM principles. Section 4 presents our evaluation methodology and section 5 presents and discusses the results. Section 6 concludes the paper.

2 The use-case: Taaable

Taaable is a CBR system in the cooking domain which retrieves and creates recipes by adaptation [4]. In this paper, we consider an instance of Taaable which uses KUs stored in aTaaable, a semantic wiki (in French).

The domain knowledge. The domain knowledge (DK) is an ontology composed of a set of atomic classes of several hierarchies (food, dish type, localization, ...). Classes are organized according to subsumption relations. Given two concepts A and B of this ontology, A subsumes B, denoted by $\texttt{A} \sqsupseteq \texttt{B}$, if the set of instances of B is included in the set of instances of A. For instance, $\texttt{FruitJuice} \sqsupseteq \texttt{OrangeJuice}$, means that all the instances of orange juice are instances of fruit juice.

Case base. The case base consists on a set of recipes. Each recipe R of the case base is represented by its index denoted by $idx(\texttt{R})$ which is a conjunction of classes of the domain ontology. For example, $idx(\texttt{R}) = \texttt{CocktailDish} \wedge \texttt{Tequila} \wedge \texttt{PineappleJuice} \wedge \texttt{AppleJuice} \wedge \texttt{Mint}$ is the index of a cocktail recipe which ingredients are tequila, pineapple juice, apple juice and mint.

Query. In Taaable, a query Q is also a conjunction of classes [1]. For example, $Q = \texttt{CocktailDish} \wedge \texttt{Gin} \wedge \texttt{OrangeJuice}$ means "I want a cocktail with gin and orange juice."

Case retrieval. The retrieval process consists in searching cases that best match the query. If an exact matching exists, the corresponding cases are returned. Otherwise, the query is relaxed using a generalization function Γ composed of one-step generalizations, which transforms Q (with a minimal cost) until at least one recipe of the case base matches $\Gamma(Q)$.

[1] Actually, the query language is more complex, but this has no importance in this paper.

A one step-generalization is denoted by $\gamma = \texttt{A} \rightsquigarrow \texttt{B}$, where A and B are classes and $\texttt{B} \sqsupseteq \texttt{A}$ belongs to the domain ontology. Each one-step generalization is associated to a cost denoted by $cost(A \rightsquigarrow B)$. The generalization Γ of Q is a composition of one-step generalizations $\gamma_1, \gamma_2, \ldots \gamma_n$: $\Gamma = \gamma_n \circ \ldots \circ \gamma_2 \circ \gamma_1$, with $cost(\Gamma) = \sum_{i=1}^{n} cost(\gamma_i)$. How this cost is computed is detailed in [4].

In the space of generalization functions Γ, the least costly such that at least one case matches exactly $\Gamma(\texttt{Q})$ is searched. With the example introduced above, TAAABLE produces $\Gamma = \texttt{OrangeJuice} \rightsquigarrow \texttt{FruitJuice} \circ \texttt{Gin} \rightsquigarrow \texttt{Liquor}$ and so $\Gamma(Q) = \texttt{CocktailDish} \wedge \texttt{FruitJuice} \wedge \texttt{Liquor}$, which matches $idx(\texttt{R})$.

TAAABLE returns all the adapted cases that match the generalized query. Less similar cases may be retrieved too by resuming the generalization process which will search for the *next* least costly generalization.

Adaptation. TAAABLE implements two adaptation processes. The first one consists in a specialization of the generalized query produced by the retrieval step. According to $\Gamma(\texttt{Q})$, to R, and to DK, AppleJuice is replaced with OrangeJuice in R because FruitJuice of $\Gamma(\texttt{Q})$ subsumes both AppleJuice and OrangeJuice. In the same way, Tequila is replaced with Gin in R because Liquor of $\Gamma(\texttt{Q})$ subsumes both Tequila and Gin.

The second adaptation process consists in using adaptation rules where some ingredients are replaced with others in a given context [5]. For example, in cocktail recipes, replacing Gin and AppleJuice with Tequila and OrangeJuice, is an example of adaptation rule.

Knowledge unit. Each subsumption relation, each recipe index, each adaptation rule is a KU potentially used by the ATAAABLE reasoning process.

