

Realising a CBR-based approach for Computer Cooking Contest with e:IAS

Alexandre Hanft, Norman Ihle, Kerstin Bach, Regis Newo, and Jens Mänz

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

Abstract. This paper describes a web-based CBR application competing at the first Computer Cooking Contest at the ECCBR 2008. It describes the technical realisation of our system, called CCCIS, and its underlying concepts, knowledge models, adaptation and completion rules as well as the web interface. CCCIS provides recipes after receiving a natural language queries from users while considering specifications on a certain diet, available ingredients and other preferences. Further on, this paper will deliver an insight into the technical background of the implementation and how CCCIS can be accessed and used.

1 Introduction

The application domain of knowledge-based systems can differ in a broad way and they cover various topics like help desk applications, diagnose systems in different domains or product recommendation in e-commerce. But beside the commercial applications, Case-Based Reasoning might also be a technique that makes life easier, for example by giving an advice for a delicious meal using the ingredients that are available, considering food one likes or cannot stand and further on giving advices how to prepare those meals or adapt from existing ones.

In this paper we present the implementation of a knowledge-based system that provides about 900 recipes that can be accessed using the natural language for requesting meals. Regarding the competition rules we will show how CBR can be used to find similar meals and adapt solutions. We will present how we have modelled the cooking domain and in which way our system, CCCIS, deals with different kinds of requests. We will depict the realisation of our system in detail. First of all we will describe the requirements of the cooking domain, followed by a short introduction in the e:Information Access Suite (e:IAS), the tool we have used to implement CCCIS (speak "cis"). In section 4 we describe several aspects of our knowledge model and in which way this affects the computation of similarities during the retrieval process before we introduce the functionalities of our Web GUI in section 5. This technical description will close up with a conclusion of our work and an outlook on how we can improve our application to make life easier.

2 Requirements of the Cooking Domain

The delivered 888 recipes have a relatively simple XML structure for each **RECIPE** consisting of elements **TI**, **ING**, **PR** with more or less structured text. For the *Compulsory Task* the following requirements should be accomplished and therefore extracted from these recipes:

1. ingredients, restricted to those occurring in the recipes in the database
2. type of ingredients, such as meat, fish, fowl, vegetables, fruits, nuts, alcohol
3. type of meal, such as starter, salad, soup, ice-cream, cake, sauce, main course, dessert, sweets, three-course menu
4. dietary practices, restricted to the following: *vegetarian*, *nut-free*, *non-alcoholic*.
5. type of cuisine, such as Italian, Chinese, Mediterranean,

For the *Negation Challenge* unwanted ingredients should be recognised, covered in phrases like "i do not like ~". The *Menu Challenge* requires the composition of complete menus with certain appropriate dishes.

According to the requirement that specialisations of a concept should be recognised too, taxonomies for all major ingredients should be built. For instance if a user want (no) poultry, chicken can(not) be offered.

3 The empolis Information Access Suite

The empolis Information Access Suite (e:IAS) [1] is being developed by empolis¹, a Bertelsmann company and consists of a server-client based knowledge provision component, the Knowledge Server, and a knowledge modelling tool, the Creator.

The Creator is based on the free eclipse integrated development environment² adopting its workbench concepts and extending it with several components of its own. The tools within the Creator, so called Managers, are used to create a model of a respective domain (ModelManager), configure the import of existing data into the model (DataManager) and define the workflow of their processing (PipeManager).

The Knowledge Server queries the system's case base(s) as well as it is building the index each retrieval is carried out on. It also uses pre-defined concepts to generate an index of imported data from different kinds of sources that can be queried using Java, JSP or web services.

e:IAS uses case-based reasoning (CBR) as its main underlying methodology and when modelling a new knowledge domain the domain is represented by an aggregate that includes freely definable attributes of different types. Further on attributes can be defined as a set of instances of a particular data type, or as a compound type, including atomic data types. Data types can be constrained or specialised for instance by defining ranges or a fixed set of possible values.

