Proceedings of the Cooking with Computers workshop (CwC)

Amélie Cordier and Emmanuel Nauer (Eds.)

Preface

We are pleased to present the Workshop Proceedings of the first Cooking with Computers workshop which is held in Montpellier, France, on August the 28th 2012, during ECAI.

The "Cooking with Computers" workshop aims at gathering researchers from as many as possible fields of AI. A core application domain, which is cooking, is fixed and the main objective of this workshop is to show how some AI existing approaches could be used to solve problems in this domain.

Since 2008, the International Case-Base Reasoning community organises the Computer Cooking Contest. The main goal of Computer Cooking Contest participants is to design systems able to propose new recipes fitting specific requests by adapting existing recipes. Contributions to the Computer Cooking Contest are mainly Case-Based Reasoning oriented. Our goal with this workshop is to go one step further in the exploration of what computers and especially AI can do for cooks!

Our motivation lies in the fact that there is numerous other initiatives investigating how specific topics of AI can contribute to cooking. These initiatives are always domain-specific. Our goal is to set up a cross-domain event to promote communication between researchers of various domains of AI.

This year, we have selected five articles for a long presentation and two articles for a short presentation. These articles are available in this document.

We hope you willl enjoy the workshop program!!!

Amélie Cordier and Emmanuel Nauer, August 2012

Table of Contents

Co-Chairs

Amélie Cordier
Université Lyon 1, LIRIS, France
amelie.cordier@univ-lyon1.fr

Emmanuel Nauer
LORIA, Université de Lorraine, France
emmanuel.nauer@liris.cnrs.fr

Programme Committee

Belén Diaz Agudo (Universidad Complutense de Madrid, Spain)
Kerstin Bach (German Research Center for Artificial Intelligence DFKI GmbH, Germany)
Robert Comber (Newcastle University, United Kingdom)
Olivier Corby (INRIA, Sophia-Antipolis, France)
Amélie Cordier (Université de Lyon 1, LIRIS, France)
Sylvie Després (LIM&BIO, Université Paris 13, France)
Antoine Durieux (Chef Jérome, France)
Jérome Euzenat (INRIA Grenoble Rhone-Alpes, France)
Patrick Gallinari (LIP6, Université Pierre et Marie Curie, France)
Eyke Hullermeier (Universitat Marburg, Germany)
Ichiro Ide (Nagoya University, Japan)
Mirjam Minor (University of Trier, Germany)
Chamin Morikawa (Tokio University, Japan)
Emmanuel Nauer (LORIA, Université Lorraine, France)
Thomas Roth-Berghofer (University West London, United Kingdom)
Franois Scharffe (LIRMM, Montpellier, France)
Tomohide Shibata (Graduate School of Informatics, Kyoto University, Japan)
Ralph Traphoner (Attensity Europe GmbH, Germany)

Design and Development of a Learning Support Environment for Apple Peeling using Data Gloves

Saori OTA[1], Masato SOGA[2] , Nami YAMAMOTO[3] and Hirokazu TAKI[2]

Abstract. We designed and developed a learning support environment that advises a learner about correct apple holding and correct knife handling for improving apple peeling. The learning support environment is designed from an apple peeling expert's motion data. We verified the learning effect by comparing experimental group with a control group. In the training phase of the evaluation experiment, each learner in the control group only watched a movie of the learner's own apple peeling. The experimental group not only watched such a movie, but also used our learning support environment. After the evaluation experiment, we administered a questionnaire survey. Results show that the experimental group obtained more concrete and varied awareness than the control group had.

1 INTRODUCTION

A decline in manual skills of the fingers and hands has been reported [1]. Figure 1 shows results of a questionnaire administered to junior high school students. In terms of the frequency of cooking, about half of them answered that they seldom cooked. Furthermore, only about 30% of the students answered that they were capable of peeling with a kitchen knife. Consequently, we infer that they have been caught in a vicious circle in which they have too few cooking opportunities to improve their kitchen knife skills.

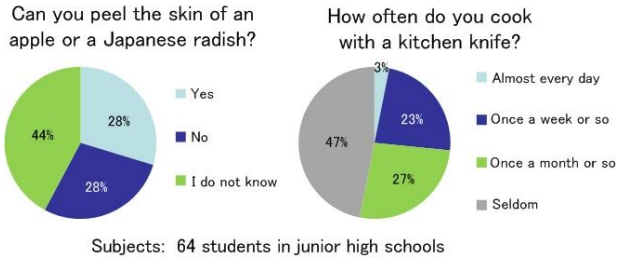

Figure 1. Results of a questionnaire administered to junior high school students

Previous studies of movements during cooking with a kitchen knife include Suzuki's studies of the effect of using a kitchen knife that has been developed specially for children on skills learning [1] and of the size of cut pieces to acquire kitchen knife skills [2]. However, they deal with a chopping motion, and They did not report any study on the peeling motion of an apple. The peeling motion is more difficult than chopping motion.

For skills learning, the early stage of skill acquisition is extremely important [3]. Once skills are wrongly ingrained in the early stage, it is difficult to correct them. The concern arises about its effect on the progress of learning, which suggests the importance of teaching the correct way to students who answered "I do not know" to the question asking about their capability of peeling.

Home economics classes, which provide opportunities to acquire kitchen knife skills, have been shortened every time the national curriculum is revised [4]. Furthermore, a teacher must teach numerous students during the class. It is difficult to say that such an environment is ideal for learning.

Based on the points presented above, we think that an environment in which students can efficiently learn the correct way rapidly is necessary to master peeling.

Regarding the current situation, this study is intended to design and develop a learning support environment in which learners can learn correct ways of holding and peeling an apple and to examine the learning effect. Subjects were those who were unable to peel at all or who peeled using an incorrect technique at that time.

2 LEARNING SUPPORT ENVIRONMENT

This section presents a description of details and how to use the developed learning support environment.

2.1 Device to be used

For this study, we used a data glove (for the right hand and the left hand, 5DT 14 Ultra series; Fifth Dimension Technologies) to measure finger motions. The fiber-optic data glove has 14 sensors. Sensors in the joint parts output the stretching state as 0 and the bending state as 1. Sensors in the crotch of the fingers output the open state as 0 and the closed state as 1. Each sensor has a designated name (Fig. 2).

2.2 System design

The learning support system (Fig. 3) is intended to support home economics teachers when giving advice to students based on objective data when teaching them apple peeling skills. The input is a learner's finger motion. The outputs are the learner's motion data and advice for the learner.

[1] Mitsubishi Motors Corporation, Japan
[2] Faculty of Systems Engineering, Wakayama University, Japan, email: soga@sys.wakayama-u.ac.jp
[3] Faculty of Education, Wakayama University, Japan

The procedures for use are as follows. A learner wears the data gloves and calibrates them. Then, when the learner holds an apple and a kitchen knife to perform peeling, the system judges whether the apple and kitchen knife are being held correctly. Subsequently the learner peels the apple. After peeling has been completed, the system gives advice by text messages and illustrations. Motion data and results of analysis are not presented directly to subjects who used the system.

