Evaluating a textual adaptation system

Valmi Dufour-Lussier[1,2] and Jean Lieber[2,3,4]

[1] Université de Moncton, campus de Shippagan, New-Brunswick, Canada
[2] Université de Lorraine, LORIA — 54506 Vandœuvre-lès-Nancy, France
[3] CNRS — 54506 Vandœuvre-lès-Nancy, France
[4] Inria — 54602 Villers-lès-Nancy, France
vdl@umcs.ca, lieber@loria.fr

Abstract. This paper presents a CBR method to retrieve and adapt processes represented as instruction texts, as well as the evaluation methodology that we developed to evaluate it. The evaluation process is user-based, blind and comparative. It is less labour intensive than most existing approaches and is more open to a variety of possible solutions to the same query, among other benefits. It also makes it possible to evaluate separately the textual adaptation process and the underlying formal adaptation process. Craqpot, a CBR system that adapts recipe texts, using a case-based process to extract domain knowledge on the fly, is presented and evaluated. We show that it generates recipes of good quality and texts of acceptable quality.

Keywords: adaptation, evaluation, textual case-based reasoning, process-oriented case-based reasoning

1 Introduction

Textual [26] and process-oriented [18] case-based reasoning are two fields of case-based reasoning (CBR) that tend to use unconventional case structures. They have therefore required the development of specific retrieval techniques, which are rather well established nowadays. Adaptation, on the other hand, has been more problematic. In textual CBR, it has mostly been limited to selecting and aggregating parts of textual cases. Different techniques have been proposed recently to make a deeper level of adaptation possible in process-oriented CBR.

We propose Craqpot, a CBR system that retrieves and adapts processes represented as instruction texts. In this system, cases are recipe texts associated with a formal case structure. The structure consists in a network of temporal constraints on events, represented using a qualitative algebra based on Allen's interval calculus [3]. When a user makes a query, a case is retrieved, and both the text and the formal structure are adapted. In order to evaluate the quality of those adaptations, we have made Craqpot available as a Facebook application that offers a helpful service to users while encouraging them to provide evaluations.

In sections 2 and 3, a short introduction to adaptation in textual and process-oriented case-based reasoning is given, and existing evaluation frameworks are

discussed. Section 4 presents CRAQPOT, and section 5 details how the domain knowledge that is needed for the adaptation is "simulated". The evaluation methodology is presented in section 6, and section 7 presents the results. Concluding remarks and future work are shown in section 8.

2 Adapting textual cases

The CBR community recognises that significant knowledge is available in a textual format and, consequently, that being able to exploit this text can be a great help in deploying CBR applications. There has been a significant interest since the very beginning of CBR in systems which use texts as cases. This interest has been expressed, among other things, by a series of workshops on textual CBR at the 1998 AAAI conference, as well as at International and European CBR Conference from 2005 to 2007. Most work in textual CBR has focused on retrieval, but a few have taken an interest in trying to reuse texts.

2.1 Principles

The problem of text reuse in textual case-based reasoning has been addressed in different manners. In [1], textual solutions are reused by identifying small chunks of text to be reused from different solutions and aggregating them, which can be seen as a type of compositional adaptation. An inverse approach is also possible, in which a text is reused in whole but parts that should be modified are identified [14].

Another way is to use a natural language generation system following the adaptation of the underlying formal representation of the textual solution [13]. The approach we propose is based on regeneration, that is starting from existing text or text fragments and making linguistic changes therein. We are aware of only one other system that uses a similar approach, which is COOKIIS [21], which performs string substitutions based on ingredient substitutions in recipe texts.

2.2 Evaluation

The part of a textual adaptation system responsible for selecting parts of a case text to be reused as part of a solution text can be evaluated. The typical way to evaluate such a system is to annotate manually the sentences that are expected to be reused in the answer to test queries, then compare the actual result of the system with the expected result, computing a precision and a recall score.

On the other hand, other aspects of textual adaptation have not really been evaluated before. In particular, we are aware of no prior work that aimed at evaluating actual text quality, nor at evaluating the quality of the text adaptation separately from the quality of the underlying adaptation mechanism.

3 Adapting procedural cases

More often, in CBR, the temporal aspect is taken into account by considering sequences of events, sometimes integrating relative or absolute time stamps [8, 15, 25]. The most advanced work in this respect is that of process-oriented CBR (PO-CBR), in which cases are often made of activities structured using workflows. In CBR, workflows are usually expressed in a graph-based formal language, such as the one described in [19], to make retrieval and adaptation possible. Again, retrieval has received substantial research interest, but little work has been done on the adaptation of procedures.

