Cooking Cake

Conference Paper · January 2009

CITATION
1

READS
50

5 authors, including:

Ralph Bergmann
Universität Trier
260 PUBLICATIONS **3,382** CITATIONS

SEE PROFILE

COOKING CAKE

Christian Fuchs, Christoph Gimmler, Simon Günther, Lukas Holthof,
Ralph Bergmann

University of Trier, Germany

{ fuch4102@uni-trier.de, gimm4101@uni-trier.de, guen4101@uni-trier.de,
holt4102@uni-trier.de, bergmann@uni-trier.de }

Abstract. In this paper we present "Cooking Cake": a CBR system for ontology-based retrieval and adaptation of recipes. "Cooking Cake" was developed in order to participate in the 2nd Computer Cooking Contest which will be held at the International Conference on Case-Based Reasoning (ICCBR '09) at the University of Washington (USA).

Keywords: Case-Based Reasoning (CBR), retrieval, similarity, ontology, Collaborative Agent-Based Knowledge Engine (CAKE)

1 Introduction

In this paper we present "Cooking Cake", a CBR system based on CAKE (Collaborative Agent-based Knowledge Engine) [1,5,7]. The CAKE system combines workflow, agent and CBR technologies following an ontology approach. "Cooking Cake" is mainly built on CAKE's CBR technology which provides efficient retrieval facilities including sophisticated similarity calculations. CAKE itself was developed at the University of Trier, Germany.

"Cooking Cake" was developed within a two-semester student project in order to participate in the 2nd Computer Cooking Contest at the International Conference on Case-Based Reasoning (ICCBR '09), held at the University at Washington, WA (USA). The Computer Cooking Contest is made up of three different challenges:

- **Compulsory Task**, which involves answering queries that require the selection and, where appropriate, modification of recipe for a single dish.
- **Adaptation Challenge**, which requires an adaptation of the list of ingredients and instructions for preparation of the dish.
- **Menu Challenge**, which requires the composition of a three-course menu based on the recipes provided by the CCC-Jury.

With "Cooking Cake" we mainly want to compete in the Compulsory Task and the Menu Challenge, but all functions needed to take part in the Adaptation Challenge will be implemented as well.

In this paper we want to give a quick overview of the important conceptual aspects of our software. First we will get into the design of our domain-ontology (Section 2). After this we will describe all steps needed to transform the recipe database - given by the Computer Cooking Contest-Jury - to our internal case base (Section 3). This will not only include the pre-processing methods used, but also we give insight on CAKE specific representation of data, the classification process

needed to evaluate a single recipe's "Type of Meal" or "Type of Cuisine" and how the system deals with dietary-practices and explicit seasonal availability of certain ingredients.

In the last part of this paper, we want to give a more detailed view of the similarity and retrieval functions that are provided by CAKE (Section 4). Finally, we give a quick introduction to our interface (Section 5), evaluate the results of given exercise queries (Section 6), draw some conclusions and describe some lines of future work (Section 7).

2 Ontology-Design

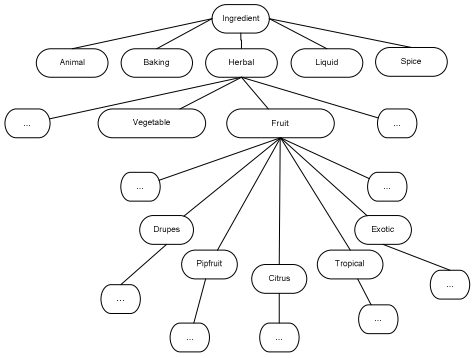
An ontology is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base [2]. As the ontology provides nearly all knowledge needed to solve given problems, we put a lot emphasis on reviewing existing resources for gaining information about the domain "cooking". Unfortunately, most available resources – like the available applications of the CCC '08 – were not appropriate for our needs as they followed a different structural approach in the design of the cooking ontology. Therefore we decided to construct a suitable ontology on our own [8] which is represented in a self developed XML-Dialect.

