On-Demand Recipe Processing based on CBR

Régis Newo, Kerstin Bach, Alexandre Hanft, and Klaus-Dieter Althoff

Intelligent Information Systems Lab
University of Hildesheim
Marienburger Platz 22
31141 Hildesheim, Germany
{lastname}@iis.uni-hildesheim.de

Abstract. The third version of CookIIS, our candidate for the third Computer Cooking Contest (CCC) is focusing on pre-processing and enriching the source data for a better performance of the 4R processes. Our goal for this year is to improve the *retrieval* process by considering the ingredients' weights, strengthen the *reuse* phase by enhancing the knowledge model and further developing the adaptation methods, introducing light *revision* approaches of retrieval results. This paper presents the improvement of CookIIS in preparation for the CCC-2010.

1 Introduction

CookIIS is a Case-Based Reasoning system realized in the cooking domain for providing recipes according to given user preferences. It is a research prototype that deals with a given standardized case base of 1,484 recipes. Each recipe consists of a title, a list of ingredients and the preparation advices. We represent each recipe as one case. In terms of CBR, the list of ingredients serve as problem description and the recipe's title as well as the preparation is the according solution. Nevertheless, usually only a subset of ingredients is given by a user and it is CookIIS's task to find the most appropriate recipe to the given desires.

The CCC takes place the third time and the challenges evolved in the comparison to the two competitions. There are four challenges this year[1]:

The *Main Challenge* focuses on retrieving recipes according to given constrains, like the ingredients, the type of ingredients, the type of dish, the provenance of a dish as well as some dietary practices. A possible question to be answered in this challenge might be "I would like to cook a Mediterranean fish dish as starter, but please avoid lemon and other citrus fruits". Since all competitors work with the same case base, the requests do not have a perfect match in the case base and it is the system's task to modify the recipes in order to match best.

The *Adaptation Challenge* this year focuses on a specific adaptation task that everybody might know from personal experiences. Once you have found a great sounding recipe and start cooking you figure out that you do not have all

[1] `http://vm.liris.cnrs.fr/ccc2010/doku.php?id=rules`

the required ingredients available. This year, the participating systems should be able to consider this information and help the amateur chef to improvise and resolve problems like "I have retrieved the Banana Butterfinger Cake recipe, but I have no sour cream or vanilla. What should I do?".

A novelty of this years CCC is the *Open Challenge* where each team can show their creativity and moreover present what technology today can do. We decided to take numeric quantities for the retrieval process into account as well as we further developed the creation of menus since this has been a challenge in the last two years and we still have some ideas on this topic.

Another new aspect of this year's CCC is the *Student Challenge* that features student teams to enter the contest and compete with other student teams. This year, our team mainly consists of PhD students and research assistants, so CookIIS will not compete in this challenge. Like the other years, CookIIS is a web-based application and is available and can be tested by visiting the CookIIS web page at the University of Hildesheim's IIS lab[2].

The remaining paper is structured as follows: Section 2 introduces the architecture of CookIIS featuring the underlying knowledge model, retrieval processes, similarity measures and workflows of the system. The following three sections focus on the novelty of this year's system. Section 3 describes the pre-processing that enables the consideration of the ingredients' amounts; section 4 introduces the adaptation processes including the already existing ones and its improvements. Section 5 shows how user feedback can influence to retrieved recipes. The experimental results including the system's responses for a set of training queries is presented in section 6, before the final section sums up the paper and gives an outlook on future work.

2 Architecture of CookIIS

Before we explain the CBR processes implemented in CookIIS, we will first present in this section how the system is built. CookIIS is built using an industrial strength tool called empolis Information Access Suite (e:IAS). The architecture of our system therefore leans on the architecture of e:IAS, which can be seen in Figure 1.

The bottom of the architecture shows the data sources. In CookIIS these are the recipes given by the CCC organizers that we have split up so each recipe is in one XML file and further preprocessed as described in Section 3. e:IAS then uses our defined knowledge model and similarity measures in order to process the user's queries. The retrieval process is defined in the search engine and it can be supported/assisted by rules and a text miner. We will now give a short explanation of some parts of the architecture.