Figure 2. Name of each data glove sensor

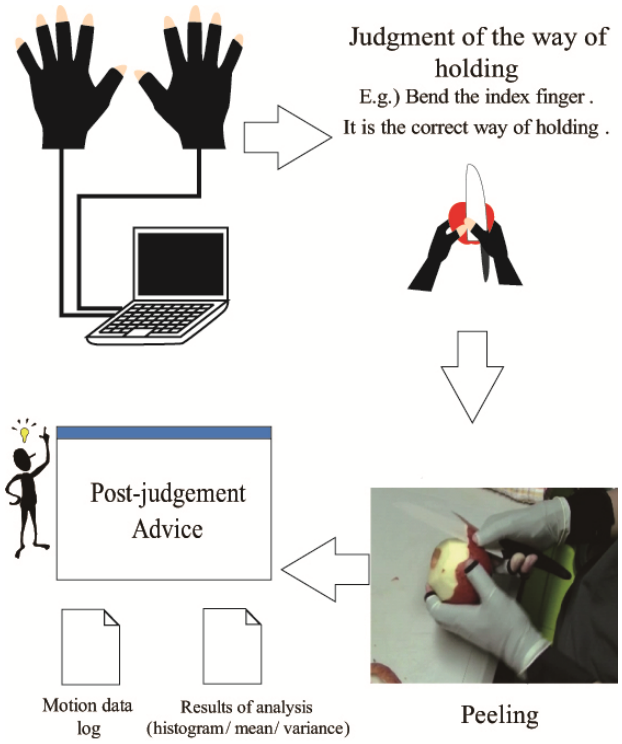

Figure 3. System overview

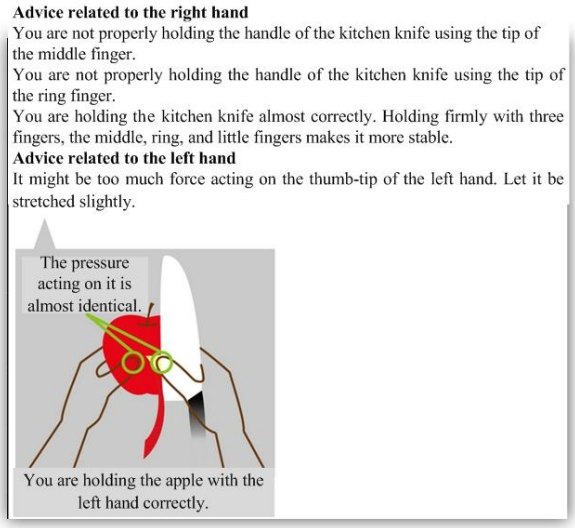

Figure 4. Example of advice

2.3 Advice examples

For learners who used the system, advice is presented after peeling, as portrayed in Fig. 4. Table 1 is an example of the threshold between sensor values for use to generate advice. The threshold is set based on the experts' data in the next subsection.

Table 1. Example of advice generation

1	Condition Message	0.3 < Thumb/Index sensor value "Hold an apple spreading the thumb."
2	Condition Message	0.2 < IndexNear sensor value "You are putting too much force on the index finger."
3	Condition Message	0.05 < MiddleNear sensor value "You are putting too much force on the middle finger."
4	Condition Message	0.06 < RingNear sensor value "You are putting too much force on the ring finger."
5	Condition Message	0.15 < LittleNear sensor value "You are putting too much force on the little finger."
Condition Message		If all formulas 1–5 are FALSE "The way of holding with the left hand is OK."

2.4 Data gathering

To determine thresholds to present advice as shown in Table 1, we gathered data from 7 experts. The condition to be an expert was to have met the qualifying standard for the *Katei ryori gino kentei* (= Certification Examination for Home Cooking)[5]. Figure 5 presents an example of a histogram showing sensor values across the experts' left-hand MiddleNear sensors as a cumulative bar chart. The relevant part of the sensor is shown in Fig. 6. The comparative frequency of the histogram is distributed around small values. According to previous studies, these are characteristics of the way

in which experts hold an apple [6]. Therefore, we set the threshold of the values for the left-hand MiddleNear sensor as 0.05. When the threshold is exceeded, advice will be presented to the subject who used the system.

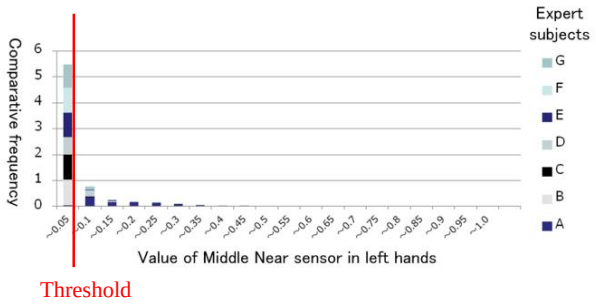

Figure 5. An example of histogram of sensor values (Example: left-hand MiddleNear sensor)

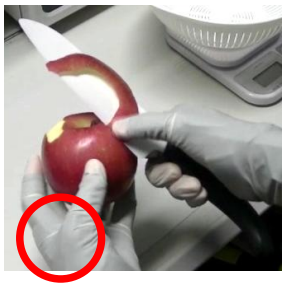

Figure 6. Relevant part in the left-hand MiddleNear sensor.

3 EVALUATION EXPERIMENT

This section explains an evaluation experiment that was conducted to verify the usefulness of the learning support environment.

3.1 Procedures of the experiment

The experiment procedures are portrayed in Fig. 7. Six subjects were divided into an experimental group and a control group. Both groups performed peeling three times.

In the pre-questionnaire shown in Table 2, we asked how often they cooked and whether they were capable of peeling. With regard to the frequency of cooking, about half answered "seldom" and half answered "once a week or so". Although one subject answered "Yes, I can peel," we did not regard the person as an expert but treated the person as a subject for the evaluation experiment because the person did not peel correctly.

The first trial of peeling was conducted as pre-test to check the skills at the stage.

The second trial of peeling was for training. In the trial, the experimental group used the system and received advice after watching an example video. The control group only watched the example video. Subsequently, an attitude survey was administered through a questionnaire.

The third trial of peeling was conducted as post-test to measure the level of improvement. Outcomes such as disposal rates were organized into the results of the experiment.

Table 2. Subjects' data list

		Capability of peeling	Frequency of cooking	Dominant hand
Exp. group	Subject A	No	Once a week or so	Right-handed
	Subject B	No	Seldom	Right-handed
	Subject C	Yes	Once a week or so	Right-handed
Cont. group	Subject D	No	Seldom	Right-handed
	Subject E	No	Once a week or so	Left-handed
	Subject F	No	Seldom	Right-handed

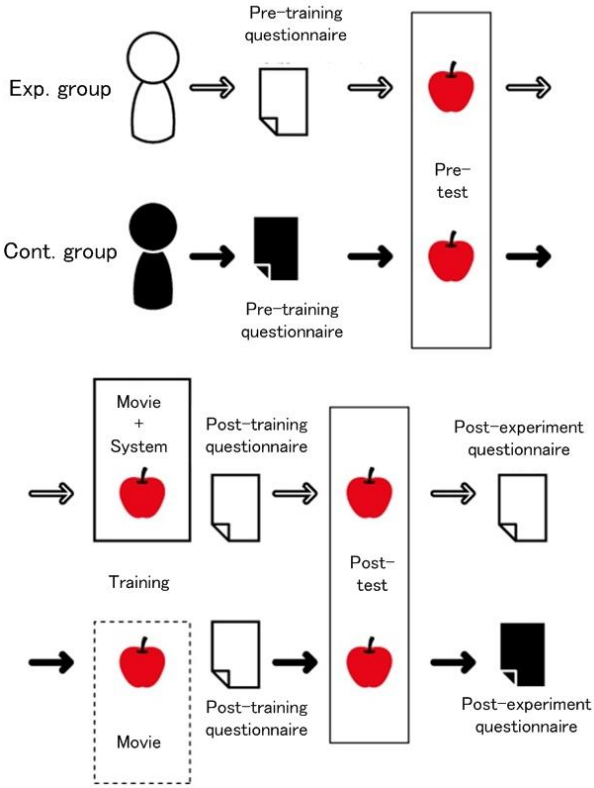

Figure 7. Procedures of the experiment

Figure 8. System utilization (in the experimental group)

3.2 Example video

The example video is a video watched by both groups (Fig. 9). We subtitled explanatory text for the video, showing a home economics teacher demonstrating peeling on the video. The explanatory text was the qualifying standard for the certification examination for home cooking and explanations that the teacher gave students.

Figure 9. Capture of the video

4 RESULTS OF THE EVALUATION EXPERIMENT

In this section, we describe the results of an evaluation experiment for learners.

4.1 Experimental results

In Table 3, we compared the results of the pre-test (= first trial) with the post-test (= third trial) of peeling by subject. The disposal rate 15% is regarded as ideal, and we regarded improvement as occurring when the difference from 15% became smaller and worsening when it became greater.

Improvement was found because the experimental group and the control group were able to shorten the time to peel. A reason why the disposal rates rose is presumed that they peeled the skin thickly to reduce the number of times the skin was being cut off.

