Workflow Extraction from Cooking Recipes

Kirstin Walter, Mirjam Minor, Ralph Bergmann

Business Information Systems II
University of Trier
54286 Trier, Germany
{walt4701|minor|bergmann}@uni-trier.de

Abstract. Extraction of workflows from unstructured text is a difficult and costly task which is currently performed by human workflow modeling experts. Our paper describes an approach for the automatic extraction of workflows from cooking recipes resulting in a formal description of cooking instructions. A chain of standard and novel information extraction methods is applied. They are dedicated to the special characteristics of textual cooking instructions (verb-centric, restricted vocabulary of ingredients, relatively independent sentences). The intended evaluation of this ongoing work is briefly sketched. The paper is a contribution to the open challenge of the Computer Cooking Contest 2011.

1 Introduction

The International Computer Cooking Contest (CCC) is going into its fourth year. Cooking recipes are given in a case base to be retrieved and reused with the assistance of a computer system. The participating systems compete by providing recipes in order to answer cooking wishes. In addition to three specific tasks like the adaptation challenge the contest includes an open challenge where no particular task is given. This paper presents ongoing work conducted during a diploma thesis and adopts the open challenge by focusing on how workflows can be automatically extracted from cooking recipes. It is an extension of our previous work that deals with the automatic adaptation of workflows [6, 7]. In our last year's CCC open challenge system CookingCakeWf [6], cooking instructions are represented by formal, machine-readable workflows (the so-called *cookery workflows*) that can be adapted automatically to a cooking request of a user by means of case-based reasoning (CBR). In a cookery workflow, cooking activities are represented by *tasks* ordered in a certain *control flow* while the ingredients and their products form the *data flow*. Using workflows as representation to perform adaptations on cooking recipes not only allows to adjust the list of ingredients but also to adapt the cooking activities. So the single activities can be edited, removed or extended by further activities. Thereby workflow similarity measures can be applied to compute similar workflows and recipes respectively. Furthermore the workflow representation gives the possibility to enrich the tasks by further information like corresponding film footage. Up to now the cookery workflows need to be created manually by an expert, which is a time-

consuming and tedious modeling task – we spent between 10 and 20 minutes on each workflow. Furthermore not everybody is able to perform the modeling task: Experience conducted in other domains shows that people being not familiar with modeling languages for workflows have difficulties in modeling workflows [8]. Therefore the modeler of the workflows needs to be domain and workflow expert at the same time. A system being able to automatically extract cookery workflows will considerably reduce the modeling effort and thus avoid the bottleneck of knowledge acquisition, e.g. as component in assistance systems for selecting/adapting recipes.

Our this year's CCC open challenge system addresses a novel extraction component that is to further extend our core system [1] [4]. To realize such a component, the unstructured textual cooking instructions need to be transformed into the cookery workflow representation. In the literature, an information extraction approach to transform cooking recipes into a formal representation is presented for the TAAABLE 3 system [1]. However, the resulting tree representation of the recipes focuses on ingredients rather than on the preparation steps that would be the base activities (tasks) of a formal workflow description. Hence, further information extraction techniques are required to identify the relevant information for a cookery workflow. The required information needs to be extracted from the cooking recipes sentence by sentence forming the basis to generate corresponding workflow parts. The particular parts need to be merged in the last step resulting in a complete cookery workflow.

```
<RECIPE>
  <TI>Beef Noodle Casserole #1</TI>
  <IN>1 1/2 lb Ground beef</IN>
  <IN>1 c Chopped onion</IN>
  <IN>1 cn (16-oz) whole kernel corn; drained</IN>
  <IN>1 cn (10-oz) cream of chicken soup</IN>
  <IN>1 cn (10.75-oz) cream of mushroom soup</IN>
  <IN>1/4 c Chopped pimiento</IN>
  <IN>1 1/2 ts Salt</IN>
  <IN>1/2 ts Pepper</IN>
  <IN>1 c Sour cream</IN>
  <IN>2 pk (8-oz) noodles; cooked</IN>
  <IN>2 c Bread crumbs</IN>
  <IN>4 tb Butter</IN>
   <PR>Brown beef and onions. Combine crumbs and butter. Mix all
  ingredients, except crumbs, and place in large casserole. Top with
  crumbs. Cook at 350 for 30 minutes.</PR>
</RECIPE>
```

Figure 1: Example of a cooking recipe taken from the CCC recipe collection.