3.1 Principles

Arguably the first approach to workflow adaptation was case-based adaptation: adaptation cases, which are combinations of a source case, a change request and the resulting case, are used as a source of adaptation knowledge [16]. A somewhat similar approach identifies small workflow parts from the case base that attain specific goals, and uses this to make substitutions of parts of the retrieved workflow [20].

3.2 Evaluation

Most evaluation work in procedural adaptation is manual: either test queries are provided along with the expected result, or users are asked to evaluate the quality of the result. This is labour intensive and requires the intervention of both domain experts and of people familiar with the formalism used.

An alternative, automatic approach is to compare the various parts of the solution with the case base, the expectation being that if a generated workflow part describes a feasible activity, it is likely to occur naturally in a large enough case base [17].

4 Craqpot

In this section, we present our own approach to textual and procedural adaptation and its software implementation, named Craqpot—the Case-based Reasoning Adaptor of Qualitative Procedures Over Texts—which provides an interface to obtain recipes in response to any query. As its case base, Craqpot uses the recipe database published for the 2nd Computer Cooking Contest.[5] The processes for case acquisition, for retrieval, for formal case adaptation and for textual adaptation are all implemented as separate modules, and will therefore be presented individually in the following subsections.

[5] `http://www.wi2.uni-trier.de/shared/eccbr/ccc09/`

4.1 Case acquisition

The cases are provided as unannotated text, and so their formal counterpart must be extracted to make adaptation possible. This is the case acquisition step, which is detailed in [11]. Our approach is based on natural language processing, and therefore goes through much of the same main steps as any other natural language understanding system:

- Identifying word, clause and sentence boundaries. This task is performed using hand-crafted regular expressions.
- Identifying the part-of-speech of each word, i.e. finding verbs, nouns, etc. This task is performed by a Brill tagger [5], a semi-supervised machine learning tool trained on a small set of annotated recipes.
- Performing syntactic analysis. This is done using a chunker, which is a parser using a regular grammar, implemented using regular expressions, which is not able to compute a complete parse tree, but can find noun and prepositional phrases. Those are sufficient to identify verb complements, which correspond to action parameters.

This is not the only possible approach: for instance, in [23], satisfying results are obtained over the same type of texts, using information extraction. Both approaches are efficient with instruction texts but would require adjustments to give good results with different types of text. Another important and difficult step is resolving anaphoras i.e. associating words from the text with the objects they are referring to. To this end, we implemented certain ideas from dynamic semantics, wherein actions expressed in the text are considered as creating, transforming and removing objects.

Once all the relevant linguistic information has been identified in a text, annotation rules are used to translate it into workflow patterns or, more interestingly in our case, in qualitative constraints between events—cooking actions and states—expressed using the qualitative algebra $\mathcal{INDU}$ [22], an extension of Allen interval calculus [3]. The 9 annotation rules used in this implementation are detailed in [9]

As an example, the following $\mathcal{INDU}$ constraints would be part of a simplified formal case representation of the recipe shown in figure 1:

$$\begin{array}{rl} \text{cook rice } ?^{=} \text{ 18 min} & \textit{Rice cooks for 18 minutes.} \\ \text{cook mushrooms } ?^{=} \text{ 2 min} & \textit{Mushrooms cook for 2 minutes.} \\ \text{cook mushrooms } \{\mathsf{f}^{>}\} \text{ cook rice} & \textit{Mushrooms start cooking after rice and finish at the same time.} \end{array} \quad (1)$$

Because case acquisition from text is not perfect, it is essential to evaluate it separately to interpret the overall evaluation results of the system, because any error at this stage will correspond to a decrease in the solution quality further down the road.

Mushroom risotto
Heat the oil and butter. Add the onion and cook until soft, about one minute. Add the rice and cook for two minutes, then add a glass of wine. Once the wine is evaporated, start adding broth, one ladleful at a time. Meanwhile, slice the mushrooms. Add them two minutes before the end.

Fig. 1. A simple mushroom risotto recipe.

4.2 Retrieval

While it could in theory rely on an approach inspired by adaptation-guided retrieval [24], in practice CRAQPOT relies on a reimplementation of TUUURBINE [12], a generic, ontology-guided case-based inference engine, using WIKITAAABLE[6] [4, 7] as its knowledge base. Our evaluation framework is based on the comparison of different adaptation approaches all using the same retrieval engine, so retrieval should not have a strong influence on the evaluation results.