In the case of text based data types these values are called concepts. When defining new data types it is also necessary to define their similarity measures,

¹ <http://www.empolis.com>

² <http://www.eclipse.org/>

which are crucial for case-based reasoning. These similarity measures can be defined as simple mathematical functions, in the case of numerical data types, or, in case of text based data types, as simple string distance measures. In case of concepts more elaborate similarity measures are also possible. Concepts can be arranged in lists or orders, forming taxonomies or similarity matrices. More complex similarities can be modelled using combined similarity measures where different measures are defined and the highest similarity is taken into account.

Once a model has been created it can be filled with data from a multitude of sources such as simple text or xml files, databases, data collected by a web crawler or provided by an MS Exchange or Tamino server. The import workflow can be pre-defined using given components to build an index representing the input sources.

Additional tools such as the TextMiner and different rule pipelets (components of the PipeManager) can be used for processing the data at import time as well as during the retrieval process. Rules can be used to enhance the index with concepts, extend queries in order to improve the retrieval result or extend the case base to learn new cases as it is described in [2].

The processing of data can be defined using the Pipe Manager to describe the indexing process as well as the querying process. The Pipe Manager provides a tool kit to realise different CBR-approaches like conversational CBR or CBR based on decision trees. Each process consists of so called Pipelets that are able to convert, analyse or process data.

e:IAS also meets most of the requirements described in section 2 that arise from the knowledge domain. It offers the possibility to define custom similarity measures for data types and combine any number of them into a combined similarity measure, which is implemented in such a way that all included measures, are consulted and the highest value is taken into account. Switching between individual similarity measures during the system's runtime is realised using adaptation rules that are a part of e:IAS.

4 Knowledge Models

The Case Representation is based on structured CBR. Hence a case is build as an Aggregate of Attributes of formerly user-defined or predefined (in e:IAS) types. According to the requirement that the type of ingredients should be recognized, we build several types for ingredients (Type is `Txt_...`):

- Basics
- Meat
- Fish
- Vegetable
- Liquids (inclusive alcoholic drinks)
- Milk(products)
- SpiceAndHerbs
- MinorIngredients
- Supplements

Through the Analysis Mapping the attribute with these types are filled with the recognised concepts. For all types except **Basics** and **MinorIngredient**, taxonomies are build. To reflect the type of meal **Specie** as well as **Tool** for the preparation are modelled in the same way. The concepts and taxonomies for these attributes are taken from other e:IAS demo applications, Wikipedia or from everyday knowledge.

In this chapter we show some examples how we modelled different kinds of data types and their according similarities.