After analyzing the given recipe database we distinguished three different types of ontology-concepts:

- **Category-Nodes**, groups of ingredients (e.g. fish or pork)
- **Ingredient-Nodes**, ingredients (e.g. squash or pumpkin)
- **Synonym-Nodes**, ingredients which are basically identical to another ingredient, but differ in their spelling (American or British English, Typos)

With the support of a local head chef we built a basic ontology with main focus on a skeletal structure of category nodes (see Fig.1). This ontology was consistently expanded and ended up consisting around 140 different category-nodes. It has a hierarchical structure with "ingredient" as the root-node and five main ingredient-origin classes as direct sub-nodes:

- **Animal**, ingredients of animal-origin. Sub-nodes e.g. fish or cheese
- **Herbal**, ingredients of herbal-origin. Sub-nodes e.g. vegetable or corn
- **Baking**, baking related ingredients. Sub-nodes e.g. yeast or gelatin
- **Liquid**, liquid ingredients. Sub-nodes e.g. alcoholic liquors or soft drinks
- **Spices**




```
<node value="ingredient">
   <node value="animal">…</node>
   <node value="baking">…</node>
   <node value="herbal">
     <node value="vegetable">…</node>
     <node value="fruit">…</node>
     …
     </node>
   <node value="liquid">…</node>
   <node value="spice">…</node>
</node>
```

Fig.1. Part of the basic ontology and its representation in XML

3 From Recipe Database To Case Base

The given recipe database is a semi-structured XML document where each recipe consists of three parts (see Fig.2 as an example):

- Title (TI)
- Single/multiple ingredients (IN)
- Preparation (PR)

In order to develop a system which can address all tasks of the Computer Cooking Contest the given recipe database has to be transformed to a fully structured and formalized document.

```
<RECIPE>
      <TI>"Lutheran" Hotdish</TI>
      […]
      <IN>1/2 lb Mild or spicy sausage</IN>
      <IN>1 lg Onion (sliced and quartered) (up to) </IN>
      <IN>1 lb Uncooked pasta (i.e. elbow; twisted; wagon
      wheels, shells; etc) (up to)</IN>
      […]
      <PR> […] Meanwhile, prepare the pasta per pkg
      instructions.    In a large pan, combine all
      ingredients.  Add enough tomato sause until mixture
      is well coated, […] </PR>
</RECIPE>
```

Fig 2. Example Recipe

3.1 Pre-Processing and Ingredient Extraction

In order to provide an accurate retrieval and similarity measurement every <IN>gredient tag needs to be reduced to a basic ingredient. This process was subdivided into four steps:

- **Filter Stopwords**
  Remove all words with no direct relation to an ingredient (e.g. “in”, “the”, “or”).
- **Filter Cooking Units**
  Remove all cooking units (e.g. “1/2 lb”, “1 pkg”, “1 tablespoon”).
- **Filter state of ingredient**
  Remove all ingredient states (e.g. “filtered”, “chopped”, “finely sliced”).
- **Simple Form Reduction**
  Reduce words to their simple form by using the lexicographic rules of Kuhlen [3] (e.g. “Onions” to “Onion”).

“`1 lg Onion (sliced and quartered) (up to)`” (taken from Fig. 2) will be transformed to “`Onion`”.

3.2 Ontology Refinement and Case Generation

To perform an efficient search the constructed basic ontology (See Section 2) had to be extended and enhanced by inserting simple form ingredients which are extracted out of the recipe database. In addition, a link between the automatically extracted ingredient in the recipe and its place in our ontology had to be stored. For this reason we developed a java based application (See Fig. 3 and Fig. 4) which:

1. Performs all the filter methods in Section 3.1.
2. Reduces an ingredient to its simple form.
3. Automatically skips an ingredient which already has been inserted.
4. Creates a link between recipe-ingredient and ontology-ingredient.
5. Supports the manual insertion of a simplified ingredient via GUI into its place in our ontology.
6. Applies “Seasonal Restrictions” and “Dietary Practices” (see Section 3.4).
7. Classifies recipes “Type of Meal” and “Type of Cuisine” (see Section 3.3).
8. Generates CAKE Files: CaseBase, Similarity DataBase and DataModel.