[1] http://proj1.wi2.uni-trier.de

The paper is organized as follows: section 2 presents a formal representation of the cookery workflow. Section 3 introduces the types of extraction tasks that result from the representation of cooking recipes as workflows. The extraction process with a chain of different information extraction methods and a corresponding example are presented in section 4. The article finishes with section 5 describing the intended evaluation that aims at comparing the results of the automatic extraction to cookery workflows generated by experts.

2 Cookery Workflow

Recipes retrieved from the web form the input of the extraction component. The current process is configured for the provided CCC recipes. Figure 1 shows an example recipe from this recipe collection in XML. All of these recipes consist of three parts: the recipe title (<TI>…</TI>), a list of ingredients (<IN>…</IN>) and a textual, unstructured description of the cooking instructions (<PR>… </PR>). The process is easily adaptable to other recipe collections as many internet platforms[2] which specialize on cooking have similar XML representations for their cookery data. In order to represent such a cooking recipe as cookery workflow corresponding data for each workflow component has to be identified. Thereby we define cookery workflow as traditional workflow [12] applied to the cooking domain. The following three classical workflow components are considered: the tasks, the control flow and the data flow.

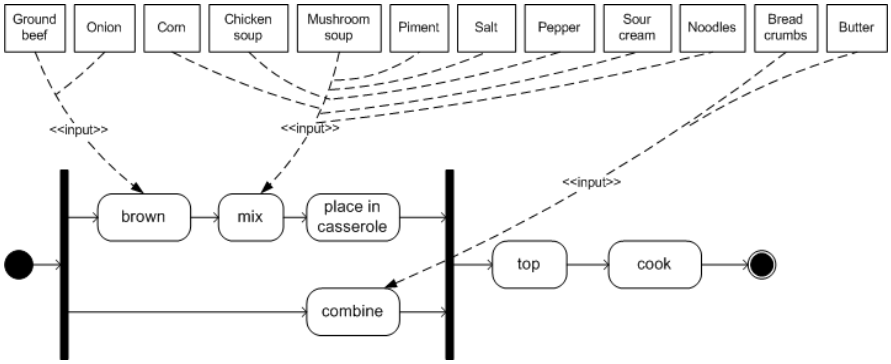

Figure 2: Example of cookery workflow.

The tasks are formed by the single cookery actions, which are mainly characterized by cookery verbs like *brown* or *mix*. As the tasks process or even combine the ingredients, the data flow between the tasks is formed by the ingredients and their aggregates. An aggregate is e.g. dough consisting of flour, water and yeast. The recipe

[2] Examples: *http://www.allrecipes.com* and *http://www.freecookingrecipes.net*

defines a specific order in which the cooking instructions need to be performed: so a sauce must be prepared first before it can be poured over pasta. The specified order forms the control flow of the workflow. A resulting cookery workflow created by hand from the sample recipe in Figure 1 is presented in Figure 2: e.g. in the upper of the two parallel branches ground beef and onions are browned. As notation we use UML activity diagrams which are extended by *Data Objects* representing the ingredients and *Data Links* indicating that an ingredient forms the input of a task (for a more detailed description see [6]).

3 Extraction Tasks

In the extraction process several aspects have to be considered. According to section 2 three different extraction tasks can be identified:

(1) *Extraction of cookery activities*: A workflow task is to describe a single cookery activity which is mainly defined by a cookery verb. Refer to Figure 3 for an example of extraction: the cookery verb *sauté* is extracted and forms a task in the resulting workflow part.

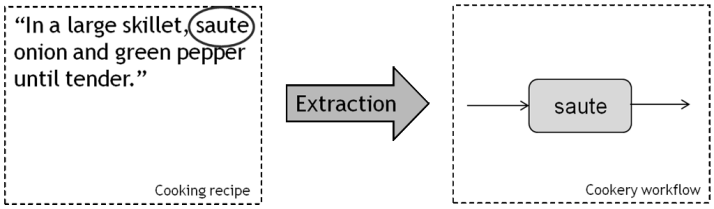

Figure 3: Example of extraction of a cookery verb.

(2) *Extraction of ingredients:* The input and the output of each task are described by the ingredients of the recipe. Apart from the ingredients included in the list of ingredients attention has to be paid to aggregates which are created by combining several ingredients. For reasons of complexity reduction we will not consider aggregates, just the base ingredients. According to our previous work this seems to produce quiet satisfying adaptation results anyway. In Figure 4 an example is depicted: After the ingredients *onion* and *green pepper* have been extracted, for each of them a new Data Object is created and linked to the corresponding task.