Table 3. Results of the evaluation experiment

Experimental group	Subject A	Subject B	Subject C
Disposal rate (%)	23.9	12.5	22.9
Difference from the first trial	6.3% worsened	5.1% improved	2.4% improved
Only skin disposal rate (%)	9.2	8.4	11.6
Difference from the first trial	0.8%	-10.0	1.5%
Time (s)	121	287	150
Difference from the first trial	-66 (-36%)	-88 (-24%)	-12 (-8%)
Number of times being cut off	0	7	3
Difference from the first trial	-11	-8	1

Control group	Subject D	Subject E	Subject F
Disposal rate (%)	23.0	24.1	16.9
Difference from the first trial	6.1% worsened	1.4% improved	2.2% improved
Only skin disposal rate (%)	9.4	12	10.7
Difference from the first trial	-0.6%	1.1%	4.6%
Time (s)	190	96	191
Difference from the first trial	-80 (-30%)	-21 (-18%)	-28 (-13%)
Number of times being cut off	2	2	5
Difference from the first trial	-35	-2	-4

4.2 Results of the post-training questionnaires

We organized the results of questionnaires conducted after the second trial, which was a training session, in Table 4 and Table 5.

According to Table 4, the experimental group was better in terms of the number and the concreteness of awareness. By verbalizing, awareness was more useful for peeling the next time.

Awareness, which is regarded as engendering future improvement, is presented in red bold letters. The experimental group gained more concrete and numerous instances of awareness. In addition, many occasions of awareness led to improvement. Subject F did not gain any awareness at all in terms of the left hand.

Table 4. Post-training questionnaire 1

Hand holding an apple		
Exp. group	Subject A	"I was only conscious of the thumb on how to hold an apple. Other fingers were neglected."
	Subject B	"I held the apple too firmly. It did not work with the hand holding a kitchen knife. The fingers from the index to the ring finger were spread open. The starting position of cutting was too low. The power balance of the fingers from the middle to the little finger was not good. I should have supported an apple lining up the fingers from the index to the little finger, but I held it with them bending."
	Subject C	"I concentrated on moving the apple. It was easier to peel upward."

Cont. group	Subject D	"I moved more restlessly than the example. I feel that too many times the position to hold is changed."
	Subject E	The angle to incline.
	Subject F	Nothing in particular.

Table 5. Post-training questionnaire 2

Hand holding a kitchen knife		
Exp. group	Subject A	Hold of the kitchen knife. Hold of the handle of the kitchen knife. The way of moving the thumb is more closely and shorter in the distance than the example.
	Subject B	"I peeled like scooping. For that reason, I often returned back and started peeling again. The surface was not smooth and like a staircase. The position placing the thumb is not fixed, such as on the edge or far from the edge. At the start of peeling, the direction of the edge was not fixed and dangerous."
	Subject C	"I was conscious to position the thumb just as shown in the example."
Cont. group	Subject D	"It is slipping upward."
	Subject E	"I peeled using the upper part of the kitchen knife. The position was not stable." Motion of the thumb.
	Subject F	Placing the thumb on the position capable of peeling the skin of an apple.

4.3 Results of the post-experiment questionnaires

We next describe the results of questionnaires conducted after all three trials of peeling were completed.

To the question for the experimental group, "Did you understand the content of advice?" all answered "Yes". In response to the question "Was it easy to follow the advice?" all answered "No".

- Subject A: I cannot put it into practice even being aware of the left hand.
- Subject B: I am absorbed in peeling and forget the contents of advice.
- Subject C: It was difficult to follow the advice telling me that the fingers of the right hand holding a kitchen knife were not bent sufficiently.

The results presented above are regarded as conditions in which "it can be understood as formal knowledge but it is not acquired as body knowledge" and in which "it is beyond the limitations of short-term memory." The latter will be resolved by adding the design capable of real-time diagnosis to the system.

We received the answer "improved" in terms of peeling an apple from both groups. When asking the experimental group the reason, we obtained the following answers.

- Subject A: Because the amount of the skin being cut off decreased.
- Subject B: Because only the skin disposal rate was reduced.
- Subject C: Because I became able to peel faster.

All three of the items shown above appear as numerical results. To maintain an appetite for learning, it will be necessary to present numerical data that are both visible and easy to understand. However, the numerical data show "results", and it is predictable that it will not engender the improvement in the "process". For example, even if peeling is performed incorrectly, it is still possible to achieve a low disposal rate without the skin being cut off.

"Gained awareness" in the control group is described in Table 6. Subject F remained unable to understand how to hold the apple even after all three trials.

Table 6. Results of the post-experiment questionnaire for the control group

Subject D	Differences existed in motion of the hands. Learning how to hold a kitchen knife was particularly helpful. I tried to practice the way of holding a kitchen knife, but returned it in the middle.
Subject E	The way of holding an apple. The position of holding an apple with the left hand. How to hold a kitchen knife. How to use the thumbs of both hands.
Subject F	Pressing the part of the skin to be peeled with the thumb of the right hand. I do not understand how to hold a kitchen knife.

4.4 Discussion

We believe that this evaluation experiment demonstrated the usefulness of the system. In questionnaire 3 for the experimental group which used the system, the outstanding answer was that it was difficult to follow the advice. However, probably the experimental group which used the system was able to have higher awareness for improvement than the control group, which did not use the system.

Learners exhibited unexpected motion for which the system generated advice. We found a way of holding in which bending of the MP joints of the left hand holding an apple and a learners' particular way of peeling which was regarded as attributable to the angle of the index finger of the right hand holding a kitchen knife. They are expected to be sorted out by gathering widely various motion data from learners.

The data gloves used this time are less invasive and are suitable to detect a peeling motion. If devices were selectable according to the learner's level, then points that they became aware of this time could be addressed. The width of the apple skin can be displayed through a head mounted display. If the relative position and the angle between a kitchen knife and hands were available by positioning sensors, then more detailed advice could be given.

5 Conclusion

For this study, we designed and developed a learning support environment using motion data, which are objective data in the motion of peeling an apple.

Results demonstrated that the control group in the evaluation experiment improved somewhat through carefully watching an expert's video. However, the observation is the learners' subjective observation and if they themselves do not find new discoveries through the observation, there will be limitations to rapid

improvement. However, the experimental group was likely to widen their views because they had advice from the system when watching the video, in addition to the observation from own perspective. In fact, the experimental group found more concrete and numerous differences between the expert and themselves than the control group.

Although it remains as a future challenge, there are apparently limitations to learning using only motion data through data gloves. To peel an apple, one uses not only one's own body but also a tool, which is an extension of the body. Using the relative position and the angle between a tool and the body, advice can be broadened to a great extent.

ACKNOWLEDGEMENTS

We would like to thank Ms. Mayumi Nishioka, belonging to Nishihama junior high school in Wakayama city. We are grateful to her for her advice from the viewpoint of home economics teacher.

REFERENCES

[1] M. Takahashi, C. Ooeda and E. Sato, Daily Activities Concerning Skilfulness in Fingers and Hands - A Questionnaire Survey for Using A Hand and Consciousness of Movement -, Mejiro Journal of Social and Natural Sciences, Vol.7, 29-39, (in Japanese), (2011)

[2] Y. Suzuki, Effect of Kitchen Knife Specifically Developed for Children on Skill Learning, Journal of Home Economics, Vol.57, No.3,169-177, (in Japanese), (2006)

[3] Y. Suzuki, Object Size, Cutting Skills and Their Acquisition Thereof, Journal of Home Economics, Vol.55, No.9,733-741, (in Japanese), (2004)

[4] M. Suzuki, A. Sugano and A. Fukuda, Home Economics Education Linked together Curricula at Primary and Junior High Schools, - From Decision Making Point of View -, Bulletin of Center for Educational Research and Practice, No.2, 83-94, Faculty of Education and Human Sciences Niigata University, (in Japanese), (2003)

[5] F. Kagawa, Textbook for Certification Examination for Home Cooking, Kagawa Nutrition University, (in Japanese), (2008)

[6] S. Ota, Comparative analysis of experts' and beginners' skills during apple peeling, Graduation thesis, Faculty of Systems Engineering, Wakayama University, (in Japanese), (2010)

Grocery Re-identification using Load Balance Feature on the Shelf for Monitoring Grocery Inventory

Rena Kamoda [1] and **Mayumi Ueda**[2] and **Takuya Funatomi** [1] and **Masaaki Iiyama** [1] and **Michihiko Minoh** [1]

Abstract. We propose a method to monitor the grocery items in a home refrigerator. Using load sensors mounted under a shelf of the refrigerator, our method matches groceries that are put into and taken from the refrigerator. Our prototype load-sensing board uses four load sensors to acquire the weight and position of the grocery items, and we use this data to re-identify the groceries. Detailed experiments show that this feature can accurately re-identify grocery items and thus provide constant monitoring of refrigerator contents.