[2] `http://cookiis2010.iis.uni-hildesheim.de/ccc`

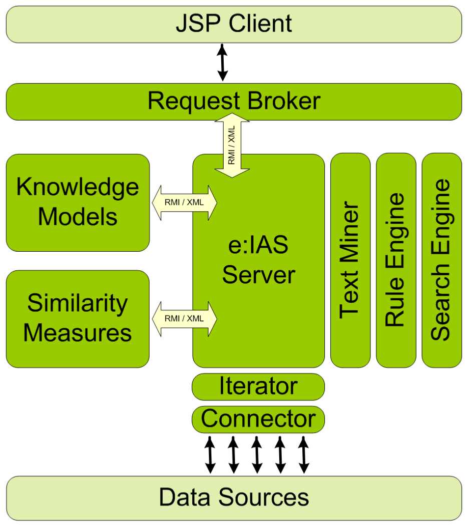


Fig. 1. The architecture of e:IAS used for CookIIS

2.1 Knowledge Model

Our knowledge model contains different elements. The main elements are ingredients (about 1000 in our model) which are organized in the following eleven classes:

Basic Ingredients	Fish	Meat	Vegetable
Supplement	Fruit	Drinks	Milk
Intermediate Ingredients	Oil and Fat	Spice and Herb	

Each ingredient is modeled as a concept with possible synonyms in English and German. The concepts are mostly organized as taxonomies and each class can contain several taxonomies. One can for example organize meat by part (fillet, haunch) or by kind (pork, beef).

Furthermore, our model contains several rules which are used for the computation of meta information like:

– the type of meal,
– the type of cuisine,
– the consideration of some diets (for example low cholesterol).

Further details on the knowledge model as well as the computation of meta information can be found in [1, 2].

2.2 Similarity Measures

Within CookIIS, similarities between recipes (or between a query and a recipe) are computed in two steps. First, a local similarity measure is defined for each class of the knowledge model in order to compare the different types of ingredients. The underlying taxonomies give a strong basis for the computation of the similarities between ingredients within a class. Here, adequate values are given for the generalization and the specialization step so that the similarities can be automatically computed. These values indicate the similarity between a concept and a child or a parent. In addition to the taxonomy based similarity measures, we also used table based similarity measures for some classes, because the values computed with taxonomies do not always reflect the reality in our opinion. The similarity between some pairs of elements is (manually) entered in the corresponding matrix. When several local similarity measures are defined within a class, the highest similarity value is always used.

In the second step, we use a global similarity measure to compute the similarity between cases (i.e. the recipes). It is a weighted sum of the local similarities of the attributes in the cases, following the local-global principle for similarity modeling [3]. The global similarity measure do not only consider the ingredients of the recipes, but also other attributes like the computed meta information mentioned in the previous Section.

2.3 Indexing and Searching

Indexing and Searching recipes in CookIIS is done using workflow-like constructs, called *pipelines* in e:IAS. The pipelines consist of the so-called *pipelets* which represent the different steps (actions) for each operation.

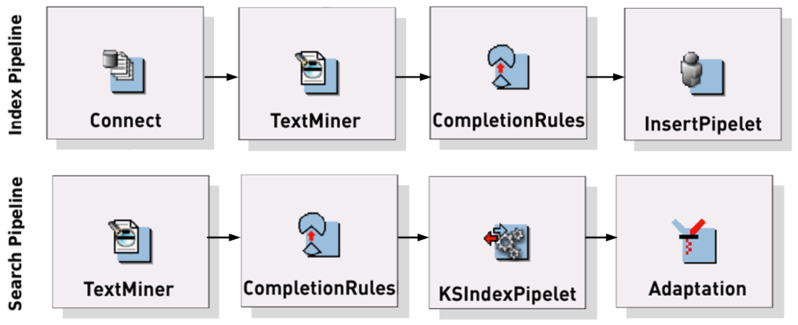


Fig. 2. The index and search pipeline for CookIIS

The Index Pipeline The aim of the index pipeline is to create an index representing the case base (i.e., insert the recipes in the case base). This is done by performing the following actions (as depicted on the left hand side of Figure 2): First a connection to the data sources has to be established before an analysis of the data and the extraction of the concepts by the text miner is carried out. The third pipelet computes of meta information based completion rules and the final pipelet inserts the cases in the retrieval structure.