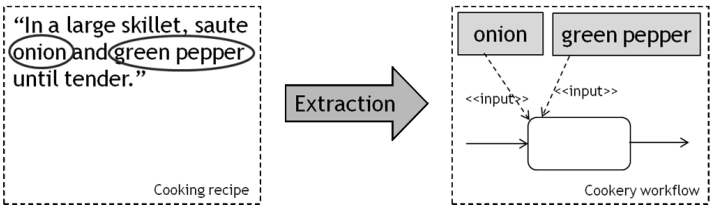

Figure 4: Example of extraction of ingredients.

(3) *Extraction of the order of the cooking activities:* In addition the recipe gives a particular order for the cooking activities which is in most cases a total order. This order must be reflected in the workflow defining the control flow. In the example of Figure 5 a sequential order for the two instructions is given, hence the corresponding tasks are ordered sequentially.

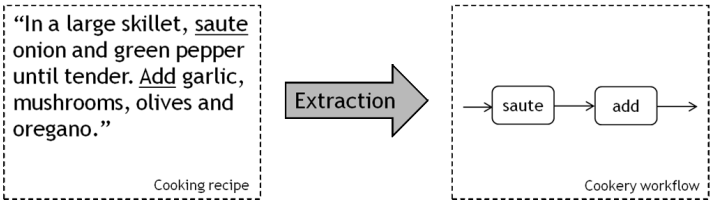

Figure 5: Example of extraction of order of cooking activities.

This work further focuses on the extraction of the cookery verbs. To extract the relevant information, methods taken from information extraction are applied. In this context two decisions have to be made: First, whether to apply rule-based or statistical methods and second, whether to develop a hand-coded or a learning-based system [10].

1. Rule-based systems work with hard predicates and are easy to develop and to interpret. Statistical systems base their decisions on the weighted sum of several predicate firings, so that they are more suitable for domain independent applications but more complex in development. As our domain is well defined and closed rule-based methods seem to be more appropriate.
2. While hand-coded systems require a domain expert creating the rules, learning-based systems use an amount of manually labeled data (which means a high initial effort) to train models of extraction. The decision is made for a hand-coded system because it does not require a high initial effort to get first results.

So our first approach for the intended extraction process applies hand-coded rules.

4 Intended Extraction Process

Figure 6 illustrates the intended extraction process – the word 'intended' is used to indicate that ongoing work is described and that parts of this process could still be subject to change. This process mainly consists of three steps. During the first step, the relevant information included in the recipe is labeled. To reduce the implementation effort an existing framework is applied here. The most common frameworks are GATE [2, 3] and Stanford CoreNLP[11]. The decision is made for GATE because it is well documented, popular, and in addition it provides a component-based model making it easy to customize. GATE is an open source software that has been in active use for all kind of natural language processing tasks for 15 years now. It consists of two components which are both used for our work:

1. GATE Developer is a graphical development environment enabling the development, evaluation and deployment of software components that process human language.
2. GATE Embedded is an object-oriented framework, written in JAVA, which allows to integrate the functionality of GATE Developer in own applications.

For each document to be processed GATE generates a model of annotations which is used to encode all the data that is read and produced by the different modules of GATE. To perform the first step of our intended extraction process the information extraction component of GATE (called ANNIE) is applied.

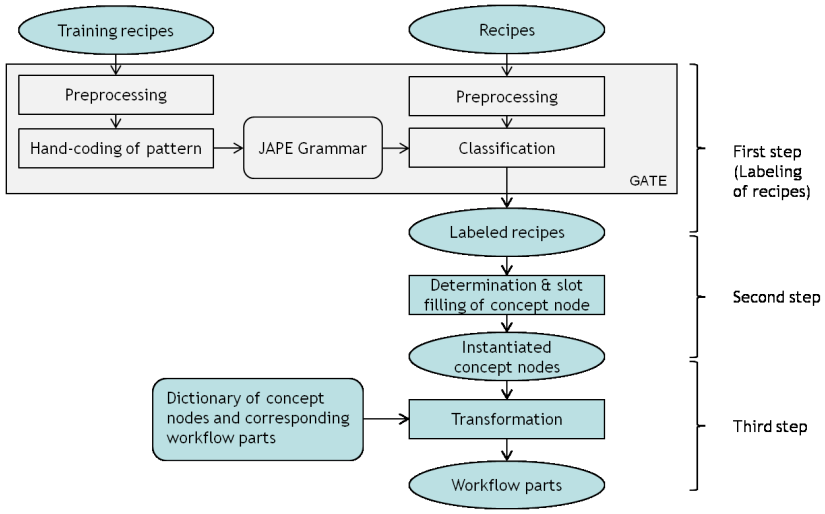

Figure 6: Intended extraction process.