3 Meta-knowledge model

MKM is based on [6] and was presented in a previous work [1]. The integration of MKM in a CBR system enables it to take into account reliability of each KU during the reasoning process. When a user queries the system, only the most reliable KUs are used for reasoning.

3.1 State of the art

Using reliable knowledge elements in a knowledge-based system allows to infer knowledge with an acceptable level of trustworthiness. Knowledge reliability is influenced by several factors, sometimes interrelated, as discussed in [6], where a generic model for representing knowledge generated by online communities is proposed. In an effort to provide a basis for exploiting partially reliable knowledge in a reasoning process, this work identifies a set of dimensions for knowledge reliability like origin, belief, quality, and trust. Several models have been proposed, both for conceptualizing and evaluating trust. The most common systems

exploiting trust are based on reputation (e.g. [7]) In the human computer interaction domain, Golbeck [8] asserts that "A trusts B if A commits to an action based on the belief that B's future actions will lead to a good outcome." She used this definition in her recommending system for movies, where users can rate both movies and other users. For a given user, movie recommender scores are computed by taking into account the community opinion: scores rely on movies ratings weighted by user reputation. More recently, [9] has presented a framework for the prediction of trust and distrust.

Meta-knowledge is already used in CBR, mainly for case base maintenance and recommendation. [10] presents a CBR system recommending movies to a group according to movies watched, in the past, by other groups. The similarity between two groups depends on the similarity between their members, one by one, and takes into account the degree of trust in addition to other criteria (age, gender, etc.). In [11], authors integrate trust in addition to provenance in a CBR approach to propose a model of collaborative web search. During a user search, web pages are filtered and ranked using their relevance to the query and the reputation of users having already selected the pages. In this work, a case is a pair (query/web page) and the provenance of the case is an indicator of the quality of the result.

In CBR systems, meta-knowledge is also used for case base maintenance. To maintain the CBR performance by stabilizing the case base size, [12] proposes to model case competencies according to their coverage set and their reachability set. [13] proposes to integrate the provenance of a case, to guide the case base maintenance and to increase the confidence of future results. For example, a repair is propagated through generated cases from the initial case and the quality of a case is measured by the length of the adaptation path. However, the quality of initial cases is set to a same maximum value and does not depend on external factors nor on additional meta-knowledge like, e.g., the provenance of initial cases.

More details about the state of art about meta-knowledge in literature, in particular provenance, quality and belief, and meta-knowledge used in CBR systems are presented in [1].

3.2 Meta-knowledge model principles

Fig. 2 shows the links between the different types of meta-knowledge used in MKM. Users of the e-community may evaluate KUs by giving rates, which will produce two types of meta-knowledge (Fig. 2, white background):

- The belief score, when a user u rates a KU ku and which represents the belief u has in ku.
- The *a priori* trust score, when a user u evaluates another user v and which represents the trust u has towards v, independently of any contributions inside the community.

Theses evaluations are the foundations of the system and allow to compute the intermediate meta-knowledge (Fig. 2, light grey background):

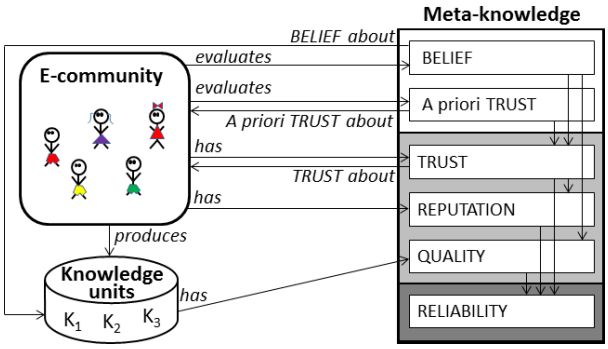

Fig. 2. Dependencies between users, knowledge units and meta-knowledge.

- A quality score of a KU ku, which represents the global quality of ku for the e-community, is inferred from all the belief scores about ku.
- A trust score from a user u towards a user v, which represents how u trusts v, is inferred from the *a priori* trust score that u has assigned to v and from the belief scores that u has assigned to the KUs produced by v.
- A reputation score of a user u, which represents the reputation of u in the e-community, is inferred from all the trust scores about u.

