Refinements for Retrieval and Adaptation of the CookIIS application

Alexandre Hanft, Norman Ihle, and Régis Newo

Intelligent Information Systems Lab
University of Hildesheim
{hanft, ihle, newo}@iis.uni-hildesheim.de

Abstract: In this paper we present parts of our CookIIS system, a CBR-based application which provides and modifies cooking recipes. We participated at the 1st Computer Cooking Contest and won the menu challenge with CookIIS. After introducing the workflows used in this application, we show how the different tasks can be refined in order to improve the processes of the retrieval with custom components and the modification of recipes with sequential adaptation. We intend to participate at the next computer cooking contest with the improved application.

1 Introduction

The CookIIS application [HIB+08] is a structured CBR-based application that focuses on the retrieval and adaptation of given cooking recipes. The application which is based on the empolis Infomation Access Suite [emp05], an industrial strength CBR tool suite, took successfully part in the Computer Cooking Contest.

The Computer Cooking Contest[1] (CCC) was established as a competition to attract more students for AI and especially CBR as well as having a comparison of different implementations and teams. The second aim was to have a popular demonstration application usable and understandable by everyone. The technology was explicitly not restricted to the usage of CBR technology, but nevertheless all finalist teams used CBR. The 1st Computer Cooking Contest was co-located at the ECCBR'08 [ABMH08] in Trier, Germany and was accompanied by a scientific CCC workshop. Each system has to prove in three competition rounds [SMT08]: Compulsory task, Negation challenge, Menue challenge, where the CookIIS team won the Menu Challenge. The evaluation was done according to scientific, technical and culinary criteria, with the latter one being executed by the star-rated chef Wolfgang Becker from Trier. Moreover, through the competition the attention of other people was caught and they gave valuable feedback.
In this paper we present parts of CookIIS and discuss some approaches for the refinement and improvement of the application, especially in regard to retrieval and adaptation. An important part of the development was the design of specific workflows to organise the order of steps needed to present a result. To introduce our application we will present the workflows used in CookIIS. We will take a closer look on some of the tasks of the workflows: the extraction of meta-information using a rule engine and the re-

[1] http://computercookingcontest.net

trieval. For the retrieval the similarity modelling is a central task. A similarity measure for a set of sets is missing in the used tool suite. We present a custom similarity that takes the number of elements into account. Next we focus on the adaptation of recipes which was a required task at the Computer Cooking Contest. We present our approach to replace unwanted or otherwise excluded ingredients with similar ones. Different ideas have come up how to refine this process and are discussed. We will conclude the paper by referring to related work and a short outlook.

2 The CookIIS workflows

CookIIS is built using the empolis Information Access Suite (e:IAS). e:IAS provides a workflow model using the ”Process Manager” component, where the single elements are called pipelets. Each of these pipelets implements a specific set of actions needed for a customised CBR application. Different pipelets can be arranged in a specific order to generate a flow of operations, which is then called a pipeline. For the CookIIS application two main pipelines were created. One pipeline, the InsertCaseProvider Pipeline, is responsible for building cases from recipes and indexing them, while the other one, the Search Pipeline, is executed whenever a query is sent to the server. In the following these workflows of the CookIIS application will be described.

The InsertCaseProviderPipeline shown on the left in figure 1 consists primarily of four different pipelets. The first one, the ConnectPipelet, establishes the connection to the provided recipe base. This recipe base, a text file, contains the 888 recipes in a simple XML structure. After importing the text into the application, the text is split into the single recipes using a specific XML tag as separator. Each of the recipes is then split into the title of the dish, a list of all single ingredients including their needed amounts, and the processing instruction. This is also done using the provided XML tags. The data is stored in different attributes for each recipe, which contain the unstructured and semi-structured text. These text attributes are then committed to the TextMinerPipelet. This pipelet maps the text to concepts of our knowledge model. Most parts of the knowledge model designed for CookIIS are described in [HIB+08] and [HIBN09]. In order to do the mapping it deletes stopwords, applies a stemmer and corrects misspelling. The ingredient text list is mapped to different ingredient attributes which represent different types (concepts) of ingredients. For example, in the attribute for fruits a taxonomy of various fruits and their synonyms is modelled. The processing instructions are mapped to a model of kitchen tools needed for preparation and a modell of different kinds of preparation (e.g. frying, baking, etc.). After the mapping of the different recipes to our knowledge model the operation is then passed on to the CompletionRulesPipelet. This pipelet tries to extract additional knowledge from the recipes and tags it to them using rules to recognise the type of cuisine (e.g. Italian, Chinese, etc.) and the type of meal (e. g. starter, main dish, salad, etc.). Our approach to defining the rules for determining the type of cuisine will be explained in