1 Introduction

You may have experienced upon returning from a shopping trip that you already had some of the purchased groceries in your refrigerator, or that you forgot to buy certain groceries. You may have also found yourself trying to remember the contents of your refrigerator when shopping for dinner in a supermarket. A system for monitoring the contents of a refrigerator would be helpful in such situations. We aim to construct a system that sends the image and weight of the grocery items in the refrigerator to the user, who can receive this information conveniently via smartphone.

If a system consistently monitors what comes into and what goes out of a refrigerator, it effectively shows the contents of the refrigerator to the user. When a grocery item enters the refrigerator, the system registers its image and weight to an inventory list. When a grocery item exits the refrigerator, the system matches it to an entry in the inventory list and deletes the entry. This matching is done using the information registered in the list. In this paper, we discuss what kind of information is useful for grocery re-identification and propose a method to re-identify groceries.

While there exist some smartphone applications that help the user to manage the grocery in his/her refrigerator, these applications require the user to manually enter what is stored or taken out. Such manual entry proves troublesome; ideally, the process should be automated. Some systems identify groceries using barcodes or radio frequency identification (RFID) tags; however, this is unacceptable from a sanitary viewpoint. Instead of using tags, we propose the use of information that is naturally available to the system. We discuss an appearance feature and a load balance feature and propose the use of load balance feature for grocery re-identification.

In section 2, we discuss the grocery inventory system. In section 3, we describe a method of re-identification that uses the load balance feature. Finally, in section 4, we demonstrate the effectiveness of this method through some experiments.

[1] Kyoto University, email: kamoda@mm.media.kyoto-u.ac.jp
[2] University of Marketing and Distribution Sciences

Figure 1. Groceries in a refrigerator displayed on a smartphone

2 Grocery inventory system

2.1 Monitoring inventory with tags

There exist smartphone applications that aim to help users to manage grocery in their refrigerators. However, these applications require users to manually register each refrigerator transaction. We take an automated approach. When a grocery item is stored in the refrigerator, the system registers it into an inventory list. When a grocery item is taken out of the refrigerator, the system re-identifies it by matching it to the groceries in the inventory list. By this approach, we can automatically know what is in the refrigerator in real time. We just monitor what is stored in and taken out of the refrigerator and we don't care about the groceries after they are taken out of the refrigerator: if the grocery which are taken out of the refrigerator is stored again, we treat it as a new entry.

To achieve such re-identification, the system needs some information about each grocery. Identification tags are commonly used for this purpose. Retail stores worldwide use barcodes and factory storage commonly uses RFID tags. Barcodes and RFID tags provide reliability: the system can identify each product with near-certainty. However, tagging is unacceptable for non-packaged products, such as vegetables. Furthermore, grocery items can be refrigerated after some form of processing, such as cutting or peeling, or used only in part before being returned to the refrigerator. In such a situation, a tag originally attached to the grocery item must be replicated and reattached to each refrigerated portion. This is obviously unrealistic.

2.2 Design of grocery inventory system

In this study, we use grocery information that the system can obtain naturally during a refrigerator transaction. We discuss the use of the

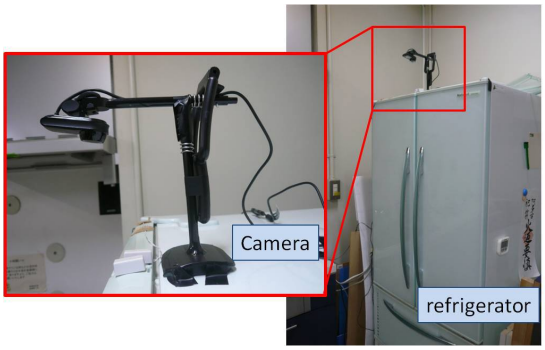

Figure 2. The camera set a refrigerator

Figure 3. The images captured by the camera. The groceries are occluded by user's hand and their visible portion is inconsistent.

appearance feature and the load feature. A camera attached to the refrigerator allows the grocery image to be captured, and we extract the appearance feature from the image; likewise, a weight scale on a refrigerator shelf allows the weight to be determined. These features are stable because they are inherent to the grocery. In this study, we suppose that grocery doesn't change their weight and position in the refrigerator during it is stored. We also suppose that grocery is stored in or taken out of the refrigerator one by one.

This information can be presented to the user directly. Figure 1 shows the possible user interface of a smartphone application, which displays the image and indicates the weight of each grocery item present in the refrigerator.

2.3 Possible features for re-identifying grocery

2.3.1 Appearance feature

Grocery re-identification in a refrigerator context is made very difficult by the complexity of the visual scene within a refrigerator. If these items were to be captured by a camera placed inside the refrigerator, many would appear occluded. In such a situation, it would be difficult to extract the appearance feature for each grocery item. In this study, the system captures grocery items during refrigerator transactions; it is possible to capture a grocery item individually with less occlusion when it is entering or leaving the refrigerator. We set a camera along the top of the refrigerator, as shown in Figure 2. Figure 3 shows the images captured by the camera. In order to extract the appearance feature, we can obtain the grocery region by background subtraction.

There is some related work on person re-identification using the appearance feature ([1]-[3]). Wang et al. [1] use a histogram of oriented gradients (HOG) and the shape context as appearance features. Bak et al. [2] use the gradient direction and the covariance of the gradient intensity. Zheng et al. [3] use a scale-invariant feature transform (SIFT) for each RGB channel in each local region and the normalized average RGB vector. Although these appearance features can be also used for grocery re-identification, we note that there are some differences between grocery re-identification and person re-identification. In the case of a person, we can minimize the effects of occlusion by using information aggregated over time. In the case of grocery items being inserted into or taken from a refrigerator, the user 's hand is always an occluding factor, as shown in Figure 3. Moreover, the visible portion of the grocery item is inconsistent between the entry and retrieval phases as shown in Figure 3.

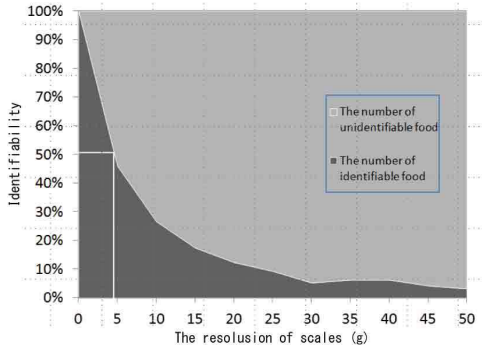

Figure 4. Resolution of weight scale and the ability of distinguishing groceries

2.3.2 Load feature

The load feature can provide an efficient way to re-identify grocery items under the assumption that the weight of a grocery item remains unchanged during refrigeration. In order to evaluate the suitability of the load feature, we performed a preliminary investigation.

Preliminary investigation: We surveyed the refrigerator of a five-member family. There were 98 grocery items in the refrigerator; we measured them using a digital weight scale of $1g$ resolution. We calculated the weight of each grocery item, and separately clustered them into each shelf; Table 1 shows these weights. As in the case of bean paste (type 1) and a grocery in a tupper (type 1) on shelf 4, and seaweed and mustard (type 2) on shelf 7, certain grocery items on a shelf weigh the same as others, within a $1g$ resolution. The maximum total weight on a shelf was about $5.1kg$.

We studied the relationship between the resolution of the scales and the identifiability of the 98 grocery items as shown in Figure 4. Figure 4 shows that it is impossible to identify half of the grocery items at a resolution of about $5g$. We conclude that the load feature is insufficient by itself.

We set a scale for each shelf in the refrigerator, and we obtain the weight feature from it. The grocery items on each shelf are arranged by weight, as shown in Table 1.