And finally, reliability (Fig. 2, dark gray background) is the meta-knowledge that will be used by the CBR system to filter knowledge and to rank results. Let `reliability`(ku) be the function which returns the reliability score of a KU ku for the e-community, computed from quality, trust, and reputation scores.

3.3 Plugging the meta-knowledge on a CBR system

As proposed in [1], MKM may be used to extend an existing CBR system by adding a filtering process and a ranking process.

Filtering is used to select the most *reliable* set of knowledge according to the query. All the KUs with a reliability score higher than a given threshold are selected to be used by the CBR engine (the KUs with a reliability score lower than the threshold are not used by the CBR engine as if there were removed from the knowledge base). In Taaable, a threshold of 0.3 gave good results during the experiments. Thus, the threshold is fixed to 0.3 for this work, but a precise method to fix the threshold remains to be studied.

Ranking computation is used to order the set of results according to the meta-knowledge associated to the KUs involved in the computation of the results. For each result `R`, an *inferred reliability*, denoted by `inferred_reliability`(`R`) is computed. The general function proposed in the initial model [1] to compute the inferred reliability has been instantiated by a probabilistic approach. The *inferred reliability* of a result can be seen intuitively as the probability that the result will be satisfactory (e.g., for a cooking application, the probability that

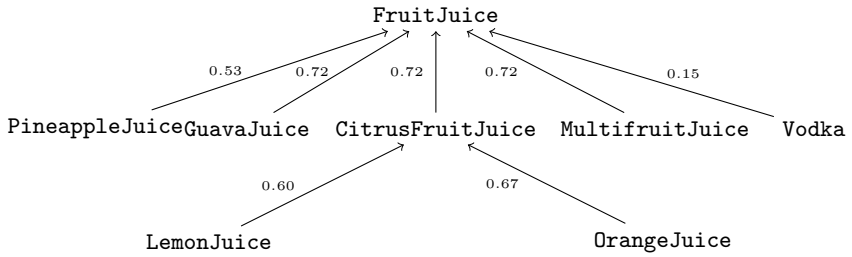

Fig. 3. Part of the food hierarchy with reliability scores of subsumption relations.

this is the recipe of a tasty dish); this probability depends on the probability that the retrieved case is satisfactory, that each KU used in the adaptation is satisfactory, and we assume that these probabilities are independent one from another. Each probability is equivalent to the reliability score of the KU.

The *inferred reliability* of a result is computed as follows:

- If the result matches exactly the query, the *inferred reliability* is the reliability score of the case.
- If the adaptation process uses an adaptation rule for producing the result `R`, the *inferred reliability* of `R` is the product of the case reliability by the adaptation rule reliability. For example, if the retrieve case is `C` (a cocktail containing apple juice), with $\texttt{reliability}(\texttt{C}) = 0.76$, and if the adaptation rule `AR`, which proposes to replace apple juice with orange juice in cocktails, has been used, with $\texttt{reliability}(\texttt{AR}) = 0.8$, then $\texttt{inferred\_reliability}(\texttt{R}) = \texttt{reliability}(\texttt{C}) \times \texttt{reliability}(\texttt{AR}) = 0.76 \times 0.8 = 0.608$.
- If the adaptation process uses a generalization-specialization path to compute the adaptation, the *inferred reliability* is the product of the case reliability by the product of the reliability of all the KUs involved during the generalization-specialization, that is to say, the product of the reliability of each subsumption relation involved. Fig. 3 presents a part of the food hierarchy with the reliability score of the subsumption relations. For example, according to Fig. 3, the reliability of the adaptation $A = \texttt{PineappleJuice} \rightsquigarrow \texttt{GuavaJuice}$ may be computed by taking into account that `PineappleJuice` has been generalized in `FruitJuice` and that `FruitJuice` has been specialized in `GuavaJuice`: $\texttt{reliability}(\texttt{PineappleJuice} \rightsquigarrow \texttt{GuavaJuice}) = 0.53 \times 0.72 \simeq 0.38$. Let `R` be the result of case `C` adapted using the substitution `A`, the inferred reliability of `R` is computed as the product of the reliabilities of `C` and `A`: $\texttt{inferred\_reliability}(\texttt{R}) = \texttt{reliability}(\texttt{C}) \times \texttt{reliability}(\texttt{A}) \simeq 0.76 \times 0.38 \simeq 0.29$.