4.3 Case adaptation

When a solution to a user query cannot be retrieved from the case base, adaptation is required, which in CRAQPOT begins with a substitution. If, for instance, the user wants a recipe for a carrot risotto and the case base does not contain one, the mushroom risotto recipe of figure 1 may be retrieved. The system will then adapt the retrieved recipe by replacing mushrooms with carrots, and making whichever modifications are necessary to the instructions to obtain a satisfactory result—for instance, adding the carrots earlier during the cooking because otherwise they would be too crunchy—as described into more details in [10].

Intuitively, this is done by finding the conjunction of the retrieved recipe modified by the necessary ingredient substitutions and of the domain knowledge that is available about the new ingredients—for instance, their required cooking time, which may be represented as

$$\text{cook carrots } ?^{=} \text{ 20 min} \quad \textit{Carrots cook for 20 minutes.} \tag{2}$$

Because a *qualitative* algebra is used, metric information must be specified with additional knowledge:

$$\begin{aligned} \text{2 min } ?^{<} \text{ 18 min} \quad & \textit{2 minutes are shorter than 18.} \\ \text{18 min } ?^{<} \text{ 20 min} \quad & \textit{18 minutes are shorter than 20.} \end{aligned} \tag{3}$$

Whenever adaptation is actually necessary, though, this will be because there is a contradiction between the retrieved recipe and the domain knowledge, and so there will be no conjunction. In the example given, replacing mushrooms with

[6] `http://wikitaaable.loria.fr`

carrots will expose a contradiction between "cook ~~mushrooms~~ carrots $?^{=}$ 2 min" from (1) and "cook carrots $?^{=}$ 20 min" from (2).

The workaround is to use belief revision theory [2] to make minimal modifications to the recipe in such a way that it becomes consistent with the domain knowledge, an approach that has already been used successfully for adaptation in CBR [6]. In the example this would, among other things, replace the last constraints of (1) with

$$\text{cook } \sout{\text{mushrooms}}\,\text{carrots } \{\mathsf{fi}^{>}\}\ \text{cook rice} \quad \begin{array}{l}\textit{Carrots start cooking} \text{ before}\\ \textit{rice and finish at the same time.}\end{array} \tag{4}$$

The implementation of a belief revision operator is a search algorithm that looks through the possible interpretations of the set of qualitative constraints that come from the domain knowledge to find those closest to the constraints of the source case. A set of constraints has an exponential amount of possible interpretations with respect to the number of intervals used in the case representation, therefore the search takes exponential time. We were able, though, to implement an approximation algorithm that reuses modified constraint satisfaction problem algorithms algorithms to obtain satisfactory results in polynomial time.

4.4 Text adaptation

Once the formal constraints have been adapted, the text must be modified to reflect the changes. The easiest solution, given that annotation rules exist that associate linguistic features to algebraic constraints, would be to use the inverse of those rules: given a constraint change, find the set of linguistic features that would have generated this constraint, and change the actual linguistic features of the text to reflect those. If the set of annotation rules were a bijection between the set of sets of possible linguistic features and the possible algebraic relations, this would be straightforward. But it is not, and therefore specific strategies are used to make approximate changes in text, with the objective always being to make the smallest possible changes, to limit the risk of introducing mistakes or diminishing the quality of the text.

Additionally, the implementation favours moving events such that they appear in the text in the order in which they begin, which minimises changes inside the sentences at the expense of maximising the movement of whole sentences.

With respect to the change described in (4), CRAQPOT makes the following modifications:

> Add the onion and cook until soft, about one minute. Meanwhile, slice the ~~mushrooms~~ carrots. Add them ~~two minutes before the end.~~ Add the rice and cook for two minutes, then add a glass of wine. Once the wine is evaporated, start adding broth, one ladleful at a time.

Observe that, because a qualitative algebra is used, it is not possible for the system to know, and therefore indicate, that the rice should be added two minutes after the carrots. This is a tradeoff for the algorithmic feasibility of the approach.

5 Simulating domain knowledge

One benefit of revision-based adaptation is that it can use whichever amount of domain knowledge is available. If no domain knowledge is available at all, the system will still work but give a result equivalent to null-adaptation. If complete domain knowledge is available, the system will give a result equivalent to a classic planning system. Any intermediary level of available domain knowledge will be used to improve the results of the adaptation.

The acquisition of domain knowledge for a case-based reasoning application falls outside the scope of this work. On the other hand, in order to get meaningful adaptation from CRAQPOT that makes it possible to evaluate the system, some quantity of knowledge is needed. We have therefore created a system to simulate domain knowledge on the fly.