Table 1. The weight of groceries on each shelf

shelf 4	
grocery name	(g)
cheese (type 1)	18
bean paste (type 1)	**41**
a grocery in a tupper (type 1)	**41**
egg (type 1)	79
a grocery in a tupper (type 2)	93
seasoning (type 2)	135
seasoning (type 1)	155
a grocery in a nylon bag (type 1)	157
jelly (type 1)	196
jelly (type 2)	206
a grocery in a nylon bag (type 2)	227
bean paste (type 2)	290

@

shelf 7	
grocery name	(g)
green horseradish paste	13
seasoning (type 3)	16
Cooling Gel Sheet (type 1)	25
ginger	31
garlic	33
cheese (type 2)	34
Cooling Gel Sheet (type 2)	35
seaweed	**49**
mustard (type 2)	**49**
seasoned laver	59
mustard (type 2)	59
vanilla flavoring	76
coffee powder	82
condensed milk	104
broad bean chili paste	152
jelly (type 3)	212
sauce	242
kimchi	277
soy sauce	369
egg (type 2)	431

2.3.3 *Load balance feature*

Weight of grocery gives us good information for our grocery re-identification, but position of grocery on a shelf also gives good information. Schmidt et al [4] proposed the load sensing table in which four load sensors were installed at the four corners underneath a table. The load sensing table can calculate the weight and position of objects on it and detect some interactions, such as object placement and removal. Pei-yu Chi et al [5] uses this sensing design to track food calories during cooking.

In order to measure the load on a shelf, we also install four load sensors, one under each corner, allowing us to measure not only the total load but also the load balance on the shelf. The load balance also indicates the positions of the grocery items. Even if two groceries weigh the same at a certain resolution, the system can distinguish them by their position using the load balance feature. This provides richer information for grocery re-identification.

3 Grocery re-identification using load balance feature on a shelf

3.1 Load-sensing board

In this section, we describe our prototype load-sensing board. The design was based on the load sensing table [4]. Figure 5 depicts the load-sensing board. Four load sensors (Figure 6) are attached to the corners of a tempered glass sheet (size; $550mm \times 290mm \times 6mm$; weight, $2.5kg$, and load capacity, $5kg$). The load on each sensor is converted to digital form using an analog-to-digital converter as shown in Figure 7.

We used 'ASFORCE' load sensors [3] made by Asakusa-Giken Co., LTD[6] and the AGB65-ADC analog-to-digital converter [4]. Each AS-FORCE sensor is capable of measuring a maximum of 2.8 kgf, so the load sensing board can measure a maximum of 11.2kgf using the four sensors. The weight of the glass is $2.5kg$ and the sum of weight

[3] Its product is stopped by February 2012. http://www.robotsfx.com/robot/ForceSen.html

[4] http://www.robotsfx.com/robot/AGB65 ADC.html

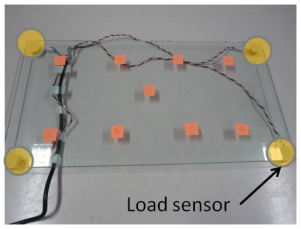

Figure 5. Load-sensing Board

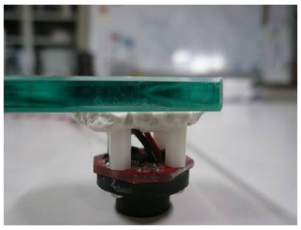

Figure 6. Load sensor (AS-FORCE)

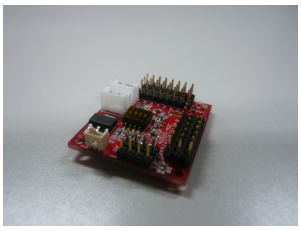

Figure 7. A/D converter (AGB65-ADC)

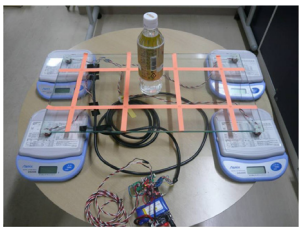

Figure 8. Calibration technique: we measure the sensor value and corresponding weight.

of grocery which is put on it is $5.1kg$. Thus, it is possible to measure grocery weight up to $7.6kg$. The 12-bit binary value obtained from the AGB65-ADC is sent to a computer via COM port at a baud rate of 9600 bps.

3.2 Algorithm for re-identification with load balance feature

Our system monitors the changes in the sensed values. Let $\boldsymbol{\Delta z} = (\Delta z_1, \Delta z_2, \Delta z_3, \Delta z_4)$ denote the changes in the values of the four sensors when grocery is placed inside (or taken out of) the refrigerator. $\boldsymbol{\Delta z}$ is the load balance feature and denotes the weight and position of the grocery for re-identification. The dissimilarity of the load balance feature is given by the distance between the two vectors $\boldsymbol{\Delta z}^{IN}$ and $-\boldsymbol{\Delta z}^{OUT}$, where $\boldsymbol{\Delta z}^{IN}$ denotes the load balance feature when a grocery item is put on the board and $\boldsymbol{\Delta z}^{OUT}$ denotes the load balance feature when a grocery item is taken off the board.

Suppose there are N groceries in the refrigerator. Let $\boldsymbol{\Delta z}_i^{IN} (i = 1, ..., N)$ be their load balance features. When a grocery is taken out and its load balance feature is $\boldsymbol{\Delta z}_k^{OUT}$, we re-identify the grocery with $argmin_i |\boldsymbol{\Delta z}_i^{IN} + \boldsymbol{\Delta z}_k^{OUT}|$.

3.3 Characteristic evaluation of load sensor

The value obtained from the load sensor is a digital form of the voltage output and does not directly express the load on the sensor. The relation between the sensed value and the actual load is expressed as a linear relation, $z = aw + b$, where z is the digital data and w is the corresponding load. The coefficients a and b depend on the individual sensor. We calculated a of each sensor as follows. First, we put digital weight scales under each sensor and loaded it with known weights. Next, as shown in Figure 8, we measure the sensor value z and corresponding weight w for variety of position and weight.

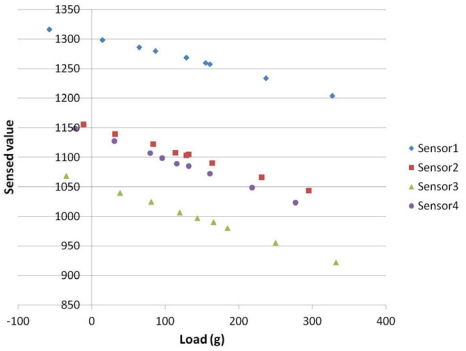

Figure 9. Sensed value and load

Figure 10. The grocery items used in the dataset 1.

Figure 11. The grocery items used in the dataset 2.

Figure 9 shows the result. The result shows the linear relation between load data and actual load. Furthermore, there are individual differences of gradient a between sensors. For measuring the weight of grocery, we convert the sensed value to load and then display the weight on a smartphone.

4 Experiment

We conducted experiments with two different item sets. The first item sets named dataset 1 consists of eight grocery items which are put on the load-sensing board in neat rows. The second item sets named dataset 2 consists of 15 grocery items which are put randomly, and some of them are stacked.

In 4.2, we describe grocery re-identification using the appearance feature. In 4.3, we describe grocery re-identification using the load balance feature: the method explained in section 3 to demonstrate the effectiveness of our method.

4.1 Settings

The grocery items used in the two datasets are shown in Figures 10 and 11. We set a camera along the top of the refrigerator, as shown in Figure 2. The system captures grocery items during refrigerator transactions. The load-sensing board is placed on a refrigerator shelf, and grocery items are placed on it, as shown in Figures 10 and 11. Then, the grocery items are removed in a different order.

Figure 12. An example of the image captured by the camera

Figure 13. The region image extracted from the image in Figure 12

4.2 Grocery re-identification using appearance feature

From the image captured by the camera, we extracted the grocery region manually. For that region, we calculated RGB color histogram. For the two datasets, we use the histogram as the appearance feature. Then, we calculated distances between groceries and conducted re-identification. Figure 12 shows an example of the image captured by the camera and Figure 13 shows the region image extracted from the image in Figure 12.

Table 2 shows the result of re-identification for the dataset 1 and Table 3 shows the result for the dataset 2. The top rows of the tables list the grocery items that are put in the refrigerator. The leftmost columns of the table list the grocery items that are taken out of the refrigerator. The middle of the table displays the Bhattacharyya distance between the features. The rightmost column displays the minima of the dissimilarities. The values on the diagonal cells, which should have minimum distance, are underlined. We searched for the nearest neighbor: searching for a inserted grocery item that is most similar to a removed grocery item. The nearest neighbor is marked with *.