3.4 Example of results using TAAABLE with and without MKM

Let $\texttt{Q} = \texttt{GrenadineSyrup} \wedge \texttt{GuavaJuice}$ be the query. Table 1 presents the three first recipes returned by CBR_s and CBR_r with their original ingredients, their reliability scores and their adaptation ids which corresponds to an adaptation

id	name	$idx(R_i)$	system	reliability	adaptation
$\mathtt{R}_1$	Bora bora	`AppleJuice` $\wedge$ `PineappleJuice` $\wedge$ `LemonJuice` $\wedge$ `GrenadineSyrup`	CBR_s + CBR_r	0.76	A_2
$\mathtt{R}_2$	Tequila sunrise	`Tequila` $\wedge$ `OrangeJuice` $\wedge$ `GrenadineSyrup`	CBR_r	0.73	A_3
$\mathtt{R}_3$	Bacardi cocktail	`Rum` $\wedge$ `GrenadineSyrup` $\wedge$ `LemonJuice`	CBR_r	0.72	A_4
$\mathtt{R}_4$	Spice shoot	`GrenadineSyrup` $\wedge$ `Tabasco` $\wedge$ `Vodka`	CBR_s	0.73	A_5
$\mathtt{R}_5$	MTS cocktail	`Martini` $\wedge$ `TripleSec` $\wedge$ `CaneSugarSyrup` $\wedge$ `LemonJuice`	CBR_s	0.4	A_1

Table 1. The three first recipes returned by CBR_s and CBR_s, according to `Q`, with their indexes, their reliability scores and their adaptation ids in Table 2.

id	adaptation	reliability
$\mathtt{A}_1$	`MultifruitJuice` $\rightsquigarrow$ `GuavaJuice`	0.52
$\mathtt{A}_2$	`PineappleJuice` $\rightsquigarrow$ `GuavaJuice`	0.38
$\mathtt{A}_3$	`OrangeJuice` $\rightsquigarrow$ `GuavaJuice`	0.35
$\mathtt{A}_4$	`LemonJuice` $\rightsquigarrow$ `GuavaJuice`	0.33
$\mathtt{A}_5$	`Vodka` $\rightsquigarrow$ `GuavaJuice`	0.11 (filtered)

Table 2. Adaptations of recipes returned by CBR_s.

id of Table 2, that presents adaptations involved in the retrieved recipes. In this example, adaptations have been computed by generalization-specialization process of the query to the part of food hierarchy presented in Fig. 3. In this figure, `FruitJuice` $\sqsupseteq$ `Vodka` is a KU created by a user and this KU is unreliable since most users have considered that Vodka is not a fruit juice.

The first three results of CBR_s are, in this order:

1. $\text{S}_{\text{s}1}$: Spice shoot with adaptation `Vodka` $\rightsquigarrow$ `GuavaJuice`;
2. $\text{S}_{\text{s}2}$: Bora bora with adaptation `PineappleJuice` $\rightsquigarrow$ `GuavaJuice`;
3. $\text{S}_{\text{s}3}$: MTS cocktail with adaptation `MultifruitJuice` $\rightsquigarrow$ `GuavaJuice`.

The first three results of CBR_r are, in this order:

1. $\text{R}_{\text{s}1}$: Bora bora with adaptation `PineappleJuice` $\rightsquigarrow$ `GuavaJuice`;
2. $\text{R}_{\text{s}2}$: Tequila sunrise with adaptation `OrangeJuice` $\rightsquigarrow$ `GuavaJuice`;
3. $\text{R}_{\text{s}1}$: Bacardi cocktail with adaptation `LemonJuice` $\rightsquigarrow$ `GuavaJuice`.

$\text{S}_{\text{s}1}$ is not returned by CBR_r because even if the case entitled "Spice shoot" has a reliability score of 0.73, the adaptation knowledge `Vodka` $\rightsquigarrow$ `GuavaJuice` has been filtered because of its reliability score of 0.15.