Before recipes can be automatically labeled/annotated, corresponding rules have to be created during a previous training phase. For this, GATE provides the language JAPE (Java Annotations Pattern Engine) which allows to describe pattern to match and

annotations to be created as result. A domain expert creates the required JAPE rules based on a set of training recipes, consisting of 15 pasta recipes taken from the CCC recipe book. These training recipes need to be preprocessed before. Here GATE offers a variety of preprocessing components like a POS-Tagger[3]. An example for a JAPE rule is shown in Figure 7. Such a rule always consists of a precondition (left side) and an action (right side). While the left side gives the identified pattern, the right side specifies the action to be performed on the detected pattern. In this example the annotation "break" is added to each token, that follows the word "until" and that has the category "JJ" (POS-Tag for an adjective). All rules in a body form the JAPE grammar which is used to automatically label recipes. Therefore it is important to preprocess these recipes in the same way as the training recipes. The output of GATE is a labeled/annotated recipe. An example is shown in Figure 8: e.g. the cookery verb is annotated with "action". In the next step suitable concept nodes are determined sentence by sentence and the corresponding slots are filled. Thereby a mechanism is required to choose the best fitting concept node for each sentence such that the instantiation of multiple concept nodes for one sentence is prevented.

Concept nodes as part of the *Selective Concept Extraction* were introduced by Wendy Lehnert within the system *CIRCUS* [5, 9]. Each concept node contains a trigger verb triggering its instantiation and a set of slots extracting the relevant information from a sentence. When a concept node is instantiated its slots are filled with the corresponding information detected in the text. As they put the main focus on verbs and not on nouns like traditional extraction models concept nodes seem to be best suited to be used for our system focusing on the extraction of cookery verbs. An instantiated concept node results from the second step. In the third step theses concept nodes are transformed into the according workflow elements. Therefore a dictionary is provided containing the corresponding workflow part for each concept node.

```
Rule: BreakCondition1
({Token.string == "until"}):temp
({Token.category == JJ}): break
-->
break.break = {rule = "BreakCondition1"}
```

Figure 7: Example of a JAPE rule.

"In a large skillet, [action saute] [ingredient onion] and [ingredient green pepper] until [break tender]."

Figure 8: Example of a labeled sentence of a recipe.

[3] A POS-Tagger annotates each word of a text according to its particular part of speech, e.g. noun or verb.

Figure 9a) shows an example of such a pair in this dictionary: the concept node includes the slots *action*, *ingredients* and *break condition* which is transformed into a workflow part consisting of a single task, a loop and a number of Data Objects and Data Links depending on the size of the slot *ingredients*.

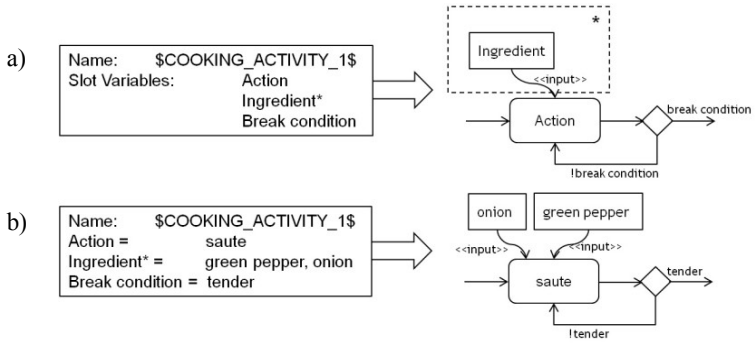

Figure 9: Concept node-workflow-pair (a) in dictionary (b) instantiated

The dictionary needs to be created manually in advance by a domain expert. Based on the corresponding concept node-workflow-pair the concept nodes are transferred in workflow parts. Figure 9b) gives an instantiation for the concept node-workflow-pair of Figure 9a). The slots of the instantiated concept nodes are filled with values, e.g. the value of slot *action* is "saute" and the size of the slot *ingredients* is two, comprising the values "green pepper" and "onion". Here, the order of occurrence of the ingredients for the slots is ignored. The corresponding instantiated workflow part consists of the task "saute" and two Data Objects representing the ingredients and the loop control-flow element.

At the moment the first step of the extraction process is implemented including a suitable configuration of GATE and the creation of the required JAPE rules. So the cookery verbs are identified by means of a couple of JAPE rules in combination with a dictionary containing the most common cookery verbs.

5 Intended Evaluation

During the evaluation the quality of the system needs to be assessed. As described in the previous section, the process consists of three transformation steps, each of which requires its own evaluation.