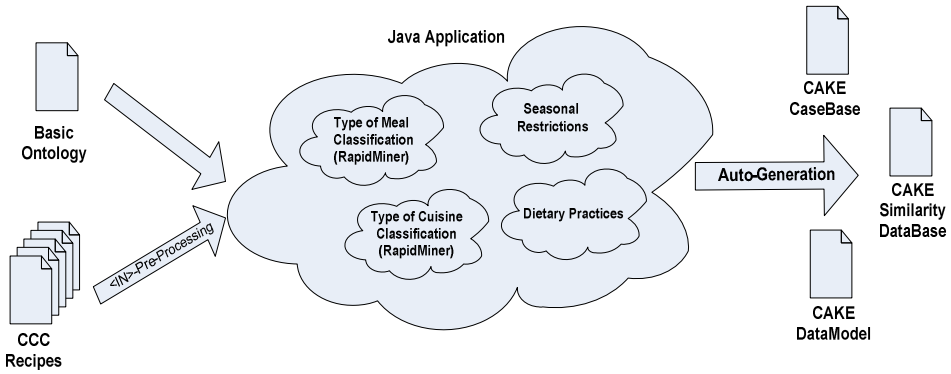


Fig. 3. Process Visualization

The CAKE DataModel consists of an enumeration of all category-nodes of the refined ontology, followed by a taxonomy order of these nodes. The CAKE CaseBase includes all recipes where the ingredients are represented by their simple form values stored in the ontology. This representation is extended by additional information, e.g.

information about “Seasonal Restrictions” and “Dietary Practices” as well as “Type of Meal” and “Type of Cuisine” classification. Finally, the CAKE Similarity DataBase is generated. In this DataBase the similarities between two different ingredients is stored (See Section 4).

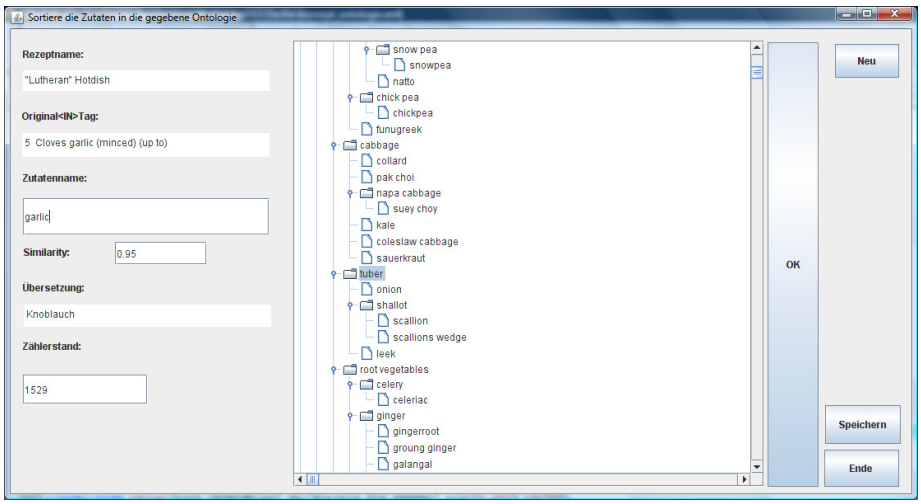


Fig. 4. Developed Application

3.3 Type of Meal / Type of Cuisine

As almost every exercise query includes information about “type of meal/type of cuisine” and “seasonal/dietary practices” the classification of those types of meals or ingredients is a main requirement for all challenges.

Our approach to classify a meal is based on RapidMiner 4.3 (community edition) and it`s implemented classifiers. RapidMiner is a java based open source data mining solution, which addresses a wide range of data mining tasks. It has implemented a bunch of learners, IO-methods and visualization functions [4].

We tested several classifiers supported by RapidMiner and came to the conclusion that the Naive Bayes classification provides the best results for our task. The Naive Bayes algorithm makes the assumptions that all relevant attributes have the same significance and are fully independent of each other. Although these assumptions can’t be fulfilled in our case, it delivers suitable and efficient results for the “type of meal” and “type of cuisine”-classification.

The selection of attributes and recipes is an important step to predict the confidence of the class. The Naive Bayes learner is based on previous calculation of probability values that reflect a link between attribute values and class membership. It determines a confidence for each class and returns the highest value as prediction for each recipe.