For the dataset 1, all eight are successfully re-identified and thirteen of fifteen are successfully re-identified for the dataset 2. Table 3 shows that the removed lotus root was identified as the inserted potato. Their color are similar to each other, so they were incorrectly identified. They might be correctly re-identified if we add other feature to color histogram as the appearance feature. Table 3 also shows that the removed ham has more similar feature to the inserted natto than the inserted ham. Figure 13 shows that the image which was taken when ham was removed out of the refrigerator. Figure 14 and Figure 15 show that the images which were taken when the ham and the natto was inserted to the refrigerator. When the ham was inserted to the refrigerator, its one side appeared to the camera, but when it was removed, its the other side appeared. Thus, the ham has dissimilar appearance features when it was inserted and removed, and that is the reason why removed ham was mis-identified with the natto.

These results shows that appearance feature could be useful information to grocery re-identification and the accuracy of re-identification could be improved by using multiple features as the appearance feature. However, the camera doesn't necessarily capture the same visible portion of groceries all the time, so grocery re-identification using only appearance feature is hard even if adding multiple features to the appearance features..

Table 2. The result of re-identification using appearance feature for the dataset 1

IN / OUT	Sausage	Tupper	Tomato	Eggplant	Carrot	Navel orange	Butter	Lotus root	NN
Sausage	* 0.41	0.90	0.80	0.70	0.90	0.93	0.83	0.67	* 0.41
Tupper	0.90	* 0.48	0.79	0.92	0.94	0.97	0.97	0.93	* 0.48
Tomato	0.81	0.80	* 0.33	0.84	0.83	0.93	0.94	0.84	* 0.33
Eggplant	0.77	0.96	0.86	* 0.66	0.93	0.93	0.92	0.84	* 0.66
Carrot	0.88	0.95	0.86	0.92	* 0.41	0.70	0.94	0.89	* 0.41
Navel orange	0.90	0.97	0.92	0.93	0.88	* 0.78	0.93	0.85	* 0.78
Butter	0.85	0.97	0.95	0.93	0.94	0.96	* 0.49	0.90	* 0.49
Lotus root	0.66	0.94	0.84	0.81	0.89	0.89	0.87	* 0.47	* 0.47

Table 3. The result of re-identification using appearance feature for the dataset 2

IN / OUT	Sausage	Cabbage	Cucumber	Potato	Jelly	Tupper	Tomato	Eggplant	Natto	Carrot	Navel orange	Butter	Ham	Peanut Butter	Lotus root	NN
Sausage	* 0.45	0.81	0.80	0.56	0.74	0.90	0.84	0.79	0.76	0.91	0.93	0.76	0.70	0.75	0.82	* 0.45
Cabbage	0.85	* 0.65	0.73	0.85	0.92	0.97	0.95	0.93	0.88	0.93	0.96	0.91	0.91	0.89	0.91	* 0.65
Cucumber	0.79	0.79	*0.66	0.82	0.91	0.96	0.95	0.80	0.83	0.94	0.95	0.92	0.89	0.89	0.93	* 0.66
Potato	0.60	0.84	0.81	* 0.45	0.77	0.93	0.88	0.81	0.69	0.88	0.91	0.78	0.74	0.71	0.79	* 0.45
Jelly	0.81	0.93	0.88	0.81	* 0.57	0.73	0.66	0.87	0.73	0.90	0.93	0.93	0.62	0.68	0.91	* 0.57
Tupper	0.90	0.97	0.94	0.92	0.85	* 0.45	0.72	0.94	0.88	0.94	0.96	0.97	0.85	0.89	0.95	* 0.45
Tomato	0.85	0.96	0.91	0.87	0.62	0.63	* 0.50	0.91	0.84	0.91	0.95	0.96	0.73	0.78	0.92	* 0.50
Eggplant	0.65	0.91	0.75	0.76	0.88	0.95	0.91	* 0.41	0.81	0.94	0.94	0.92	0.83	0.86	0.94	* 0.41
Natto	0.82	0.90	0.82	0.77	0.79	0.92	0.87	0.81	* 0.43	0.87	0.90	0.89	0.80	0.71	0.90	* 0.43
Carrot	0.90	0.95	0.92	0.88	0.90	0.94	0.91	0.95	0.89	* 0.51	0.79	0.95	0.91	0.89	0.93	* 0.51
Navel orange	0.92	0.96	0.94	0.91	0.92	0.95	0.92	0.95	0.90	0.75	* 0.50	0.97	0.92	0.89	0.95	* 0.50
Butter	0.76	0.90	0.87	0.76	0.92	0.97	0.94	0.84	0.86	0.92	0.93	* 0.72	0.89	0.88	0.91	* 0.72
Ham	0.80	0.93	0.87	0.81	0.75	0.93	0.87	0.83	* 0.64	0.94	0.95	0.92	0.65	0.77	0.94	* 0.64
Peanut Butter	0.73	0.87	0.82	0.70	0.73	0.87	0.79	0.78	0.63	0.84	0.85	0.89	0.72	* 0.62	0.87	* 0.62
Lotus root	0.68	0.87	0.81	* 0.55	0.84	0.94	0.89	0.85	0.83	0.89	0.92	0.88	0.83	0.74	0.74	* 0.55

Figure 14. The region image extracted from the inserted ham image.

Figure 15. The region image extracted from the inserted natto image.

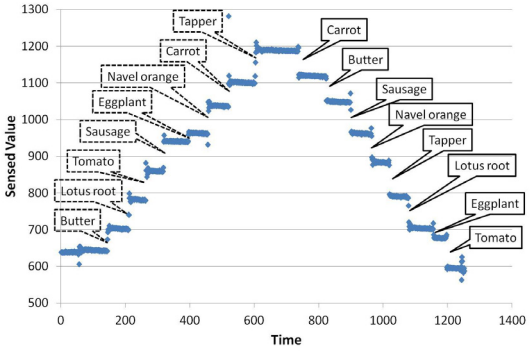

Figure 16. The changes in the sum of the four sensors

.

4.3 Grocery re-identification using load balance feature

Figure 16 shows the changes in the sum of the four sensors during refrigerator transactions for the datset 1. As seen in Figure 16, the load data varies drastically when a transaction occurs. In our experiments, the static part was extracted manually. We calculated the averages of the data in each static portion and used them for re-identification.

Tables 4 and 5 show the result of re-identification for the dataset 1 and Table 6 shows the result for the dataset 2. The results in Tables 4 and 6 used the load balance feature, while the result in Table 5 used the load feature. In Table 5, the load features are listed with grocery name in the top rows and the leftmost columns. The middle of the table displays the dissimilarities (here, the differences in the load balance vectors or the load) between the features.

For the dataset 1, all eight are successfully re-identified with the load balance feature while two are re-identified with the load feature. From these results, we conclude that the load balance feature gives more accurate results than the load feature. For the dataset 2, all fifteen were successfully re-identified with the load balance feature. However, the tapper and the navel orange have similar distance. The distance between the tapper taken out of and the tapper stored in is 4.63. The distance between the tapper taken out of and the navel orange stored in is 5.92. In this case, the tapper could be identified as the navel orange. This is because for two reasons below. They have similar weights: the tapper weighs $197g$ and the navel orange weighs $213g$. They were put on a similar position on the load-sensing board: the navel was stacked on the tapper. Thus, they have similar load balance features and they could be re-identified incorrectly.