First, the quality of the labels assigned by GATE is evaluated. Therefore the automatically labeled recipes are compared with reference samples which have been manually labeled by a domain expert. GATE offers an evaluation component which is used to perform the comparison and to calculate figures like recall, precision and F-score. As experimental setup an excerpt from the provided CCC recipe book just

comprising the pasta recipes will be used. The results will be presented during the competition.

To evaluate the second step, we examine if the best matching concept node is each instantiated. Thereby different criteria can be applied to decide which concept node is instantiated if more than one concept node is matching:

1. The concept node with the highest number of slots is instantiated.
2. The concept node with the highest percentage of filled slots is instantiated.
3. The concept node with the highest total number of filled slots is instantiated.

For each criterion precision and recall are determined to assess the respective quality and to determine the criterion with best coverage. Furthermore the discussed approach using concept nodes is contrasted with a single general template that is applied for each instantiation so that the determination of the best matching concept node would cease. To assess this alternative approach precision and recall are calculated and compared to the results of the concept node approach.

The last step of evaluation is formed by the evaluation of the resulting workflow parts: A user experiment is conducted to analyze if the resulting workflow parts are suitable to represent the corresponding part of the recipe. A disagreement may be further used to improve the according concept node-workflow-pair included in the dictionary.

6 Summary and Future Work

In this paper we presented the general idea for an extension of our previous work to extract workflows from cooking recipes. We introduced cookery workflows as formal representation for cooking recipes and outlined the three steps of the intended extraction process. First the recipes are annotated by using the framework GATE which requires to configure GATE and to create domain specific annotation rules. Then the best matching concept node is instantiated by filling the slots of the concept nodes with the corresponding information. By means of a dictionary the resulting concept nodes are transformed into appropriate workflow parts. This paper deals with ongoing work conducted during a diploma thesis, but until the competition a first implementation and respective evaluations as outlined in the previous section will be available so that the results can be presented during the competition.

Future work will address the maintenance of the case base comprising the generated cookery workflows. Recipes consist of human language which is prone to irregularities, possibly resulting in incorrect workflows. As the workflows are processed by means of CBR which is a robust technique a certain extent of incorrectness can be accepted. To reduce this extent the user has to manually check each extracted workflow and, if required, to correct it. Injecting this feedback in the process to improve the result may be subject to future work. Future work will also concern further domains. As we are aware of the fact that the workflow approach is particularly well suited to the cookery domain, a first focus will be set on domains having similar characteristics, e.g. generating workflows from how-tos on the internet.

7 References

1. Blansché, A., Cojan, J., Dufour-Lussier, V., Lieber, J., Molli, P., Nauer, E., Skaf-Molli, H., Toussaint, Y.: *TAAABLE 3: Adaptation of ingredient quantities and of textual preparations*. Workshop Proceedings ICCBR 2010, pp. 189-198, 2010.
2. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: *GATE: an Architecture for Development of Robust HLT Applications*. Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, 2002.
3. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Ursu, C.: *The Gate User Guide*, http://gate.ac.uk, 2002.
4. Fuchs, C., Gimmler, C., Günther, S., Holthof, L., Bergmann, R.: *Cooking Cake*. Workshop Proceedings ICCBR 2009, pp. 259-268, 2009.
5. Lehnert, W.: Symbolic/Subsymbolic Sentence Analysis: *Exploiting the Best of Two Worlds*. Advances in Connectionist and Neural Computation Theory 1, pp. 135-164, 2001.
6. Minor, M., Bergmann, R., Görg, S., Walter, K.: *Adaptation of Cooking Instructions Following the Workflow Paradigm*. Workshop Proceedings ICCBR 2010, pp. 199-208, 2010.
7. Minor, M., Bergmann, R., Görg, S., Walter, K.: *Towards Case-Based Adaptation of Workflows*. Proceedings ICCBR 2010, pp. 421-435, 2010.
8. Recker, J., Safrudin, N., Rosemann, M.: *How Novices Model Business Processes*. Business Process Management – BPMN 2010, pp. 29-44, 2010.
9. Riloff, E., Lehnert, W.: *Information Extraction as a Basis for High-Precision Text Classification*. ACM Transactions on Information Systems 12(3), pp. 296-333, 2004.
10. Sarawagi, S.: *Information Extraction*. Foundations and Trends in Databases 1(3), pp. 261-377, 2007.
11. The Stanford Natural Language Processing Group: *Stanford CoreNLP*, http://nlp.stanford.edu/software/corenlp.shtml.
12. Workflow Management Coalition: *Workflow Management Coalition Glossary & Terminology*, http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf, 2007.