First, we had to manually select attributes for the classification. Second, we had to manually pre-classify a set of recipes. We ended up selecting about 70 relevant ingredients for “type of meal” and 80 relevant ingredients for “type of cuisine” which then were the basis of our classification process. Also we had to pre-classify about

130 recipes for “type of meal” and about 90 for “type of cuisine”, e.g. the recipe “Char Siu Gin Doi” which has a typical combination of ingredients used in the asian cuisine like “Bamboo shoots”, “Soy sauce” and “Chinese noodles”. Based on these pre-classifications RapidMiner built a classification model which then could be used to classify all other recipes of our database.

For the purpose of estimating and improving the accuracy of our classification we decided to implement the m-fold cross validation method in RapidMiner. This validation determines the classification accuracy by dividing the pre-classified dataset into m disjunctive subsets, learning a classification concept with m-1 of these subsets and applying the learned concept to the skipped subset. By comparing the pre-classified class of a recipe with the predicted class, the classification error can easily be identified. In our case we used 10 randomized subsets of pre-classified recipes.

Selected Ingredients	**Accuracy**
General Vegetable, Fruit and Meat Ingredients	51%
Sophisticated Herbal and Animal Ingredients	59%
Refined Selection of Herbal, Animal and Spice Ingredients	66%

Fig. 5. Process “Type of Meal”

Although the accuracy percentages in Fig. 5 and Fig. 6 may seem low, most errors are false positives, which derive from intersections between different recipe classes as Naïve Bayes requires fully independent classes. An example for a false positive would be the classification of a pizza recipe as “main course”, which is not a fault at all, but the best classification would be “pizza”. Most errors we are facing are of this kind.

Selected Ingredients	**Accuracy**
All Spice Ingredients	46%
Spices and some Vegetables Ingredients	71%
Spice, Vegetables and Meat Ingredients	91%

Fig. 6. Process “Type of Cuisine”

We were able to improve the accuracy and the selection of relevant attributes (ingredients) by evaluating and analyzing results we achieved with several ontology-subsets. As seen is Fig. 5 and Fig. 6 we were able to increase the accuracy from 51% to 66% for “Type of Meal” and from 46% to 91% for “Type of Cuisine” by refining the ontology as well as the selection of relevant attributes for the classification.
It was not possible to achieve a perfect classification of all recipes concerning “type of meal” or “type cuisine”. Finally we had to correct some fatal error classifications manually.

3.4 Seasonal Restrictions / Dietary Practices

Not only “Type of Meal” and “Type of Cuisine” are possible requirements in a query. “Seasonal Restrictions” and “Dietary Practices” can be desired by a user as well.

“Seasonal Restrictions” occur for 23 different ingredients and can range in their value for each month between “fresh/is in season” and “storable”. As an example, “Cabbage” can be stored from January to March, while it is “in season” from September to December.

“Dietary Practices” exist for “Gout Diet” and “Cholesterol Diet”. In both cases preferable and restricted ingredients can be found in a chart given by the CCC-Jury. Following a “Cholesterol Diet” the guidelines – which are no medical information – recommend to prefer low-fat milk products and avoid butter, eggs, turkey, or pork.

In order to offer best possible retrieval results we decided to treat all seasonal information and dietary recommendations as extended ingredients. In the current version of “Cooking Cake” we are able to define similarities between each month and even closer between “storable”-months and “fresh”-months. Dietary information for a single recipe is right now only available as a Boolean expression, which means a recipe either is “Gout Diet” or “Cholesterol Diet”-valid or it is not.

4 Retrieval & Similarity

Considering the ontology as the main knowledge base of our software, the weight initialization of every node was a key process in the development of “Cooking Cake”. We finally decided to calculate the weight by the formula

$$\mathbf{weight(node)} = \left(\frac{\mathbf{height(node)}}{\mathbf{maxlevel}}\right) \quad (\mathbf{1})$$

where “maxlevel” stands for the depth of the path the node is on. In some evident cases manual enhancement of these weights were necessary.

Choosing CAKE as framework for developing “Cooking Cake“ allowed us to use the CAKE Data Model and the already implemented CBR technology. The CAKE Data Model describes all kinds of data that can occur in the system. It is an object-oriented model using specialization and aggregation to define the data classes. Available data classes are atomic classes like boolean, integer, double, date or time as well as compound data classes like aggregates, collections and intervals. These system classes are used to define a cooking specific data model.