The Search Pipeline The aim of the search pipeline is to retrieve and adapt the recipes from the case base according to a query. The actions performed are the following (right hand side on Figure 2): First, the query is analyzed and concepts are extracted using the text miner. The following pipelet computes meta information based on the input given by the query. The third pipelet executes the retrieval of the most similar cases from the case base with the aforementioned similarity measure. The final piplet carries out the adaptation of the retrieved cases (this is done in many steps as explained in Section 4).

3 Recipe Pre-Processing

The information given in recipes still has potential to improve the CookIIS system. There are different ways of using the case base obtaining knowledge that can be provided in within the knowledge containers. This point has also been discussed by the Taable team [4] as well as the JaDaCook team [5]. For example, we have applied association rules aiming at adaptation knowledge, but caused by the high variation we only received rules containing very obvious associations, like eggs and sugar usually come with flour. We think that a higher amount of recipes is required for acquiring more special association rules.

The global similarity measure of the last versions of CookIIS only takes the ingredient without the according amount into account. Hence, recipes that contain many ingredients are retrieved more often because the probability that one of the containing ingredients was requested is higher. However, users that are looking for recipes usually name those ingredients that should be characteristic for the whole dish: when looking for a recipe with carrots you might not be satisfied with a casserole that includes half of a carrot. Therefore, we are taking the amounts of the ingredients into account.

3.1 Pre-processing Process

We decided to enrich the information given in the case base including an attribute that marks the major ingredients of a recipe. Since the amounts are given in various measures, we built up a dictionary that normalizes each amount to milligram. Further, we used the normalized amount information for improving the retrieval aiming at retrieving recipes that contain the requested ingredient as a major ingredient. Algorithm 1 shows how the major ingredients of a recipe are determined.

Given the normalized amount of each ingredient, we first determine the median of the amounts occurring in the recipe. The median splits the ingredients into major and minor ingredients of a recipe. In the third step we mark all major ingredients by adding the attribute "major" to the ingredients tag. In the last

Algorithm 1 Marking Major Ingredients

```
1: Order ingredients descending according to its amounts of one recipe
2: Calculate the median over all amounts of one recipe
3: Mark all ingredients with $amount > median$ as major
4: Mark the position of major ingredients according to their amounts
```

step we determine the order (based on the ingredients' amounts) of the major ingredients and assign it to the ingredients. This results in an enhanced ingredient tag in the xml source file, like the following:

<IN amnt='425243' major='5'>1 cn (15oz) tomato sauce</IN>

Obviously this information can be used in various ways. However, currently we only use it within the retrieval process. For that purpose, we doubled the weight of major ingredients within the global similarity calculation. In the similarity assignment, we have an attribute that represents a set of major ingredients, so each major ingredient that has been requested gets the double weight. As a result, the retrieved recipes containing the desired ingredients are more emphasized. We are aware that for instance spices and herbs, which have a huge influence on a recipe, but only occur in small amounts, are not considered in this approach. However, in the CookIIS knowledge model and within the global similarity calculation spices and herbs are treated separately.