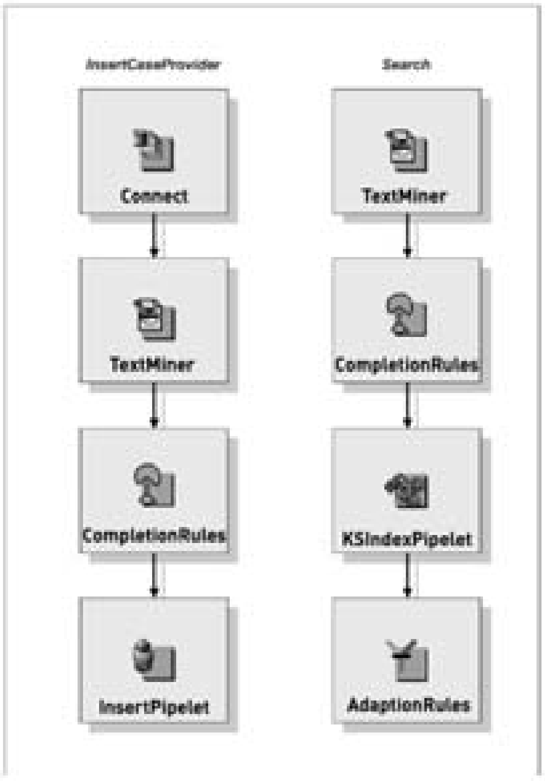

Figure 1: The workflows of CookIIS consisting of the IndexPipeline and the SearchPipeline.

chapter 3.1. The information gained by the rules is stored in additional attributes. All of these attributes are passed on to the InsertPipelet which uses them to store the attributes for each recipe in a case as attribute value pairs and to build up an efficient retrieval structure: a Case Retrieval Net [Len99].

The Search Pipeline shown in figure 1 also consists of four pipelets. A query by the user is stored in a text attribute and passed on to the TextMinerPipelet. As in the Case-Provider-Pipeline the input is mapped to the different concepts of the knowledge model. Additional filter rules identify expressions like ”do not have” and ”do not like” and store related concepts to special attributes to mark ingredients that are negated in the query. With the following CompletionRulesPipelet, rules are used again. This time they take care of query restrictions like vegetarian, nut-free, and non-alcoholic by setting filters based on the taxonomies in the knowledge model. If the user asks for a complete menu special attributes for the menu design are set here. If required, there is also a change of the weighting of single attributes applied. The expanded query is then passed on to the KSIndexPipelet which is responsible for the actual retrieval. The retrieval is based on the local similarity model provided for each attribute. The local similarity measures are based on the taxonomies and on manually set measures. Each attribute is weighted and a sum of all is computed to represent the global similarity. For each query at least five recipes are retrieved as long as there are any similar ones. After the retrieval the AdaptionRulesPiplet takes care of the adaptation process if applicable. Here an adaptation advice for the user is generated by retrieving similar ingredients to excluded and unwanted ones.

If all operations have been applied, the retrieved recipe together with the computed similarity to the query, the type of cuisine and meal, and the adaptation advice is sent back to the user interface.

3 The origin of a recipe

3.1 Determining the origin of a recipe

One requirement of the ComputerCookingContest tasks was the identification of the type of cuisine of a recipe (e.g. Italian, Chinese, Mediterranean, etc.), which was not given in the recipe itself. For this purpose we created a special attribute which includes all possible origins and some synonyms of those. All origins were ordered into a taxonomy to represent and be able to compute similarities automatically. For example the concept ”Mediterranean” is the parent for Portuguese, Spanish and Italian, while ”Mediterranean” itself is a child of the node ”Southern European”.