While it would have been possible, for instance, to consider that the domain knowledge about the cooking of carrots is the disjunction of all the ways that carrots are cooked in our recipe base, this would have given little constrained knowledge, resulting in limited adaptations. For instance, we may have a recipe for a carrot salad in which the cooking time is 0 minutes, and one for a soup in which the cooking time is 60 minutes, which would suggest that any cooking time between 0 and 60 minutes is acceptable, with the effect that the mushroom risotto recipe would not be modified at all. We considered it would be more relevant for an evaluation of adaptation to use highly constrained knowledge, which requires a high adaptation effort.

Therefore, we have developed a system for on-the-fly extraction of relevant domain knowledge. This method can be seen as an additional retrieval stage, during which more cases are retrieved to be used in guiding adaptation. Given a recipe `Source` and a substitution $p \rightsquigarrow q$, a new recipe `KnowledgeSource` containing q is retrieved, such that ingredient q in this recipe is treated as much as possible in a similar way as ingredient p in `Source`.

For instance, if the user requests a carrot risotto recipe and a mushroom risotto recipe is retrieved, the system will attempt to retrieve some recipe with carrot and obtain carrot knowledge from it. Suppose that three recipes with carrots exist: a soup recipe in which carrots are cooked for one hour until they decompose in the broth, a salad recipe in which carrots are shredded and used raw, and a Asian recipe for sauteed pasta and vegetables. The system will retrieve the recipe in which carrots are cooked in the way most similar to how mushrooms are cooked in the mushroom risotto, which will be the Asian recipe. The way the carrot is used in this recipe, e.g. how it is cut and how long it is cooked, will become the carrot knowledge used to perform this adaptation, which is referred to in (2).

In our implementation, the distance function used is a Hamming distance: the distance between the recipe `Source` containing ingredient p and a candidate `KnowledgeSource` recipe containing ingredient q is the amount of actions applied to p in `Source` that are not applied to q in the `KnowledgeSource` candidate, plus the amount of actions applied to q in the `KnowledgeSource` candidate that are not applied to p in `Source`. An unweighted Hamming distance was chosen

because it makes the retrieval engine simple, but a different distance function may be desirable if not all actions or action substitutions are considered to be of equivalent importance.

Using this type of overly constrained domain knowledge can affect the outcome of the adaptation both positively and negatively. For instance, if the carrot recipe most similar to the mushroom risotto is in fact a carrot soup, it may be that our domain knowledge will demand for carrots to be cooked for one hour, and the adaptation result will suffer from this. On the other hand, the alternative, under-constrained approach could result in accepting a two-minute cooking time for carrots on the basis, for instance, of a carrot salad recipe in which carrots are not cooked at all.

6 Evaluation framework

This section presents the evaluation methodology we propose for adaptation in textual CBR. It is a comparative, blind, user-based approach: a user makes a query, and is shown a result obtained from one of various different adaptation techniques available. They are then asked to evaluate the result based on a set of criteria.

This type of evaluation is less time-consuming than an evaluation based on combinations of test queries and expected results, and we also think it is more accurate because it does justice to the creativity of the system, which may be able to provide results that did not occur to the designers of the test cases yet fully satisfy the users. With respect to user-based evaluation of workflow adaptation systems, it is also very advantageous in that the users need not be fluent in the formalism underlying the adaptation in order to be able to evaluate the system. As all user-based evaluation methodologies though, this evaluation is by definition a black box type evaluation: while we can know, for a given query–result pair, whether it gave satisfaction to the user, there is no automatic way to determine what went wrong in case it doesn't.

In the next subsections, the evaluation interface, the compared methods, and the evaluation criteria are shown.

6.1 Interface

A new user first needs to create an account, which is automatic if they are accessing the application from Facebook. They can then immediately make a query, as shown in figure 2.

6.2 Presentation of the adaptation methods

A method is selected randomly between the two control methods and the experimental method, and the query is processed with this method. If processing fails—which is theoretically possible only with the retrieval method—processing is transferred to another method.