The first result returned by CBR_r is $\text{R}_{\text{s}1}$ which is only the second result returned by CBR_s. The *inferred reliability* of $\text{R}_{\text{s}1}$ is 0.29 corresponding to the

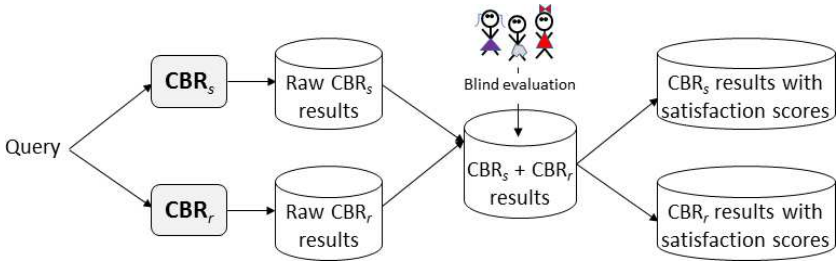

Fig. 4. Evaluation process.

products of the reliability score of its case and its adaptation. The *inferred reliability* of s_{s2} is 0.26 and the *inferred reliability* of s_{s3} is 0.24. With lower *inferred reliability* s_{s3} is not returned in the first three results of CBR_r.

This example illustrates that KUs which are not reliable (e.g., `FruitJuice` $\sqsupseteq$ `Vodka`) are not used by CBR_r and that a case with a *good* reliability adapted by a long generalization/specialization path can be better ranked than a case with a *poor* reliability adapted by a shorter generalization/specialization path (e.g., s_{s3} vs. R_{s3}).

4 Evaluation

This section proposes a methodology of evaluation and explains how to analyze the results coming from user tests in order to compare two CBR systems. Section 5 applies this methodology to the TAAABLE use case.

4.1 Evaluation methodology

As our objective is to demonstrate that managing reliability thanks to MKM in a CBR system improves results returned by the system, the two systems that will be compared are CBR_s and CBR_r and the hypothesis we have to validate is:

(H) CBR_r returns more satisfactory results than CBR_s.

In order to validate (H), a set of queries is performed on the two systems CBR_s and CBR_r. The results of both systems are presented to users who have to evaluate their relevance using an evaluation scale. Fig. 4 shows that for each query, results of the two systems are computed. The results of these two systems are gathered and displayed to a user in a random order so that the user does not know from which system a result has been computed nor the ranking of this result in the result list.

For each evaluation, the user has to evaluate her satisfaction for each proposed results inferred by the two systems according to the target problem. Many

possibilities exist to collect the satisfaction feedback of the user. One of them is the Likert scale [14], which is based on a set of grades, allowing the users to qualify their satisfaction degree. The most frequent scale uses 5 grades, so that users have not too many nor too few options to express their opinions. Among these 5 grades, 2 are positive (very satisfied, satisfied), 1 is neutral (neither satisfied nor unsatisfied) and 2 are negatives (unsatisfied and very unsatisfied). An advantage of this scale is also that it can be easily be turned into a numerical score.

4.2 Result analysis

In order to verify (H), we have to analyze data from the user evaluation. We draw inspiration from results analysis in recommender systems evaluation. Recommender systems are information filtering systems which predict ratings of users about items (e.g., movies) and propose the best rated item to a user. Recommender systems and CBR systems may be considered as similar both because they match a user query and they return a set of results which may satisfy the user. For this reason, and like it was done in [15], it is possible to analyze the results provided by two CBR systems in the same way that results are analyzed in recommender systems evaluations. In recommender systems, evaluation usually consists in comparing at least two systems that return a set of recommended items in order to measure the performances of the systems. As we are in a similar case, i.e. evaluating the performance of two different systems, the way to compare their performance must be discussed.

Comparing recommender systems. Let A and B be two recommender systems. Testing whether B is better than A consists in testing that the results provided by B are better that those provided by A. For that, if a positive difference between the two systems is observed, this difference has to be significant. A standard measure for that is the p-value which goal is to evaluate the null hypothesis. The null hypothesis assumes that positive results of an experimentation are obtained due to chance. In order to obtain significant results, the null hypothesis must be rejected. If p-value ≤ 0.05 the null hypothesis is rejected and results are significant. To test (H), the results of $\mathrm{CBR_s}$ and $\mathrm{CBR_r}$ will be compared. Data resulting from the user evaluations must be prepared and cleaned before analysis. Moreover, variables of the analysis must be determined in order to establish the appropriate statistical test.