4 Adaptation

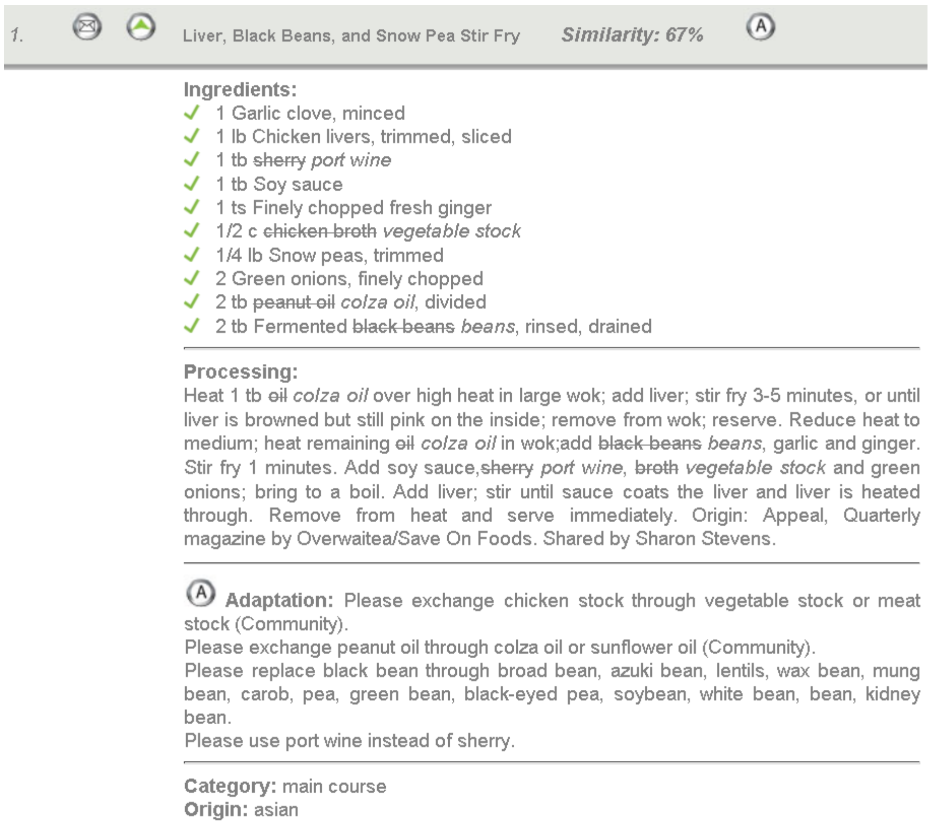
The given recipe base contains 1484 recipes, which are not enough recipes for the whole variety of desired as well as unwanted ingredients. Therefore, an adaptation of the queried recipes to the users needs is necessary.

This section describes three different kind of adaptations we use in CookIIS. Two approaches are known from former versions of CookIIS: the model-based and the community-based adaptation. Furthermore, we implemented an in-place adaptation and an adaptation approach regarding user feedback.

All of our adaptation approaches share some assumptions and have a certain generality in the used methods with the aim to be able to transfer these methods to other domains. First, we denote ingredients that are excluded by a diet or explicitly by the user as *forbidden* ingredients. If a *forbidden* ingredient occurs in a retrieved recipe we consider it as *critical* (for this recipe) because it has to be omitted or replaced. Second, we restrict the replacements to the same class as the replaced ingredient to assure certain a level of applicability.

4.1 Integration of the three Approaches

All three adaptation approaches are performed sequentially and each of them uses the results of the preceding one. First, the community-based adaptation is



1. Liver, Black Beans, and Snow Pea Stir Fry **Similarity: 67%**

Ingredients:
- ✓ 1 Garlic clove, minced
- ✓ 1 lb Chicken livers, trimmed, sliced
- ✓ 1 tb ~~sherry~~ *port wine*
- ✓ 1 tb Soy sauce
- ✓ 1 ts Finely chopped fresh ginger
- ✓ 1/2 c ~~chicken broth~~ *vegetable stock*
- ✓ 1/4 lb Snow peas, trimmed
- ✓ 2 Green onions, finely chopped
- ✓ 2 tb ~~peanut oil~~ *colza oil*, divided
- ✓ 2 tb Fermented ~~black beans~~ *beans*, rinsed, drained

Processing:
Heat 1 tb ~~oil~~ *colza oil* over high heat in large wok; add liver; stir fry 3-5 minutes, or until liver is browned but still pink on the inside; remove from wok; reserve. Reduce heat to medium; heat remaining ~~oil~~ *colza oil* in wok;add ~~black beans~~ *beans*, garlic and ginger. Stir fry 1 minutes. Add soy sauce,~~sherry~~ *port wine*, ~~broth~~ *vegetable stock* and green onions; bring to a boil. Add liver; stir until sauce coats the liver and liver is heated through. Remove from heat and serve immediately. Origin: Appeal, Quarterly magazine by Overwaitea/Save On Foods. Shared by Sharon Stevens.

Adaptation: Please exchange chicken stock through vegetable stock or meat stock (Community).
Please exchange peanut oil through colza oil or sunflower oil (Community).
Please replace black bean through broad bean, azuki bean, lentils, wax bean, mung bean, carob, pea, green bean, black-eyed pea, soybean, white bean, bean, kidney bean.
Please use port wine instead of sherry.