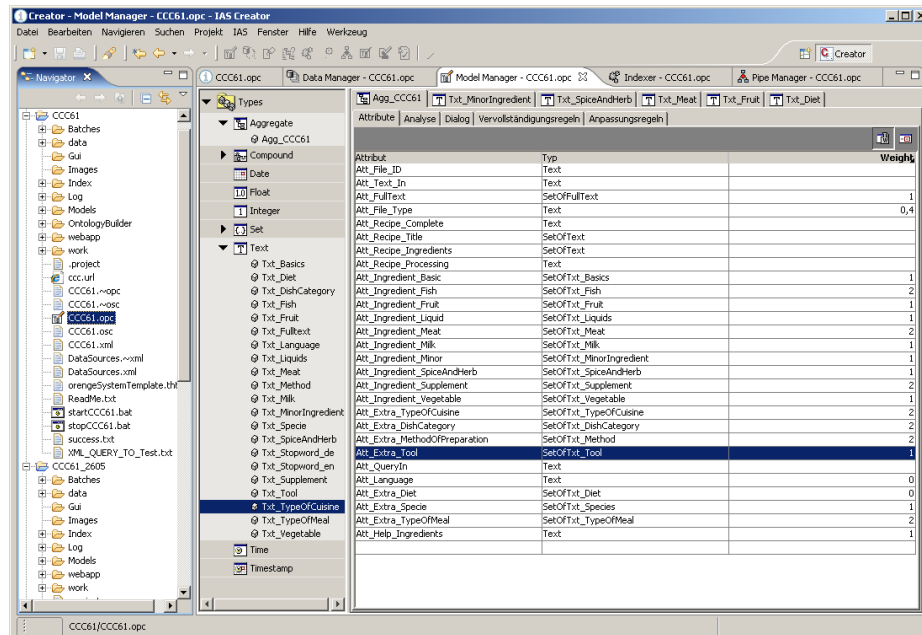


Fig. 1. e:IAS Creator with the Aggregate Agg_CCC61 used in this project

4.1 Aggregate

Each case (and query) is modelled as an Aggregate **Agg_CCC61**. Most attributes types based on Text. For each of these text types all allowed concepts are indicated, inclusive their representing terms for each considered language. These are German and English. See as an example the concept kiwi figure 2 where the terms "chinese gooseberry" and "kiwi" indicate this concept. If a case or query could have more than one value for an attribute, then a Set of this type is defined and used in the aggregate instead of the type itself. A weight is assigned to each attribute to reflect the importance of this attribute for the case. Figure 1 shows the GUI Creator with the Aggregate used in this project. The attributes

reflect parts of the Query (for example `Att_QueryIn`), original parts of a case (for example `Att_Recipe_Title` or `Att_Recipe_Ingredients`), or concepts and features found in the case/query, for example `Att_Extra_TypeOfMeal`. The last one are filled via an Analysis mapping or *completion rules*.

4.2 Attribute `Txt_Fruit`

This section describes `Txt_Fruit` and nuts modelling. The data type `Txt_Fruit` contains a list of all possible fruits. Figure 2 shows the definition of the concept *kiwi*. On the left hand side of the figure the concepts are displayed and for each concept we are able to define synonyms ensuring an appropriate mapping. All concepts are ordered in a taxonomy, a part of this can be seen in figure 3.

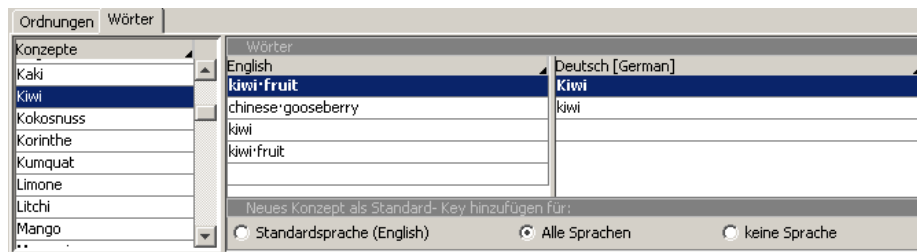


Fig. 2. Concept *Kiwi* of Type `Txt_Fruit` with different terms for each language