Each data class of this data model can be instantiated as a CAKE data object. The main function of the CBR technology is the similarity based retrieval of above described data objects. For this purpose a similarity model is defined on the top of the data model combining and configuring similarity functions predefined in a similarity library [5].

The CAKE's library provides about 30 similarity measures. Similarities in the provided version of “Cooking Cake” are calculated by determining the first common father of query and case object. This solution is meant to be an initial solution as we are working on the development of an independent calculation method in order to get the best possible and differentiated results for the cooking-domain.
Right now the retrieved similarity between a case object and a query object is calculated as

$$\sum_{i=0}^{\#AggAttribute} AggAttributeWeight(i) * weight(CommonFather(Query, Case)) \quad (2)$$

where aggregate (Agg) contains for example the five main categories (see Section 2) like “animal” or “herbal”. For each of these aggregate-attributes a weight is defined which represents the importance of this attribute for the retrieval. This weight is multiplied by the weight of the common father-node of the query and the case object.
The possibility to avoid ingredients is implemented as an initial solution as a filter on the retrieved cases, but will be reworked and re-implemented in the similarity measurement.

5 “Cooking Cake” / Graphical User Interface

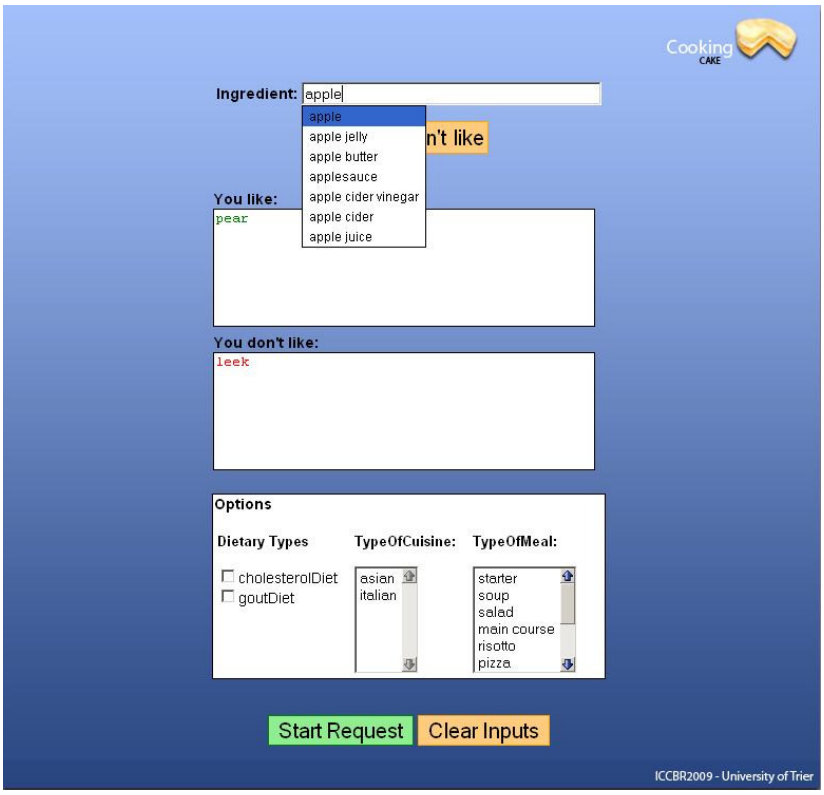


Fig. 7. Interface

The current version of “Cooking Cake“ comes as a MySQL and PHP-based interface. Hence, in order to work with “Cooking Cake” a webserver with PHP support is required as well as our expanded version of the CAKE Core.
Relating to Figure 7 the interface is divided into two parts. At the top the user can

specify ingredients which are desired ("Do") or ingredients which should be excluded ("Don't"). At the bottom of the interface "Dietary Practices", "Type of Cuisine" or "Type of Meal" can be chosen.
By clicking on the button "Menu Challenge" in the top left corner of the website the user is able to specify each of the described options for three courses which are Starter, Main Dish and Dessert.

6. Exercise Queries

In order to understand how "Cooking Cake" works given Compulsory Task queries - taken from the Computer Cooking Contest website - and their results will be discussed in this section.