Category: main course
Origin: asian

Fig. 3. Replacement of forbidden concepts (crossed out) with new replacements (italicised) in the ingredient list and preparation: sunflower oil replaces olive oil in the ingredient list and only oil in the preparation because olive oil does not exist there.

executed. For those forbidden ingredients where no community-based replacements can be offered, the standard model-based adaptation is carried out afterwards. Both approaches only add adaptation advice in an extra text attribute of the retrieved recipes but leave all attributes for ingredients and preparation unchanged. In contrast, the third approach changes the ingredient list and preparation text visible according to the advice of both preceding approaches.

The integration of the all approaches is realised with three subsequent pipelets within the the Search Pipeline used for retrieval and reuse. A user-defined pipelet for the community adaptation is inserted into this pipeline which is followed by the a standard AdaptationRulesPipelet implementing the model-based adaptation. The in-place adaptation as third approach is realised as another user-defined pipelet and integrated after the AdaptationRulesPipelet in the same pipeline. All approaches are described in the next sections.

4.2 Model-based and Community-Based Adaptation

For the community-based adaptation we collect lots of adaptation advices from comments of recipes within a large cooking community using the knowledge model of CookIIS. With the aim of this model we classify each ingredient of the comment as OLD (if it exits in the recipe already) or NEW (if it exists only in the comment). Afterwards we interpret the co-occurrence of a OLD and a NEW ingredient belonging to the same ingredient class as an adaptation advice and sum them up over all 70.000 recipes. This results in 2967 adaptation suggestions for 410 different ingredients. If there exist a suggestion for a forbidden ingredient, the two most frequent adaptations are presented. A more detailed description and evaluation of this approach is given in [6].

In contrast to the community-based adaptation, which delivers adaptation suggestions for 40% of the ingredients, the model-based adaptation works for all concepts, based on the similarity between ingredients. In general, it determines similar ingredients to the *forbidden* ingredients and suggest them as replacement. The similarity between two ingredients is based on the knowledge model that are applied along with thresholds to determine nephew and sibling concepts as replacements [2].

4.3 In-place Adaptation

Both aforementioned approaches only add adaptation advices in an extra text attribute, but did not change the original attributes for ingredients and preparation which makes it harder to take the ingredient list as shopping list or work through the preparation step by step. Therefore we add an additional adaptation component which executes these adaptation advices directly on the original recipe. By investigating the preparation instructions we detect the following difficulties that prevent using a simple string matching: ingredients are mentioned in their (irregular) plural form, referred in an abbreviated form or more general term or described by their synonyms. Additionally, the user should recognise in the modified recipe what the changes are, especially if an adaptation suggestion is not good as expected by the user. Therefore, the in-place adaptation has to tackle two problems: Simple string matching for the replacement often does not succeed and the original ingredient should be visible if one of the replacement suggestions is not accepted by the user.

The replaced and the new text are marked up with HTML tags in order to make changes on the original text of the case visible. Hence, the user can notice the original ingredient. To identify concepts that should be replaced in the given text four consecutive possibilities are tested until one is successful:

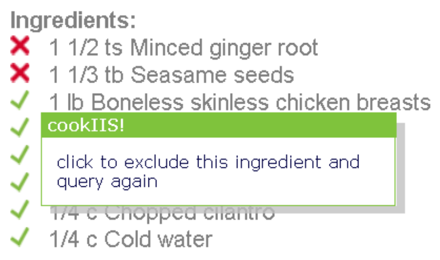
1. find the concept name in the text
   (a) search and replace plural form (considering -es, -ies, -ves and -oes) and
   (b) search and replace singular form,
2. consider common synonyms belonging to certain concepts,
3. if it is a 2-word-concept, replace only the last word, otherwise

4. look for the class name of this concept and replace if exists.

Figure 3 shows an example where "black bean*s*" (1.b) are replaced by "beans" (1.a). In the preparation only "oil" instead of *forbidden* "peanut oil" (3.) occurred and is replaced by "colza oil" (from Community). Furthermore, "broth" in the preparation stands for "chicken stock" (2.) and is replaced by "meat extract".