In order to map the recipes to one of our type of cuisine concepts we realised three approaches using rules:

- identify the origin of the recipe in the recipe title
- identify characteristic strings in the recipe title and map them to an origin
- using the occurrences of spices and herbs and other ingredients to find some that are characteristic for a type of cuisine

The first approach is based on the finding that many recipes display their origin in their title. Examples for this are the recipes for ”Chinese Cashew Chicken” or ”Swedish Cheese Pie”. Here the origin can be directly mapped to the type of cuisine.

The second approach needs some background knowledge. It is based on the fact that some foods are characteristic for a specific type of cuisine. For example ”Hot Dogs” and ”Muffins” are typical American dishes while ”Wurst” and ”Sauerkraut” are typical for the German kitchen, since they are even German words appearing in English recipes. A set of rules maps those appearances to the according type of cuisine.

The third approach is the one with the most background knowledge needed. For this approach we use the ingredients rather than the recipe title to determine where a recipe is originated. The vegetable ”Olive” is a good hint for the ”Mediterranean” kitchen while the use of ”Gorgonzola” is only common in Italy. Furthermore the tools used to prepare the dish are worth a look. A ”Wok” is usually used to prepare Asian food. For recipes that do not reveal their origin in the title or in the main ingredients, single spices and herbs or special combinations of those are often an indicator for the origin. ”Curry” is strongly used in the Indian kitchen and ”Cayenne Pepper” has its origin in Mexico.

In order to map a recipe to an origin the rules are prioritised and only applied if the type of cuisine has not already been set. The highest priority is assigned to the first approach followed by the second and the third. The prioritisation also takes into account that more specific types of cuisine are mapped before taking care of more general types. This means that a ”Pizza”, which by the rules could be mapped to Italian as well as to Mediterranean, is mapped to Italian first, so that the mapping to Mediterranean is discarded since the recipe is already mapped. Overall the CookIIS application includes about 28 rules that map the recipes with a precision of 80 to 90 percent.

3.2 Resolving conflicts

The third approach presented in the previous chapter does not take into account the possibility of ingredients hinting contradicting origins. For example, a recipe that contains the ingredients ”curry” and ”wurst” is tagged as ”Indish” (curry) or ”German” (wurst) depending on the rule that fires first. To consider these conflicts, we will count the number of ingredients that hint each origin and weight them with the amount of this ingredient that is needed to prepare the meal. To make the amounts comparabel a normalized measurement is computed. The origin with the highest score will at the end be the one that is choosen. In our example that would be ”German” because there is more ”wurst” needed than ”curry” to prepare a ”currywurst”.

4 Refining retrieval and similarity measurements

As stated before, the similarity model of the CookIIS application is made up of two parts: a local similarity measure which calculates the similarity between values of one attribute and a global similarity measure that sums up those local similarities to an overall one regarding different weights of the attributes. The local similarity measure is first of all a combination of similarity measure based on the taxonomies and one that is manually assigned in tables. Therefore the maximum value of both measures is taken as the similarity value. If a manually assigned similarity between a pair of values is higher than the one computed using the taxonomy the maximum of both measures is taken.

However, for the local similarity measure another point has to be taken into account. Since an attribute can be filled with more than one instance there is a need for a set measurement. The IAS offers three different ways of calculating this. The first one is called ”query included”. Hereby the similarity equals 1 when all query elements are contained in the attribute set. In the similarity measure ”case included” the number of those elements of the case set is determined that are also contained in the query set of the specific attribute. This means a similarity of 1 is reached when all elements of the case are included in the query. A third alternative is the intersection where the sets are more similar the more elements in both sets are contained. This results in a similarity of 1 if both sets are equal. For our application the ”query included” measure is used, because with this measurement we reach a similarity of 1 quite often and it is quite reasonable for the user why the according recipe has reached this value: simply because it contains all elements of the query.