Data preparation and cleaning. Degrees of the Likert scale must be transformed into numerical scores to allow quantitative analysis. We associate a score to each term of the evaluation scale. Very satisfied: 2, satisfied: 1, neither satisfied nor dissatisfied: 0, unsatisfied: -1 and very unsatisfied -2.

Each query is rated by several (at least 4) users, so that the lowest and the highest ratings are excluded in order to limit the impact of *random* ratings.

Variables and tests. The choice of the system is the independent variable (or controlled variable), that is to say, it is the variable that impacts results. The user satisfaction score is the dependent variable (or measured variable) which depends on the independent variable. The measured variable is a non parametric (ordinal) variable which does not have a Gaussian distribution and where possible values are -2, -1, 0, 1 and 2.

The users which evaluates the results of $\mathrm{CBR_s}$ and of $\mathrm{CBR_r}$ are the same. More precisely, a same user evaluates the set of results returned by both systems on a same query. These two sets are paired samples.

The test requires to evaluate that a significant result must take into account no parametric variables and paired sample. The *Wilcoxon signed-rank test* [16] allows to test that the median of the aggregated satisfaction scores of $\mathrm{CBR_r}$ is significantly higher than the median of the added satisfaction scores of $\mathrm{CBR_s}$.

Performance measurement. In recommender systems, precision and recall are performance metrics frequently used [16]. Precision is the proportion of retrieved items which are relevant among the set of items returned by the system while recall is the proportion of relevant items retrieved on the total set of relevant items. For a given query, a retrieved item is relevant for a user if the preference score given by the user is higher or equal than a given threshold. In our evaluation, the threshold is set to 1. Because results to a query in CBR systems is not a finite set, we cannot apply any recall measure.

5 Evaluation of $\mathrm{CBR_s}$ and $\mathrm{CBR_r}$ in the Taaable use case

The evaluation aims at validating (H) where the two compared systems have to reason on knowledge coming from the Web. The methodology presented in the previous section has been used to perform this evaluation in the context of the Taaable system, using knowledge of aTaaable[2], a collaborative web site in which users may interact with KUs used by Taaable. aTaaable was initially built by translating in French the domain knowledge of WikiTaaable[3], the recipes which form the case base (a case is a recipe described by ingredients, dish types, etc.), as well as the adaptation rules have been entered manually by the users of the community. At the time of writing this paper, aTaaable contains 2325 classes linked by 2551 hierarchical relations (coming from WikiTaaable), and 163 recipes (of which 129 cocktail recipes), 11 specific and 25 generic adaptation rules, were entered by 80 users. Entering one of these KU is facilitated by specific interfaces, so that it is very easy to add new ones. Each of the KU may also be easily rated (in one click) by the users of the community using a 5-point scale.

However, the KUs describing the domain knowledge of aTaaable are rather consensual, because they were built by a knowledge representation specialist. To simulate knowledge coming from an e-community, the domain knowledge

[2] `http://ataaable.loria.fr/`
[3] `http://wikitaaable.loria.fr/`

Fig. 5. Evaluation interface for the query `GrenadineSyrup` $\wedge$ `GuavaJuice`.

container has been a little bit damaged, by adding, for example, a few bad subsumption relations (e.g. `FruitJuice` $\sqsupseteq$ `Vodka`), in order to test if MKM is able to manage erroneous KUs using the evaluation of these KUs by the aTaaable users.

5.1 Applying the evaluation methodology on TAAABLE

In order to apply the evaluation on Taaable, some choices have been made to fix the experimental conditions. In order to limit the number of tests, we focused on a subset of the case base: the aTaaable recipes about cocktails. Each query used for the test is composed of 2 required ingredients, each of them appearing at least once in a cocktail recipe of the base. The choice of 2 ingredients is motivated by the expectation of results which will be built by adaptation instead of simply being retrieved as an exact matching recipe. The set of queries has been randomly generated and has been slightly filtered, in order to keep interesting queries, i.e. queries that might be submitted by a real user. Two criteria were taken into account for this filtering:

1. The query must only contain ingredients known from a majority of users, because the users have to evaluate the results. For example, queries containing "*Pisang Ambon*" or "*Angustura*" have been manually removed.
2. The query must look like a real user request. For example, a query like `Salt` $\wedge$ `Pepper` has been manually removed.