5 Revise through User Interaction

During a small evaluation we investigated that users often discover that they either dislike or do not have certain ingredients available that are required for preparing the desired recipe. To make this more faster and comfortable the user can now click on an icon before each ingredient to exclude this ingredient instead of adding the text by hand into the exclude text box in the GUI. At start, each ingredient has a green check mark, which is changed into a red cross if the user clicks on it to exclude this ingredient. In a subsequent search request these ingredients where excluded as well. According to the fact that in our current implementation the amount of forbidden ingredients in an recipe did not affect the similarity of a recipe to a query, the same recipes would be retrieved again, but with additional adaptation suggestions. Of course, the exclusion can be undone by clicking again on the icons. The complete functionality is realised with Javascript called by the JSP code of website. Figure 4 shows a scenario where the user has excluded the ingredients minced ginger an Sesame seeds after the first retrieval.


Ingredients:
- ✗ 1 1/2 ts Minced ginger root
- ✗ 1 1/3 tb Seasame seeds
- ✓ 1 lb Boneless skinless chicken breasts
- ✓
- ✓
- ✓
- ✓ 1/4 c Chopped cilantro
- ✓ 1/4 c Cold water

cookIIS!

click to exclude this ingredient and query again

Fig. 4. Feedback: The user can easily exclude additional ingredients by clicking on the green icon in front of each ingredient

6 Experimental Results

As stated earlier, many features of CookIIS have been improved (from the previous versions), like for example, the computation of meta information (type of meal, type of cuisine), the adaptation of recipes (model based, community based and in-place). We will concentrate in this Section on two new features of CookIIS, namely the impact of the preprocessing of the data source and the revise step in CookIIS.

We will consider for this purpose the query "I want a dessert with strawberries". CookIIS returns many recipes which are 100% similar to the query because they contain strawberries and were tagged as desserts. These recipes are automatically ranked according to the amount of strawberries contained (in comparison to the remaining ingredients). CookIIS thus ranks the recipe "My Strawberry Pie" (which contains few ingredients) higher than the "Summertime Strawberry Gelatin Salad" which contains much more ingredients (because strawberry is this case less important). Furthermore, if an ingredient is not available, the user can explicitely select the failing or unwanted ingredients as can be seen in Figure 4. Gelatin for example in the "Summertime Strawberry Gelatin Salad" is then replaced through agar-agar.

7 Conclusion

The paper presents new developments of the CookIIS system - mainly the preprocessing for taking ingredients' amounts into account, applying adaptation knowledge to recipes and replacing the modified ingredients within the recipe as well as enabling user interaction for interactive search refining.

However, we have addressed the first three of the four CBR processes and since the case base is standardized and static, the only way for retention is acquiring and updating modification knowledge, similarity and vocabulary knowledge.

Future work will be focused on including the amount information, now available for each recipe, in the adaptation process as well. This also affects the substitution of ingredients or a modification that leaves out certain ingredients instead of looking for a substitute.

References

1. Ihle, N., Newo, R., Hanft, A., Bach, K., Reichle, M.: Cookiis - A Case-Based Recipe Advisor. In Delany, S.J., ed.: Workshop Proceedings of the 8th International Conference on Case-Based Reasoning, Seattle, WA, USA (July 2009) 269–278
2. Hanft, A., Newo, R., Bach, K., Ihle, N., Althoff, K.D.: Cookiis - a successful recipe advisor and menu advisor. In Montani, S., Jain, L., eds.: Successful Case-based Reasoning applications. Springer (2010)
3. Bergmann, R.: Experience Management: Foundations, Development Methodology, and Internet-Based Applications. Volume 2432 of LNAI. Springer-Verlag (2002)
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 Proc. (2009)
5. Herrera, P.J., Iglesias, P., Sanchez, A.M.G., Diaz-Agudo, B.: Jadacook 2: Cooking over ontological knowledge. In Delany, S.J., ed.: ICCBR 2009, Workshop Proc. (2009)
6. Ihle, N., Hanft, A., Althoff, K.D.: Extraction of adaptation knowledge from internet communities. In Delany, S.J., ed.: ICCBR 2009, Workshop Proc. (2009) 35–44