Although having this variety of local similarity measures offered by the IAS, there is still a problem. Trying to retrieve a recipe based on a query which contains only little arguments leads to a large number of results. Often a number of retrieved recipes have a similarity of 1. For example if you ask for a recipe with chicken and cheese the first seven hits have the highest possible similarity. This is based on the fact that these recipes actually contain the two ingredients chicken and cheese. All the recipes having the same similarity are presented in the order in which they are stored in the index. This order does not represent the actual relevance in an adequate way, since it has to take context into account. One simple idea would be to look into the recipe title if one of the ingredients asked for in the query is mentioned in the title, since this would be a major hint that this ingredient is highly relevant for the recipe. Whereas in recipes where the query ingredients are not in the title, but others, they might be rather unimportant. A problem would also be a recipe with a title like ”Kung Pao” which does not give any hint on the ingredients. Another possibility is to take into account the amount of additional ingredients needed to prepare the recipe. A recipe that can be prepared with only a few more ingredients than the ones given in the query is probably more relevant than a recipe which needs a lot of additional ingredients. To compute a measure the cardinality of the attributes containing ingredients that are given in the query as well as in the case can be set into the ratio to the total number of ingredients in the case. Following the Jaccard coefficient this can mathematically be written as

$$\frac{card(ingr(query) \cap ingr(case))}{card(ingr(case))} \qquad (1)$$

whereby the denominator takes only the case into account because the ingredients of the query are supposed to be contained in the case here.

As stated in [HIB+08] the ingredients are modelled as 10 sets of different types of ingredients. Since the e:IAS does not offer a similarity measure for a defined set of sets as needed for the one above, we implement this measurement as a custom pipelet. This pipelet is placed in the Search Pipeline between the KSIndexPipelet and the AdaptionRulesPipelet and expands the workflow by one more step. The result is a similarity measure in the interval between 0 and 1 which can be used as a way to sort the recipes with the same similarity as retrieved by the KS/Index with regard to the query.

This similarity measure does not take into account the specification of the amount of ingredients in the recipe. This is due to the problem of normalising measurements (e.g. are 5 slices equal or similar to 500 grams) as well as to technical issues handling those measurements with the corresponding ingredient. One way to solve this problem would be to tag the ingredients as major and minor ingredients based on the amounts of the ingredients using additional rules. This could then be used for weighting.

5 Dietary Practises and Unwanted Ingredients

CookIIS considers several dietary practices such as *vegetarian*, *nut-free*, and *non-alcoholic*. Besides the dietary practices the user has the ability to exclude one or more ingredients explicitly. The user interface offers an extra textbox for exclusion of ingredients as well as pattern recognition in the normal "Query" text box to recognise them.

Dietary practices as well as the exclusion of ingredients can be seen as the same kind of restriction – these are *forbidden* ingredients. We found four methods to handle them if we have a recipe in our result set containing (at least one of) these forbidden ingredients:

1. leave this recipe out
2. leave only this ingredient out
3. replace this ingredient with another
4. modify the similarity model in the way that recipes do not appear in the result due to low similarity

Depending on the type of ingredient category we implemented different methods to handle these restrictions. For example if the user chooses a *nut-free* diet, all cases containing the concept *nut* or a child concept in the corresponding taxonomy are ignored (method 1). For *vegetarian* all cases containing meat or fish are excluded, because most of the time we cannot offer a good replacement. For *non-alcoholic* a filter is set to the concept *Alcohol* in the same way (both method 1).

For unwanted ingredients an adaptation determines similar ingredients for the unwanted ingredients in order to replace them in the recipes (method 2 and 3) which will be described in the next chapter.

6 Refinement of Adaptation

6.1 First Approach for Adaptation

The adaptation of the recipes is done via rules after the retrieval. It is similarity-based, because we try to exchange forbidden ingredients with similar ones. The reason for using this approach is to avoid a single adaptation rule for each ingredient with explicit replacement candidates. An alternative would be tables in which pairs of ingredients are explicitly listed.