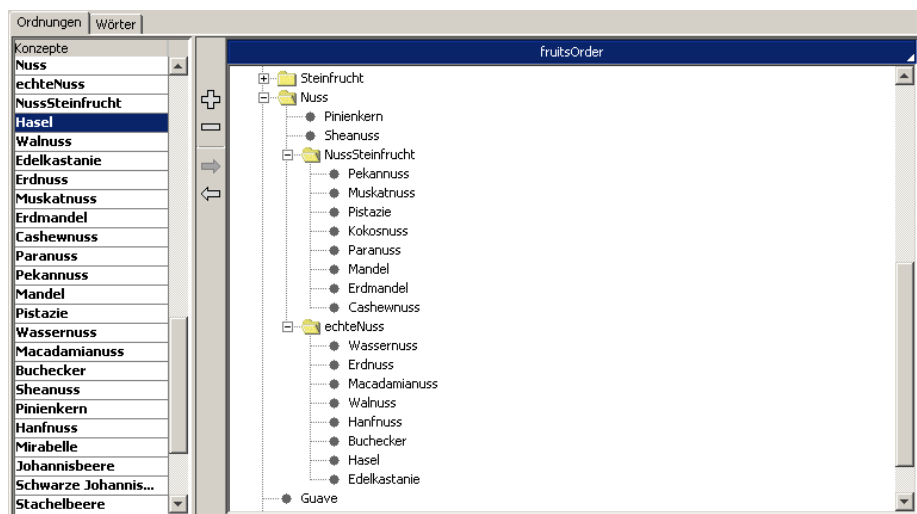


Fig. 3. Taxonomy of Nuts as part of Modelling of Fruits in Creator

4.3 Attribute `Txt_Meat` and `Txt_Fish`

For `Txt_Meat` and `Txt_Fish` modelling concepts are modelled in taxonomies. Both are necessary for the decision if a dish vegetarian or not. Additionally to the modelled concepts we need an extra rule to exclude "chopped" as ingredient, because the Textminer recognises the often used "chopped" as concept *chop*, which is not wished.

4.4 Attribute `Txt_SpiceAndHerb` and `Txt_TypeOfCuisine`

This section describes `Txt_SpiceAndHerb` modelling and the assignment to type of cuisine. All occurred concepts spices and herbs are listed and ordered in two taxonomies `Spices` and `Herbs`. To each element of `Txt_TypeOfCuisine` a bunch of spices or herbs are assigned to recognise the type of cuisine depending the spices and herbs a recipe contains.

4.5 Attribute `Att_Extra_TypeOfMeal`

The attribute `Att_Extra_TypeOfMeal` describes the type of any given recipe (or case). That is, the content of this attribute indicates whether a recipe is e.g. a starter or main course. We used the types that are given on the competition's website. The attribute is represented as a set, because most of the time, a meal can have different types. For example, an ice-cream can always be seen as a dessert. For the modelling of this attribute, we first had to extend our general model by adding a new class `Txt_TypeOfMeal`. That class contains an enumeration of all possible types of meal. Second, we added the class `SetOfTxt_TypeOfMeal`, which represents a set of types of meal (i.e. a set of `Txt_TypeOfMeal`). We then just had to insert the attribute `Att_Extra_TypeOfMeal` in the aggregate as an object of the class `SetOfTxt_TypeOfMeal`.

In the next step, we implement some rules which are used for the classification of each recipe. Those rules are mostly based on two features:

- some keywords in the title (e.g. "sherbet" in the title indicates that the meal is an ice-cream)
- some indicative ingredients (e.g. "salad dressing" is only used for salads)

The rules are implemented as *completion rules*, i.e. they are used to complete or extend the cases with the content of the attribute `Att_Extra_TypeOfMeal`.

4.6 Similarity modelling

For nearly all attributes we build a taxonomy to reflect the generalisation/specialisation relationship and to assign a similarity. Therefore we use a *TaxonomyPath similarity measure* based on these taxonomies, which assign each specialisation the value 0.9 and the value 0.3 for generalisation, so we don't have to assign a similarity value between each pair of concepts. Because in e:IAS Taxonomy-Path all edges have (for same direction) same value, it might be useful to define

different values for certain pairs. Therefore *Table similarity measure* could be used, where a dedicated similarity value can be assigned to each pair of concepts. *Combined similarity measures* are subsequently used to combine values of different measures. We decide to "mark" higher values, hence our *Combined similarity measures* `Sim.Fruit.Comb` take the maximum value of both aforementioned measures.

4.7 Dietary practises

To accomplish the dietary practise *nut-free* all nuts are modelled as part of `Txt_Fruit`, see figure 3. The input from the user is mapped as concept *vegetarian*, *nut-free*, and *non-alcoholic* into the attribute `Att_Extra_Diet`. For each of these dietary practises an exclusive filter on the Taxonomy of the according attribute is set in order to omit each case in the retrieval result containing these concepts or child concepts of them. For example if the user choose *nut-free*, all cases containing the concept *Nut(Nuss)* or child concept in the Taxonomy *fruitsOrder* are ignored. This Rule is given in the syntax used in the Creator:

```
IF HasElement(Att_Extra_Diet; "Txt_Diet.V1:nut-free")
THEN SetHasElementTaxonomyFilter(Att_Ingredient_Fruit;
    "fruitsOrder"; "Txt_Fruit.V1:Nuss"; none; override);
```

Additionally if the user writes "nut-free" or "no nuts" in the query, the attribute `Att_Extra_Diet` is set too and the same rule above fires. For *vegetarian* all cases with any concept in `Txt_Meat` are excluded. For *non-alcoholic* a Filter is set to the concept *Alcohol* in *Tax.Liquid* in the same way.