The system has access to two control methods to answer requests:

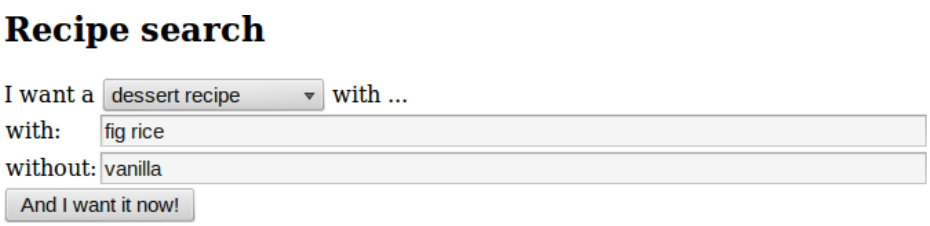


Fig. 2. CRAQPOT query interface.

- A retrieval-only method that performs no adaptation and fails when it is not possible to find a recipe that corresponds exactly to the query. Since CRAQPOT uses the same textual case base, the retrieval system defines the maximum possible scores that could be obtained in text and recipe quality: our proposal is not expected to adapt recipes in such a way that the result is better than the original.
- A method based on a reimplementation of COOKIIS text adaptation, which performs a smart string replacement [21]. This provides a baseline: given the simplicity and efficiency of COOKIIS text adaptation, our proposal would be difficult to justify if it did not offer better quality.

Applying the query from the running example (a carrot risotto) to the retrieval-only method would fail—the query would then be handled by one of the other methods. Applying the same to the COOKIIS method would work, but the instructions would only be modified insofar as the word "mushrooms" would be replaced with the word "carrots", resulting in a two minute cooking time for carrots: "Meanwhile, slice the ~~mushrooms~~ carrots. Add them two minutes before the end."

6.3 Evaluation questions

The result is then presented to the user, as shown in figure 3. The user has no way of knowing which method was used to process their query.

Before the user can make a new query, they are asked to tell how much they agree with the following statements:

- "This recipe seems tasty." We postulate that this provides the most relevant indicator to evaluate the quality of the *content* adaptation of the recipe—independantly of the way it is written.
- "This text is well written." We postulate that this provides the most relevant indicator to evaluate the quality of the *textual* adaptation of the recipe.
- "This recipe fits my query." We postulate that this provides a general indicator as to whether the adaptation approach used was appropriate.

Users rate their degree of agreement on a 4-point Likert scale—where 1 indicates strong disagreement and 4 indicates strong agreement.

The hypotheses we made are:

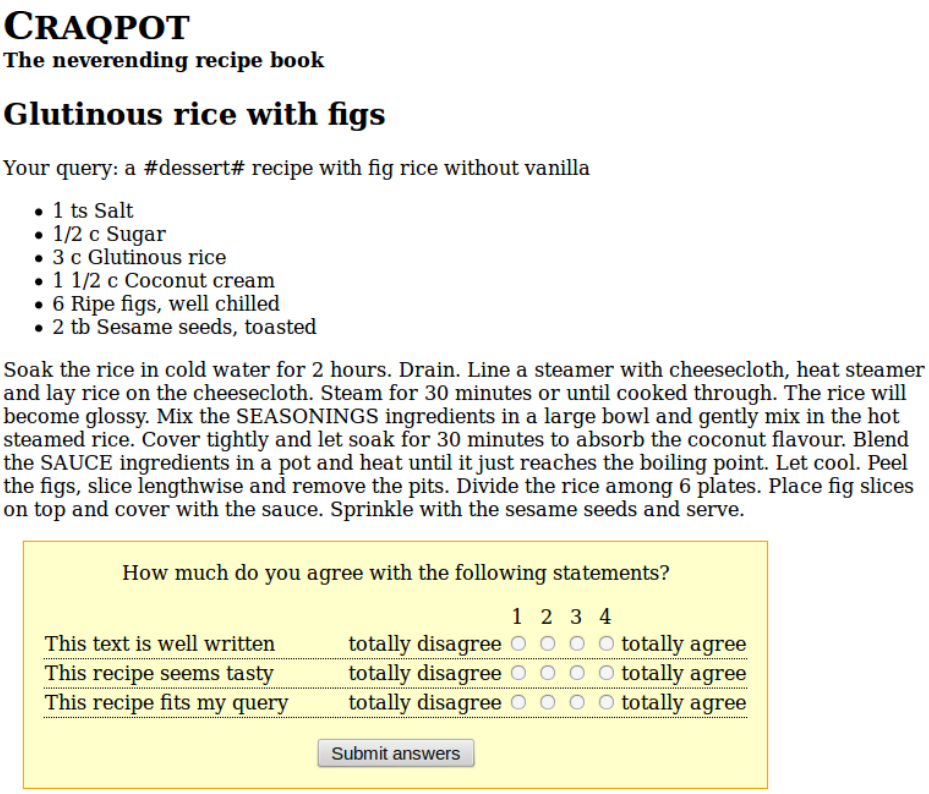


Fig. 3. CRAQPOT response and evaluation interface. This screenshot shows a recipe that was adapted by the COOKIIS method.