Exercise Query #1: Cook an Asian soup with leek.
(Main focus: type of meal, type of cuisine)

As requested the retrieved recipe is an Asian soup with leek. No other restrictions applied.

Exercise Query #2: I would like to have a salad with celery. Please consider that I follow a gout diet.
(Main focus: dietary practice)

Dietary restrictions applied in this query. Therefore only gout diet conform salads with celery were retrieved. No other restriction applied.

Exercise Query #3: Prepare a low-cholesterol dessert with strawberries and avoid citrus fruits.
(Main focus: dietary practice)

The result of this query is a low-cholesterol dessert which includes strawberries and no citrus fruits. Low-Cholesterol and Citrus restrictions applied.

Exercise Query #4: Cook a risotto with carrots.
(Main focus: similarity / modification of recipes)

The result of this query is a risotto with carrots. No other restrictions applied and no modification had to be made.

Exercise Query #5: I would like to cook a pear pancake.
(Main focus: similarity / modification of recipes)

The result of this query is a apple pancake. This result derives from the close similarity between apple and pear. No other restrictions applied.

Exercise Query #9: I do have a filet of beef, carrots, celery, field garlic and cucumber. Potatoes are available, too. For the dessert, we have oranges and mint.

A soup would be preferable for the starter.
(Menu Challenge)

As the result of this query "Cooking Cake" recommends a soup as a starter, a main dish with most of the requested ingredients and a dessert with oranges. No restriction applied and no modification had to be made.

7 Conclusion And Related Work

The main difference towards the systems of the CCC'08 [9] is that "Cooking Cake" is based on the CAKE Framework. Also we want to point out, that the required integration of extended information, like "Type of Cuisine" and "Type of Meal", was implemented by using a Naïve Bayes-Classificator.

On the other hand our system shows similarities in some parts to the COOKIIS'08 which was the only available resource from CCC'08, beside the technical papers provided in the ECCBR 2008 proceeding [10]. For example the ideas behind the existing ontology of this system were partly integrated into the design of our ontology and customized to the needs of our CBR-Software. Therefore our ontology and the COOKIIS' ontology agree in several parts.

With "Cooking Cake" we developed a new CBR-Application for the generic "Collaborative Agent-based Knowledge Engine"-Framework. In this application we were able to implement a highly sophisticated preprocessing method which allows a user to semi-automatically build a large ontology where the manual effort decreases in relation to the increasing amount of ingredients in the ontology.

Currently the focus of our application is to provide a solution for the Competition requirements, but it can be easily extended to a bigger application.

8 References

1. Collaborative Agent-Based Knowledge Engine, Anforderungen - Konzept – Lösungen, http://www.wi2.uni-trier.de/publications/2007_CAKE_Maximini-V2.pdf
2. Swartout, B., Patil, R., Knight, K., Russ, T.: "Toward Distributed Use of Large-Scale Ontologies". In: Proceedings of the Tenth Knowledge Acquisition for Knowledge- Based Systems Workshop. Banff, Alberta (1996)
3. Kuhlen, R.: Experimentelle Morphologie in der Informationswissenschaft, p. 57 Verlag Dokumentation, München (1977)
4. RapidMiner main documentation, http://rapid-i.com/content/view/36/23/
5. CAKE – Technical Report, http://www.wi2.uni-trier.de/publications/2005_technicalReport_cake.pdf
6. Rechernberg, P.: Informatik-Handbuch, p. 987, Hanser-Verlag (2006)
7. Bergmann, R., Fressmann, A., Maximini, K., Maximini, R., Sauer, T., Case-Based Support for Collaborative Business. In: Advances in Case-Based Reasoning ECCBR 2006. pp. 106--120. Springer (2006)
8. Hering, R., Herrmann, F.J.: Herings Lexikon der Küche. Pfanneberg Verlag, Haan-Gruiten (2001)
9. Stahl, A., Minor, M., Traphöner, R. (2008). Call for Participation. Computer Cooking Contest 2008, http://www.wi2.uni-trier.de/eccbr08/index.php?task=ccc
10. Martin Schaaf. Workshop Proceedings. ECCBR 2008. Tharax Verlag (2008)