4.8 Recognition of Negation Phrases

To recognise if the user do not want certain ingredient, we implement a rule for each type for ingredient. The forbidden ingredients are collected in extra attributes. After the first step in the retrieval a second retrieval queries similar ingredients for the unwanted ingredients in order to replace them in the recipes. Recipes containing an ingredient with no similar (that means adequate for substitution) ingredient, are omitted. An example for such a rule is shown here:

```
IF Pattern(Att_QueryIn; "hate~[Txt_Fruit.V1=INGR]"; all)
THEN SetAttribute(Att_Extra_Forbidden_Ingredient_Fruit;
    Union(Att_Extra_Forbidden_Ingredient_Fruit; $INGR;
    SetOfTxt_Fruit.V1); none; override);
    RemoveConcept(Att_Ingredient_Fruit; Att_QueryIn; $INGR)
```

5 Web GUI

The usage of the web interface of CCIIS ³ is explained as follows. The recognition of the users wishes happens text-based, that means the user can write arbitrary

³ available at <http://c135-2.iis.uni-hildesheim.de:8080/ccc/>

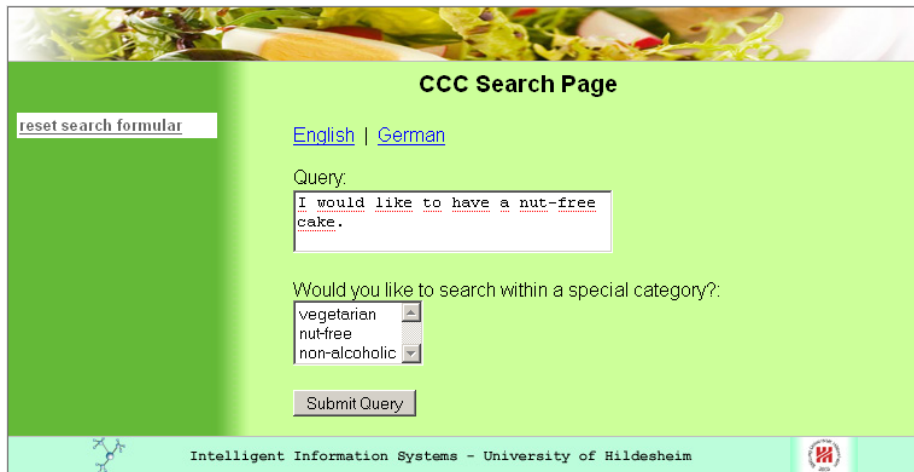


Fig. 4. Web GUI showing a search for a nut-free cake

text in an input text box, in natural language. Only for the three dietary practices s/he can choose them from a list box, see figure 4. After submitting a Query, the user gets a list of the top five recipes. Modification of recipes is not implemented at the moment yet, hence the user see the original recipes with their title, ingredients and preparation together with the assigned similarity. One of them is in 5. The next five recipes can be obtained through clicking "forward" Link on the bottom of this page. The search form can be reset using "reset search formular" at the upper left. This figure shows the search on the web application.

This query returns 5 of 81 Result, one of the best recipe is shown in figure 5.

5.1 JSP and TagLib Library

Integrated into the source code of the graphical user interface are some JSP Tags that control the access from the client to the server of the e:Information Access Suite. JSP-Taglibs are mechanisms which allow HTML tags to be associated with Java classes. The JavaServer Pages Tag Library (JSTL) ⁴ encapsulates as simple tags the core functionality common to many Web applications. Additionally to the Standard TagLibrary empolis provides a library with Tags for accessing the functions of the e:IAS server. This library is configured for general options by the two files "api.properties" and "ias.properties". While the api.properties file contain specific values for the language, the request aggregate and format of the date, the ias.properties specifies JNDI options. In the source code a request object is created with the "createRequest"-Tag, before common parameters as the current language, the cursor position and the size of the result is set with the "setGeneralParameters"-Tag if the user input is not empty. This user input is

⁴ <http://java.sun.com/products/jsp/jstl/>

Results for the query:

"I would like to have a nut-free cake."