H$_1$ CRAQPOT and COOKIIS will output lower text and recipe quality, and lower fitness—i.e. lower score on the third criterion—than simple retrieval.
H$_2$ CRAQPOT will output higher recipe quality than COOKIIS.
H$_3$ CRAQPOT and COOKIIS will output a similar text quality.
H$_4$ CRAQPOT will leave its users with a better impression that the answer fits the query than COOKIIS.

We postulate H$_1$ because of the inherent risk of automatic adaptation, H$_2$ because CRAQPOT, unlike COOKIIS, integrates domain knowledge, H$_3$ because the risk of adaptation is mitigated by finer linguistic processing, and H$_4$ because the adaptation is less superficial. The null hypothesis H$_0$ is that all three systems are comparable and any difference in score would be the result of chance.

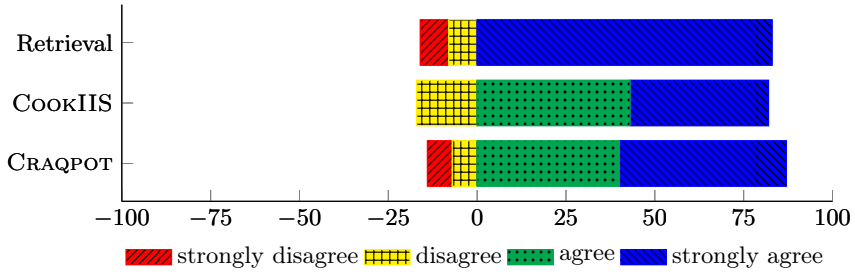
7 Results

Raw results based on 9 users performing 50 queries are shown in figure 4. Wilcoxon signed-rank tests were performed for each criterion to compare methods pairwise and measure the probability that the observed differences in scores are the result of chance. The resulting p-values thresholds are shown in table 1. It is commonly assumed that p-values between .05 and .1 offer a weak presumption against H$_0$, whereas p-values below .05 offer a strong presumption against H$_0$. Values below .01 offer a very strong presumption.

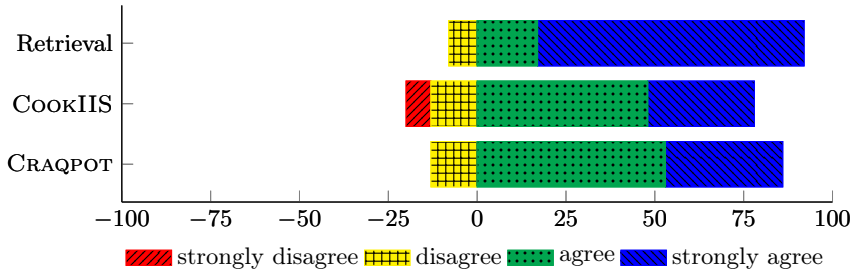
	Text quality		Recipe quality		Fitness	
	COOKIIS	CRAQPOT	COOKIIS	CRAQPOT	COOKIIS	CRAQPOT
Retrieval	$< .05$	$< .1$	$< .01$	$< .05$	$< .001$	$> .5$
CRAQPOT	$> .5$		$< .5$		$< .001$	

Table 1. Significance of the pairwise method comparisons. For instance, "$< .05$" at the intersection of "Retrieval" and "COOKIIS" below "Text quality" means that the difference in text quality between the retrieval and the COOKIIS method has a probability $p \leq 5\%$ of being due to chance. The table is arranged in such a way that the system named in the row header systematically gives better results than the one named in the column header.