Subjects were bachelor students who participate to an approximative 20-30 minutes test session organized in a classroom by the authors of this paper. They received an short (5 minutes) oral explanation about the aTaaable and Taaable systems, the instructions for their participation to the evaluation test being written in the test interface. They register on aTaaable to become users and evaluate at least 10 KUs each, randomly proposed but having a link with the adaptation of cocktail (cocktail recipes and hierarchical relations about liquids). The goal was to refine the reliability about these KUs; 396 rates have been collected from 18 users in 5 minutes. Then, each user evaluates the results of about 4 queries, and each query has been evaluated by 4 different users. For the experimentation presented here, 15 queries have been evaluated (thanks to 22 users). For each query, the system results were displayed to users. A result

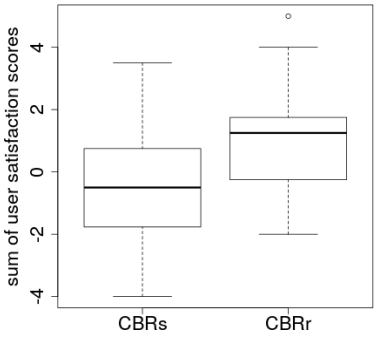

Fig. 6. Comparison of median satisfaction scores of users for each query in CBR_s and CBR_r.

is composed of the title of the original recipe (a link allows the user to access the recipe), and the ingredient list of the adapted recipe is the result of the adaptation. Fig. 5 shows a part of the evaluation interface for one result of the query `GrenadineSyrup` $\wedge$ `GuavaJuice`. A user has to enter her satisfaction only according to the composition of the cocktail (so, without testing it).

5.2 Results analysis

The results provided in this section show that using reliability on the knowledge of aTaaable using by Taaable returns more satisfactory results for users than not using it.

Wilcoxon signed-rank test. Fig. 6 compares box-plot of CBR_s and CBR_r. We observe that the median of the satisfaction scores attributed to CBR_s is -0.5 and the median of the satisfaction scores attributed to CBR_r is 1.25. The *Wilcoxon signed-rank test* allows to observe a true difference between the results of CBR_r and CBR_s. The p-value is 0.05: the null hypothesis is rejected. Fig. 7 compares CBR_s (black background rectangles) and CBR_r (white background rectangles). A rectangle corresponds to the sum of the average of the satisfaction scores attributed by users on each of the five first results of the query (as those presented in Tables 3 and 4, for the query with the id 13). We observe that the user satisfaction is higher with CBR_r than with CBR_s 11 times out of 15. The query with the id 13 in Fig. 7 is the query `GrenadineSyrup` $\wedge$ `GuavaJuice`. Tables 3 and 4 show the average of the user satisfaction scores of each of the five first results returned by CBR_s and CBR_r for this query. Except for the second result, user satisfaction scores are better with CBR_r.

Precision. The precision of the CBR_s system is 0.3 while the precision of the CBR_r system is 0.43. Thus, precision of CBR_r system is higher than the one of CBR_s.

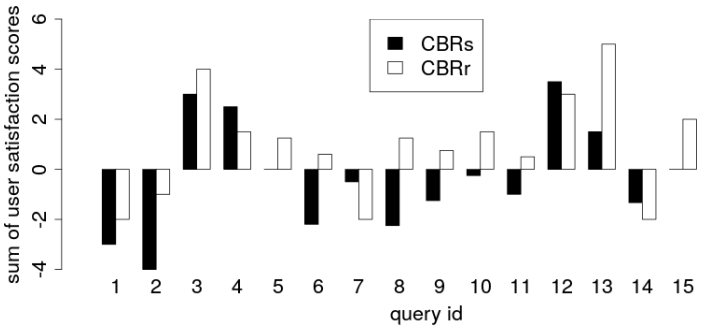

Fig. 7. Comparison of the aggregated satisfaction score averages for each query in CBR_s and CBR_r.

result id	user satisfaction score average
S_{s1}	-0.5
S_{s2}	1.5
S_{s3}	1
S_{s4}	-1
S_{s5}	0.5

Table 3. Average of user satisfaction scores of CBR_s for query #13.

result id	user satisfaction score average
R_{s1}	1
R_{s2}	1
R_{s3}	1
R_{s4}	1.5
R_{s5}	0.5

Table 4. Average of user satisfaction scores of CBR_r for query #13.