Table 4. The result of re-identification using load balance feature for the dataset 1

IN / OUT	Sausage	Tupper	Tomato	Eggplant	Carrot	Navel orange	Butter	Lotus root	NN
Sausage	* 4.33	47.22	21.99	58.42	43.87	56.51	60.32	44.59	*4.33
Tupper	49.29	* 3.54	53.28	63.17	35.23	47.28	74.27	66.47	*3.54
Tomato	21.02	48.54	* 5.13	44.49	33.88	43.42	38.56	22.73	*5.13
Eggplant	53.95	57.28	41.59	* 3.37	26.38	27.56	32.75	38.58	*3.37
Carrot	41.93	28.18	35.77	33.50	* 4.52	19.69	46.53	42.57	*4.52
Navel orange	55.66	43.41	44.82	31.85	17.50	* 2.51	41.07	42.50	*2.51
Butter	54.68	69.15	35.51	33.49	40.90	36.73	* 6.90	15.19	*6.90
Lotus root	40.35	62.46	21.29	41.58	39.72	40.92	22.09	* 4.81	*4.81

Table 5. The result of re-identification using load feature for the dataset 1

(g) / OUT	IN	Sausage 208.61	Tupper 215.62	Tomato 211.67	Eggplant 65.65	Carrot 168.44	Navel orange 202.70	Butter 182.58	Lotus root 228.06	NN
Sausage	222.62	14.01	7.00	10.95	156.97	54.18	19.92	40.04	*5.44	*5.44
Tupper	229.90	21.28	14.28	18.22	164.25	61.45	27.20	47.32	* 1.83	*1.83
Tomato	223.51	14.90	7.90	11.84	157.86	55.07	20.81	40.93	*4.55	*4.55
Eggplant	77.63	130.98	137.98	134.04	* 11.98	90.81	125.07	104.95	150.43	*11.98
Carrot	182.65	25.97	32.97	29.03	117.00	14.20	20.05	* 0.07	45.41	*0.07
Navel orange	213.13	4.51	2.49	* 1.45	147.48	44.68	10.43	30.55	14.93	*1.45
Butter	208.91	* 0.29	6.71	2.77	143.26	40.46	6.21	26.33	19.16	*0.29
Lotus root	247.78	39.16	32.16	36.11	182.13	79.33	45.08	65.20	*19.72	*19.72

Table 6. The result of re-identification using load balance feature for the dataset 2

IN / OUT	Sausage	Cabbage	Cucumber	Potato	Jelly	Tupper	Tomato	Eggplant	Natto	Carrot	Navel orange	Butter	Ham	Peanut Butter	Lotus root	NN
Sausage	* 2.81	58.84	40.69	36.31	30.98	60.80	19.11	44.90	39.42	44.39	58.07	59.80	50.73	37.37	36.56	*2.81
Cabbage	59.51	* 1.59	80.00	87.52	67.53	106.53	71.15	86.42	89.81	86.32	105.82	97.64	96.73	88.74	71.40	*1.59
Cucumber	39.77	79.29	* 0.70	38.06	61.00	56.92	33.98	7.30	18.06	22.57	55.40	34.46	18.48	34.79	25.31	*0.70
Potato	34.86	87.53	37.94	* 0.51	36.70	25.68	17.82	37.63	29.49	26.00	22.68	39.45	36.61	5.14	29.61	*0.51
Jelly	31.44	67.34	61.79	36.74	* 3.67	50.23	29.18	65.03	58.88	54.91	48.65	68.64	68.80	40.37	46.63	*3.67
Tupper	58.85	106.59	57.69	25.16	50.37	*4.63	41.62	55.75	49.87	39.14	5.92	44.19	51.86	27.08	45.48	*4.63
Tomato	18.26	71.16	33.92	17.64	28.81	41.89	* 1.43	36.32	30.34	28.92	39.44	44.48	39.83	19.38	24.20	*1.43
Eggplant	44.62	86.42	6.72	38.04	64.71	55.14	37.06	* 1.36	15.38	21.45	53.58	31.59	11.67	34.32	28.55	*1.36
Natto	38.26	88.40	16.25	29.85	57.62	49.46	30.10	14.28	* 1.59	24.33	46.82	37.51	14.08	25.79	32.06	*1.59
Carrot	43.43	85.75	21.12	26.62	55.17	40.12	29.95	19.44	24.12	* 2.19	39.48	17.86	21.13	24.72	15.80	*2.19
Navel orange	57.26	106.24	56.31	23.25	49.53	4.39	40.19	54.39	47.72	38.85	* 2.21	44.97	50.37	24.80	45.50	*2.21
Butter	59.41	97.42	34.60	39.74	68.66	45.22	45.66	31.06	38.16	16.98	45.71	* 0.38	29.37	38.78	27.62	*0.38
Ham	49.74	96.29	17.98	37.22	68.13	52.27	40.24	11.53	13.57	22.80	50.32	30.06	* 1.52	33.40	34.56	*1.52
Peanut Butter	35.01	87.77	34.63	4.54	39.44	27.12	18.34	34.17	25.92	23.93	24.20	37.98	33.10	* 1.58	28.68	*1.58
Lotus root	35.97	71.90	25.12	29.39	47.15	45.66	25.14	27.28	32.45	15.03	45.28	27.12	33.32	29.38	* 0.88	*0.88

5 Conclusion

In this paper, we aim to re-identify grocery when it is put into or taken from a refrigerator in order to automatically monitor grocery inventory. We proposed using the load balance feature on a shelf in the refrigerator to reflect both the weight and the position of grocery items. We demonstrated that the system can re-identify groceries using the load balance feature by means of a prototype device and the two simulated experiments. The experimental results show that the load balance feature allows fairly accurate grocery re-identification when grocery items are put on the load-sensing board in neat rows. It also shows that some grocery items could be incorrectly identified as others even with the load feature, when one of the grocery items having similar weight to another is stacked on it. For future studies, we intend to investigate more complex situations encountered in real-world usage, and we will consider using appearance as an additional feature for grocery re-identification.

ACKNOWLEDGEMENTS

This work was partly supported by KAKENHI (23500137,70166099).

REFERENCES

[1] X. Wang, G. Doretto, T. Sebastian, J. Rittscher and P. Tu: “Shape and Appearance Context Modeling”, Proc. of ICCV, pp.1-8 (2007).
[2] S. Bak, E. Corvee, F. Bremond and M. Thonnat: “Person Re-identification Using Spatial Covariance Regions of Human Body Parts”, Proc. of AVSS, pp. 435-440 (2010)
[3] W.-S. Zheng, S. Gong and T. Xiang: “Associating Groups of People”, Proc. of BMVC (2009)
[4] Schmidt, M., Strohbach, K., Van Laerhoven, A., Friday, H.-W.G.: Context Acquisition based on Load Sensing. In: Proc. Ubicomp 2002: the 4th international conference on Ubiquitous Computing (2002)
[5] Pei-yu Chi, Jen-hao Chen, Hao-hua Chu, and Jin-ling Lo, “Enabling calorie-aware cooking in a smart kitchen,” Persuasive2008, Oulu, Finland, June 2008.
[6] The instruction of load sensing device AS-FORCE supplied by AsakusaGiken Co., LTD. (in Japanese). http://www.robotsfx.com/robot/ForceSen.html

Extracting Generic Cooking Adaptation Knowledge for the TAAABLE Case-Based Reasoning System

Emmanuelle Gaillard[123] and **Emmanuel Nauer**[123] and **Marie Lefevre**[4] and **Amélie Cordier**[4]

Abstract. This paper addresses the issue of interactive adaptation knowledge acquisition. It shows how the expert's involvement in this process can improve the quality and usefulness of the results. The approach is defended in the context of TAAABLE, a CBR system which adapts recipes to user needs. In TAAABLE, adaptation knowledge takes the form of substitutions. A datamining process allows the discovery of specific substitutions in recipes. A second process, that must be driven by an expert, is needed to generalise these substitutions to make them usable on other recipes. For that, we defend an approach based on closed itemsets (CIs) for extracting generic substitutions starting from specific ones. We focus on a restrictive selection of objects, on a specific filtering on the form of the CIs and on a specific ranking on support and stability of the CIs. Experimentations demonstrate the feasibility of our approach and show some first results.

Keywords: adaptation knowledge discovery, interactive knowledge acquisition, closed itemset, cooking.

1 Introduction

This paper addresses the issue of interactive adaptation knowledge (AK) discovery. It shows how experts can be involved in a datamining process in order to improve the overall results of a system. The approach is implemented within TAAABLE, a case-based reasoning (CBR) system which is a regular contestant of the *Computer Cooking Contest* (`http://computercookingcontest.net/`). This contest proposes to compare systems that are able to adapt cooking recipe to users constraints. For example, the user wants a pie recipe with raspberries. According to the user constraints, TAAABLE searches, in the recipe base (which is a case base), whether some recipes satisfy these constraints. Recipes, if they exist, are returned to the user; otherwise the system is able to retrieve similar recipes (i.e. recipes that match the target query partially) and to adapt these recipes, creating new ones. TAAABLE uses WIKITAAABLE, a semantic wiki in which the knowledge required by TAAABLE is stored.