Total Number of Hits: 81

1. Apfelquarkkuchen (Apple And Cream Kuchen)

Similarity: 75%

Ingredients:

1 Tblsp Lemon Juice
1 ea Egg; Large
1 pk Yeast; Dry, Active
1 tsp Cinnamon
1/2 c Milk
1/2 tsp Salt
1/4 c Butter or Margarine
2 Tblsp Flour; Unbleached
2 c Flour; Unbleached, Unsifted
3 c Apples; Tart, Sliced
3/4 c Sugar
4 Tblsp Sugar
8 oz Cream Cheese; Softened
CAKE:
FILLING:

Processing: CAKE: Mix yeast, salt, sugar and 3/4 cup flour. Add butter to milk. Heat until very warm (120 - 130 degrees F.). Gradually add milk to flour mixture. Beat for 2 minutes. Add egg and 1/2 cup flour. Beat with an electric mixer on high speed for 2 minutes. Mix in enough flour to form a soft dough. Knead for 5 to 10 minutes, until dough is shiny and elastic. Place in greased bowl and let rise for 1 hour or until doubled in bulk. Pat dough into wellgreased 10 inch springform pan pressing the dough 1 1/2 inches up the sides of the pan. FILLING: Toss apples with lemon juice, cinnamon, 1/4 cup sugar, and flour. Arrange in rows on top of the dough. Beat together cream cheese, 1/2 cup sugar and egg. Spread over apples. Let rise in warm place for 1 hour. Bake at 350 degrees F. for 30 minutes. Best when served warm.

Category: cake, dessert

Fig. 5. Web GUI showing a first search result for a nut-free cake

than added to the request object by directly setting the values for the attributes `Att_Query_In` and `Att_Extra_Diet` to the values from the input form using the `"setAttributeValue"`-Tag. When the request object is completed the search is executed, using the `"executeRequest"`-Tag. Now the search is carried out on the server side by sending the request to the Knowledge Server where the pre-defined retrieval steps are executed. The results of the request are retrieved from the server using the `"getResult"`-Tag which receives the results directly from the `KSIndexPipelet` that is one of the Pipe Manager components provided by the `e:IAS`. The results are stored in a variable and presented to the user by iterating over the result set of recipes. These recipes are grouped by ten (reduced to 5 according to competition rules) and sorted by similarity. By navigating forward the next ten/five recipes can be accessed.

6 Conclusion and Outlook

In this paper we presented the realisation of the web-based computer cooking contest application `CCCIIS` including the basic ideas behind our knowledge model and how the application can be used. We explained which knowledge models we have created and on which kinds of similarity measures of computation is executed. Also, we have explained how our rule set (completion and adaption rules) work and how they effect the result. Further on we gave a brief overview about the technical background of the `e:IAS` before described the functionalities of our implementation.

The knowledge models described in section 4 are already quite strong and in combination with the rule set the system's results are promising. Although we can still improve our knowledge models according to evaluate how users query `CCCIIS` and what kinds of results are returned.

In the future we can possibly improve the usability of our system after getting feedback of all kinds of users. Although, the strength of a CBR system, the knowledge model could be used to create not only meals, maybe it can be extended to create menus for a whole week regarding all kinds of constraints.

Currently our application runs as a stand-alone application, but the domain can also fit as a multi-agent system (MAS) in which each agent represents one dinner participant and the MAS is supposed to create a meal that everybody likes and the host can cook. In this kind of application the collaboration of different instances has to be developed.

References

1. empolis GmbH: Technical white paper e:information access suite. Technical report (September 2005)
2. Aamodt, A., Plaza, E.: Case-based reasoning : Foundational issues, methodological variations, and system approaches. *AI Communications* **1**(7) (March 1994)