Hypothesis conclusion. Since a significant positive difference has been found between CBR_r and CBR_s and precision of CBR_r is higher than precision of CBR_s, (H) is validated: CBR_r returns better results than CBR_s. Thus, using MKM model in a CBR system improves results returned by the system.

6 Conclusion and ongoing work

This paper shows that extending a CBR system reasoning on knowledge coming from the Web, with MKM, a meta-knowledge model which manages KUs reliability, increases user satisfaction. MKM model ensures to reason on the most reliable knowledge and to rank inferred results according to a reliability point of view.

An experimentation validates the hypothesis that the results of CBR_r are more satisfying than those of CBR_s and that there is a significant difference between user satisfaction of CBR_s and CBR_r.

In a short term, we want to demonstrate that personalizing the reliability for users of the e-community, taking into account user preferences in the computation of the *reliability score* of KUs increases user satisfaction. Next, we want to implement filter and ranking functions of MKM in the engine of a CBR system, where retrieval and adaptation processes will be guided by reliability.

References

1. E. Gaillard, J. Lieber, Y. Naudet, and E. Nauer. Case-based reasoning on e-community knowledge. In *Case-Based Reasoning Research and Development*, pages 104–118. Springer Berlin Heidelberg, 2013.
2. C.K. Riesbeck and R.C. Schank. *Inside Case-Based Reasoning*. Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey, 1989.
3. M. Richter. The knowledge contained in similarity measures, 1995. Invited talk at the International Conference on Case-Based Reasoning.
4. A. Cordier, V. Dufour-Lussier, J. Lieber, E. Nauer, F. Badra, J. Cojan, E. Gaillard, L. Infante-Blanco, P. Molli, A. Napoli, and H. Skaf-Molli. Taaable: a Case-Based System for personalized Cooking. In Stefania Montani and Lakhmi C. Jain, editors, *Successful Case-based Reasoning Applications-2*, volume 494 of *Studies in Computational Intelligence*, pages 121–162. Springer, January 2014.
5. 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*, 2011.
6. Y. Naudet, T. Latour, G. Vidou, and Y. Djaghloul. Towards a novel approach for high-stake decision support system based on community contributed knowledge base. In *10th International Conference on Intelligent Systems Design and Applications (ISDA)*, pages 730 –736, 2010.
7. D. Artz and Y. Gil. A survey of trust in computer science and the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):58–71, June 2007.
8. J.A. Golbeck. *Computing and applying trust in web-based social networks*. PhD thesis, University of Maryland, 2005.
9. A. K. Young and A. A. Muhammad. Trust, distrust and lack of confidence of users in online social media-sharing communities. *Knowledge-Based Systems*, 37(0):438 – 450, 2013.
10. L. Quijano-Sánchez, D. Bridge, B. Díaz-Agudo, and J. Recio-García. Case-based aggregation of preferences for group recommenders. In *Case-Based Reasoning Research and Development*, pages 327–341. Springer Berlin Heidelberg, 2012.
11. Z. Saaya, B. Smyth, M. Coyle, and P. Briggs. Recommending case bases: Applications in social web search. In *Case-Based Reasoning Research and Development*, pages 274–288. Springer Berlin Heidelberg, 2011.
12. S. Barry and M.T. Keane. Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'95, pages 377–382, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.
13. D. Leake and M. Whitehead. Case provenance: The value of remembering case sources. In *Proceedings of the 7th international conference on Case-Based Reasoning: Case-Based Reasoning Research and Development*, pages 194–208, 2007.
14. R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 22(140):1–55, 1932.
15. L. Quijano-Sánchez, J. Recio Garcia, and B. Díaz-Agudo. Using personality to create alliances in group recommender systems. In A. Ram and N. Wiratunga, editors, *Case-Based Reasoning Research and Development*, volume 6880 of *Lecture Notes in Computer Science*, pages 226–240. Springer Berlin Heidelberg, 2011.
16. F. Ricci, L. Rokach, B. Shapira, and P.B. Kantor. *Recommender Systems Handbook*. Springer, 2010.