The idea behind the rules is based on set theory. This is shown for the ingredient category fruit in figure 2. At first the intersection of the set of all forbidden ingredients from the query with all fruit ingredients in the recipe is calculated to determine which ingredients are *critical* in this recipe and have to be replaced. In a second step a relaxed *intersection* function of e:IAS is used, which works a bit different than the same function in set theory. It determines for the first set all similar concepts out of a second set over a certain threshold. Applying this function (marked bold in figure) on the critical ingredients and all (modelled) fruit concepts we get the fruits which are similar to the critical ones and can be used as replacement. If for an ingredient no similar (therefore adequate for substitution) ingredient is found, omitting them is recommended (method 2).

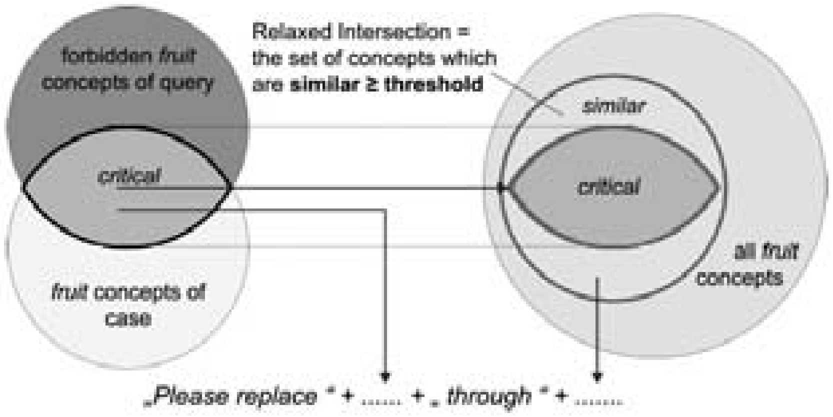
forbidden fruit concepts of query
Relaxed Intersection = the set of concepts which are similar ≥ threshold
similar
critical
critical
all fruit concepts
fruit concepts of case
„Please replace " + ... + „ through " + ...

Figure 2: Schema for the Adaption: Exchange forbidden ingredients with similar ones

6.2 Problems with the (current) Adaptation

The available *Intersection* function only considers the default similarity measure, which is mostly based on the underlying taxonomies. That means, we can not use a more specific similarity measure for the adaptation. Using our adaptation approach, replacement through parent concepts as well as child concepts are suggested, which is not appropriate under certain circumstances. For example, in order to replace the concept ”berries”, we should not propose strawberries or blueberries which are child concepts, although they have the highest similarity value (to ”berries”). We should rather propose siblings (in our fruit taxonomy) of berries like ”drupes”.

Furthermore we have to consider that if the replacement candidate is already contained in the original recipe, its amount has to be increased. This might not be not the best choice, because of decreasing diversity of taste. Let us suppose for example that we should avoid corn in a recipe which contains beans and corn. Although beans and peas are equally similar to corn, the better option would be to replace corn with peas.

6.3 Refinement of the Adaptation

According to problems with the adaptation results under certain circumstances we decided to differentiate the context of an adaptation:

1. adaptation for replacement of unavailable ingredients with similar one within the same category

2. adaptation for replacement of forbidden ingredients with similar ones from different categories according to their nutrients as used in dietary practices

3. adaptation for refinement of the taste of a recipe

To solve the problems stated above we need a more sophisticated adaptation with several sequential steps. After the two steps mentioned in section 6.1, we cut all parent and child concepts from the result set. In a fourth step we count the replacement candidates and increase/decrease the similarity threshold if they are too much/few. In a fifth step we remove ingredients from the replacement advice, which already exist in the original recipe with the aim of preserving the diversity of taste of this recipe. We implement this using a set of prioritized adaptation rules which are called sequentially and save intermediate results in extra attributes of the case. According to the fact that this adaptation consists of a sequence of steps, we call it *sequential adaptation*.

7 Related Work, Conclusions and Outlook

JULIA [Hin92] and CHEF [Ham86] are early CBR systems in the same application domain. CHEF is a planning application which builds new recipes in the domain of Szechwan cooking. To satisfy the goals of a request for a new recipe it anticipates and tries to avoid problems. Therefore it stores and retrieves occurred problems and ways of dealing with them. JULIA integrates case-based reasoning and constraints for menu design tasks. It uses a large taxonomy of concepts and problem decomposition with fixed decomposition plans. Unlike these systems our CookIIS application focuses on the retrieval and adaption of existing recipes. We try to gather additional information from them to allow a widespread retrieval. If necessary, ingredients are adapted by similar ones based on our domain model and a relaxed intersection function. The finalist teams of ”JaDaCook” [HIR+08] and ”What’s in the fridge?” [ZHND08] at 1st CCC use a similarity-based adaptation, too, but don’t distinguish between four ways of handling restrictions as we do.