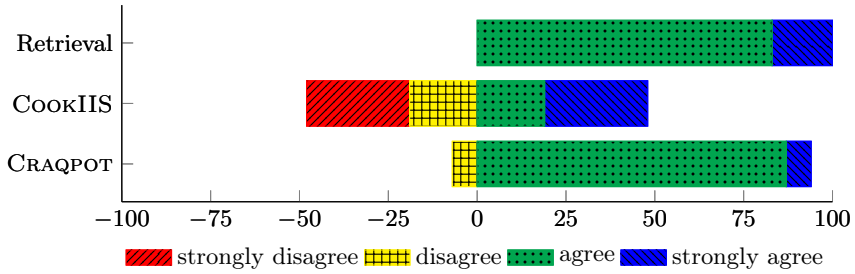
In all three indicators, retrieval ranked first, and CRAQPOT ranked second. As expected, CRAQPOT's and COOKIIS overall performance is worse than simple retrieval, partially validating H$_1$. This is mitigated, though, by the fact that simple retrieval was able to process only 46% of the queries assigned to it. In recipe quality, retrieval performed significantly better than CRAQPOT and strongly significantly better than COOKIIS, but the score difference between CRAQPOT and COOKIIS, while important, was not significant: the exact p-value is .30. This indicates a 30% chance that CRAQPOT's better scores with respect to COOKIIS were the effect of chance, and therefore H$_2$ is not supported by the results. Further evaluations may change this. In text quality, retrieval performed strongly



(a) Results for criteria: "This text is well written."



(b) Results for criteria: "This recipe seems tasty."



(c) Results for criteria: "This recipe fits my query."

Fig. 4. Detailed user evaluations.

significantly better than CRAQPOT and very strongly significantly better than COOKIIS. Although the evaluations surprisingly show that CRAQPOT did better than COOKIIS, the difference is not statistically significant, confirming H_3. In fitness, CRAQPOT performed just as well as retrieval, and both methods were very strongly significantly better than COOKIIS, confirming H_4.

The few available evaluations make it possible to claim that H_3 and H_4 are verified, and that H_1 is partially verified. H_2 is not verified but more evaluations will be necessary.

8 Conclusion

We have presented CRAQPOT, a CBR system that retrieves and adapts processes represented as instruction texts and the evaluation methodology we developed to evaluate it.

The evaluation approach we propose has many benefits with respect to existing approaches used in textual and process-oriented CBR. Compared to sets of queries–expected results that are often used, it is much less labour-intensive to put in place, and it gives value to creative solutions proposed by a system. Additionally, because we provide the results as text, we can rely on domain experts that are not fluent in the formalism to provide evaluations. Yet, we are able to obtain a separate evaluation for the textual and for the underlying formal case adaptation. Certain details about the evaluation process are specific to the adaptation of processes, but we believe with further work it would be easy to redefine our methodology in a more generic way for various textual CBR applications.

The evaluation results for our application, CRAQPOT, were mostly satisfactory, although more evaluations would be needed to obtain stronger statistical significance. There are many benefits from developing CRAQPOT as a Facebook application that have been left as future work: for instance, the possibility of using the user's timeline in order to advertise for the application, and even obtain multiple evaluations for the same query, using the different methods, by appealing to their network.

We also proposed a system based on case-based retrieval for on-the-fly extraction of relevant knowledge. This system made it possible to evaluate our application without having to specify complete domain knowledge. We also think that, given further study to make it more generic, this approach could actually be developed into a fully adequate way of integrating the experience of many cases in order to adapt a source case.

References

1. Adeyanju, I., Wiratunga, N., Lothian, R., Sripada, S., Lamontagne, L.: Case retrieval reuse net (CR2N): An architecture for reuse of textual solutions. In McGinty, L., Wilson, D.C., eds.: Case-Based Reasoning Research and Development. Volume 5650 of Lecture Notes in Computer Science. Springer (2009) 14–28

2. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: Partial meet contraction and revision functions. The Journal of Symbolic Logic **50**(2) (1985) 510–530
3. Allen, J.F.: Maintaining knowledge about temporal intervals. Communications of the ACM **26**(11) (1983) 832–843
4. Badra, F., Cojan, J., Cordier, A., Lieber, J., Meilender, T., Mille, A., Molli, P., Nauer, E., Napoli, A., Skaf-Molli, H., Toussaint, Y.: Knowledge acquisition and discovery for the textual case-based cooking system WikiTaaable. In Delany, S.J., ed.: ICCBR 2009 Workshop Proceedings. (July 2009) 249–258
5. Brill, E.: A simple rule-based part of speech tagger. In: Workshop on Speech and Natural Language, Association for Computational Linguistics (1992) 112–116
6. Cojan, J., Lieber, J.: Applying belief revision to case-based reasoning. In: Computational Approaches to Analogical Reasoning: Current Trends. Springer (2014) 133–161
7. Cordier, A., Dufour-Lussier, V., Lieber, J., Nauer, E., Badra, F., Cojan, J., Gaillard, E., Infante-Blanco, L., Molli, P., Napoli, A., Skaf-Molli, H.: TAAABLE: a Case-Based System for personalized Cooking. Studies in Computational Intelligence. In: Successful Case-based Reasoning Applications. Springer Berlin Heidelberg (2013) In press.
8. Dojat, M., Ramaux, N., Fontaine, D.: Scenario recognition for temporal reasoning in medical domains. Artificial Intelligence in Medicine **14**(1–2) (1998) 139–155 Selected Papers from AIME '97.
9. Dufour-Lussier, V.: Spatial-temporal qualitative reasoning from textual cases. PhD thesis, Université de Lorraine (2014)
10. Dufour-Lussier, V., Le Ber, F., Lieber, J., Martin, L.: Adapting spatial and temporal cases. In Agudo, B., Watson, I., eds.: Case-Based Reasoning Research and Development (ICCBR 2012). Volume 7466 of Lecture Notes in Computer Science. Springer (2012) 77–91
11. Dufour-Lussier, V., Le Ber, F., Lieber, J., Nauer, E.: Automatic case acquisition from texts for process-oriented case-based reasoning. Information Systems (mar 2014) 153–167
12. Gaillard, E., Infante-Blanco, L., Lieber, J., Nauer, E.: Tuuurbine: A generic CBR engine over RDFS. In: Case-Based Reasoning Research and Development. Springer (2014) 140–154
13. Gervás, P., Hervás, R., Recio-García, J.A.: The role of natural language generation during adaptation in textual CBR. In: 4th Workshop on Textual Case-Based Reasoning: Beyond Retrieval (ICCBR 2007). (2007) 227–235
14. Lamontagne, L., Lapalme, G.: Textual reuse for email response. In Funk, P., González Calero, P.A., eds.: Advances in Case-Based Reasoning. Volume 3155 of Lecture Notes in Computer Science. Springer (2004) 242–256
15. Ma, J., Knight, B.: A framework for historical case-based reasoning. In Ashley, K.D., Bridge, D.G., eds.: Case-Based Reasoning Research and Development. Volume 2689 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2003) 246–260
16. Minor, M., Bergmann, R., Görg, S., Walter, K.: Towards case-based adaptation of workflows. In Bichindaritz, I., Montani, S., eds.: Case-Based Reasoning. Research and Development. Volume 6176 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2010) 421–435
17. Minor, M., Islam, M., Schumacher, P.: Confidence in workflow adaptation. In Agudo, B., Watson, I., eds.: Case-Based Reasoning Research and Development.

Volume 7466 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 255–268
18. Minor, M., Montani, S., Recio-García, J.A.: Process-oriented case-based reasoning. Information Systems **40**(0) (2014) 103–105
19. Minor, M., Schmalen, D., Bergmann, R.: XML-based representation of agile workflows. In Bichler, M., Hess, T., Krcmar, H., Lechner, U., Matthes, F., Picot, A., Speitkamp, B., Wolf, P., eds.: Multikonferenz Wirtschaftsinformatik, GITO-Verlag Berlin (2008) 439–440
20. Müller, G., Bergmann, R.: Workflow streams: A means for compositional adaptation in process-oriented CBR. In: Case-Based Reasoning Research and Development. Springer (2014) 315–329
21. Newo, R., Bach, K., Hanft, A., Althoff, K.D.: On-demand recipe processing based on CBR. In: ICCBR 2010 Workshop Proceedings. (2010) 209–218
22. Pujari, A.K., Kumari, G.V., Sattar, A.: INDU: An interval & duration network. In Foo, N., ed.: Advanced Topics in Artificial Intelligence. Volume 1747 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (1999) 291–303
23. Schumacher, P., Minor, M., Schulte-Zurhausen, E.: On the use of anaphora resolution for workflow extraction. In Bouabana-Tebibel, T., Rubin, S.H., eds.: Integration of Reusable Systems. Volume 263 of Advances in Intelligent Systems and Computing. Springer International Publishing (2014) 151–170
24. Smyth, B., Keane, M.T.: Adaptation-guided retrieval: Questioning the similarity assumption in reasoning. Artificial Intelligence **102**(2) (1998) 249–293
25. Sánchez-Marré, M., Cortés, U., Martínez, M., Comas, J., Rodríguez-Roda, I.: An approach for temporal case-based reasoning: Episode-based reasoning. In Muñoz-Ávila, H., Ricci, F., eds.: Case-Based Reasoning Research and Development. Volume 3620 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2005) 465–476
26. Weber, R.O., Ashley, K.D., Brüninghaus, S.: Textual case-based reasoning. Knowledge Engineering Review **20**(3) (2005) 255–260