After introducing the 1st Computer Cooking Contest and its motivation we described the workflow concept of e:IAS with the pipelines we used for CookIIS. At next we analysed the problems with the current similarity measures and retrieval results and explained how we refine the retrieval results by implementing a custom pipelet. Afterwards we depicted the idea behind the adaptation which is to exchange forbidden ingredients with retrieved similar ones, analysed the problems and presented the sequential adaptation we implement to overcome them.

The CookIIS application won the Menu Challenge the 1st Computer Cooking Contest and we decide to participate at the next CCC this year at the ICCBR’09 with improved retrieval and adaptation capabilities. Future improvements will be done on a more precise adaptation according to the context of adaptation. Therefore we intend to gather possible adaptation recommendations from other user in existing communities on cooking using the SEASALT architecture [BRA07]. Other ideas are to consider more dietary practices and even more recipes. In the future we will improve the usability of our system after getting feedback from all kinds of users.

References

[ABMH08] Klaus-Dieter Althoff, Ralph Bergmann, Mirjam Minor, and Alexandre Hanft, editors. *Advances in Case-Based Reasoning, 9th European Conference, ECCBR 2008, Trier, Germany, September 1-4, 2008. Proceedings,* volume 5239 of *LNCS*, Heidelberg, 2008. Springer.

[BRA07] Kerstin Bach, Meike Reichle, and Klaus-Dieter Althoff. A Domain Independent System Architecture for Sharing Experience. In Alexander Hinneburg, editor, *Proceedings of LWA 2007, Workshop Wissens- und Erfahrungsmanagement*, pages 296–303, September 2007.

[emp05] Technical White Paper e:Information Access Suite. Technical report, empolis GmbH, September 2005.

[Ham86] Kristian J. Hammond. CHEF: A Model of Case-Based Planning. *In American Association for Artificial Intelligence, AAAI-86, Philadelphia*, pages 267–271, 1986.

[HIB+08] Alexandre Hanft, Norman Ihle, Kerstin Bach, Régis Newo, and Jens Mänz. Realising a CBR-based approach for Computer Cooking Contest with e:IAS. In Schaaf [Sch08], pages 249–258.

[HIBN09] Alexandre Hanft, Norman Ihle, Kerstin Bach, and Régis Newo. CookIIS – Competing in the First Computer Cooking Contest. *Künstliche Intelligenz,* 23(1):30–33, 2009.

[Hin92] Thomas R. Hinrichs. *Problem solving in open worlds*. Lawrence Erlbaum Associates, 1992.

[HIR+08] P. Javier Herrera, Pablo Iglesias, David Romero, Ignacio Rubio, and Belén Díaz-Agudo. JaDaCook: Java Application Developed and Cooked Over Ontological Knowledge. In Schaaf [Sch08], pages 209–218.

[Len99] Mario Lenz. *Case Retrieval Nets as a Model for Building Flexible Information Systems.* Dissertation, Mathematisch-Naturwissenschaftliche Fakult¨at II der Humboldt-Universität zu Berlin, Humboldt University, Berlin, 1999.

[Sch08] Martin Schaaf, editor. *ECCBR 2008, The 9th European Conference on Case-Based Reasoning, Trier, Germany, September 1-4, 2008, Workshop Proceedings,* 2008.

[SMT08] Armin Stahl, Mirjam Minor, and Ralph Traph¨oner. Preface:Computer Cooking Contest. In Schaaf [Sch08], pages 195–198.

[ZHND08] Qian Zhang, Rong Hu, Brian Mac Namee, and Sarah Jane Delany. Back to the Future: Knowledge Light Case Base Cookery. In Schaaf [Sch08], pages 239–248.