[1] Université de Lorraine, LORIA, UMR 7503 — Vandœuvre-lès-Nancy, F-54506, France
[2] CNRS, LORIA, UMR 7503 — Vandœuvre-lès-Nancy, F-54506, France
[3] Inria — Villers-lès-Nancy, F-54602, France
[4] Université de Lyon, CNRS - Université Lyon 1, LIRIS, UMR 5205 - F-69622, Lyon, France

Currently, searching similar recipes and adapting them is only guided by the cooking ontology stored in WIKITAAABLE. AK is a type of knowledge a CBR systems may use to improve its results. For adapting recipes, an AK is considered as a substitution of ingredient(s) by other(s), in a given context. Two types of substitution are actually considered depending on the context in which the substitution can be applied:

- if the context of the substitution (e.g. "replace strawberries with raspberries") is a specific recipe (e.g. in the *"My Strawberry Pie"* recipe), then the AK is specific and can be reused for adapting only the specific recipe.
- if the context of the substitution (e.g. "replace butter with margarine") is a set of recipes (e.g. cake recipes), then the AK is generic and could be reused for adapting each of the recipes belonging to the set of recipes defining the context.

The main objective of this work is to define an approach to acquire generic AK. AK is required to produce fine-grained adapted recipes. In the current TAAABLE system, the adaptation process consists of two steps. First, a recipe similar to the query is retrieved. Then, the recipe is adapted to the specific constraints through a process of generalisation/specialisation. This process is arbitrary and does not take into account any specific AK. Therefore, a mid-term objective is to extend the current TAAABLE reasoning process by taking into account AK for computing better adaptation.

An approach for extracting generic ingredient substitutions based on similarity of cooking actions is studied in [17]. Our approach addresses the discover of generic AK from specific AK, using closed itemsets (CIs). This work is the continuity of a previous work on specific AK discovery [9], that has been integrated in WIKITAAABLE for improving the man-machine collaboration for acquiring AK [6].

This paper uses a classical knowledge discovery in database (KDD) process. Its originality lies in the special attention that has been given to the validation step. A dedicate interface has been built in order to interact with cooking experts for validating and adjusting AK proposed by the KDD process.

The paper is organised as follows. Section 2 introduces the context and motivation of this work. Section 3 explains our approach. Section 4 gives some results and evaluation, including a scenario for repairing generic AK. Section 5 describes the interface for acquiring AK and highlights the various features available for the expert. Section 6 discusses related work. Section 7 concludes the paper and discusses ongoing work.

2 Context and motivation

TAAABLE is a CBR system that has been designed for participating to the Computer Cooking Contest[5], an international contest which aims at comparing CBR system results on a common domain: cooking. Several challenges are proposed in this contest. Among them, two challenges (won by TAAABLE in 2010):

- the *main challenge*, asking CBR systems to return recipes satisfying a set of constraints given by the user, such as inclusion or rejection of ingredients, the type or the origin of the dish, the compatibility with some diets (vegetarian, nut-free, etc.). Systems have to search into a set of limited (approximately 1500) recipes for recipes satisfying the constraints, and if there is no recipe satisfying all the constraints, the systems have to adapt existing recipes into new ones.
- the *adaptation challenge*, asking CBR systems to adapt a given recipe to specific constraints. For example, "adapt the *My strawberry pie* recipe because I do not have strawberry".

An illustration of the TAAABLE interface is given in 1.

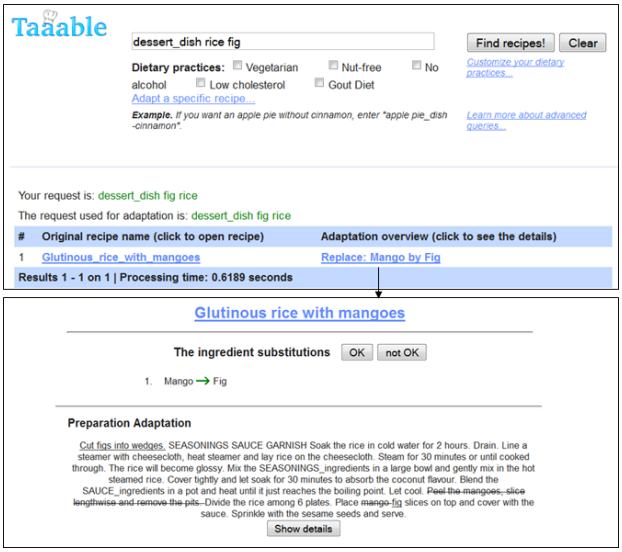

Figure 1. The TAAABLE interface. Queried for a dessert dish, with rice and fig, TAAABLE proposes to replace mango by fig in the "Glutinous rice with mangoes" recipe. After viewing the adapted recipe, the user can give feedback about the substitution ("OK" or "not OK").

2.1 TAAABLE principles

Like many CBR systems [16], TAAABLE uses an ontology to retrieve source cases that are the most similar to a target case (i.e. the query). TAAABLE retrieves and creates cooking recipes by adaptation. According to the user constraints, the system looks up, in the recipe base (which is a case base), whether some recipes satisfy these constraints. Recipes, if they exist, are returned to the user; otherwise the system is able to retrieve similar recipes (i.e. recipes that match the target query partially) and adapts these recipes, creating new ones. Searching similar recipes is guided by several ontologies, i.e. hierarchies of classes (ingredient hierarchy, dish type hierarchy, etc.), in order to relax constraints by generalising the user query. The goal is to find the most specific generalisation (with the minimal cost) for which recipes exist in the case base. Adaptation consists of substituting some ingredients of the source cases by the ones required by the user.

2.2 WikiTAAABLE

WIKITAAABLE is a semantic wiki that uses Semantic MediaWiki [11] as support for encoding knowledge associated to wiki pages. WIKITAAABLE contains the set of resources required by the TAAABLE reasoning system, in particular: an ontology of the domain of cooking, and recipes.

Figure 2. The Berry concept in WIKITAAABLE.

The cooking ontology is composed of 6 hierarchies: a *food* hierarchy (ingredients used in recipes, e.g. `Berry`, `Meat`, etc.), a *dish type* hierarchy (types of recipes, e.g. `PieDish`, `Salad`, etc.), a *dish moment* hierarchy (when to eat a dish, e.g. `Snack`, `Starter`, `Dessert`, etc.), a *location* hierarchy (origins of recipes, e.g. `France`, `Asia`, etc.), a *diet* hierarchy (food allowed or not for a specific diet, e.g `Vegetarian`, `NutFree`, etc.), an *action* hierarchy (cooking actions used for preparing ingredients, `toCut`, `toPeel`, etc.).

[5] `http://computercookingcontest.net/`

The set of recipes contained in WikiTaaable are those provided by the contest, that have been semantically annotated according to the domain ontology. Each recipe is encoded as a wiki page, composed of several sections: a title, which is the name of the recipe, an "*Ingredients*" section containing the list of ingredients used in the recipe, each ingredient being linked to its corresponding *Category* page in the food hierarchy, a "*Textual Preparation*" section describing the preparation process, some possible "*substitutions*" which are adaptation knowledge, and "*other information*" like the dish type, for example.

2.3 Adaptation knowledge

Improving the current results of Taaable could be done by acquiring *adaptation knowledge* (AK). In CBR systems, using AK is a classical approach for producing more fine-grained adaptations [15]. In the Taaable/WikiTaaable context, an AK is a substitution of some ingredients with other ones (e.g. in *"My Strawberry Pie"* recipe, `Strawberry` could be replaced with `Raspberry`). Formally, an adaptation knowledge is a 4-tuple $(context, replace, by, provenance)$, where:

- *context* represents the recipe or the class of recipes on which the substitution can be applied. An AK is specific if its *context* is a single recipe and generic if its *context* is a class of recipes (a specific type of dish, for example).
- *replace* and *with* are respectively the set of ingredients that must be replaced and the set of replacing ingredients.
- *provenance* is the source the AK comes from. Currently, four sources have been identified:

  1. Taaable, when AK results from a proposition of adaptation given by the reasoning process of Taaable.
  2. AK *extractor*, when AK results from a specific knowledge discovery system called "AK Extractor", which implements a KDD process also based on closed itemsets for discovering specific AK [9].
  3. *user*, when AK is given by a user by editing the wiki, as it is usually done in cooking web site, when users add comments about ingredient substitution on a recipe. See, for example, `http://en.wikibooks.org/wiki/Cookbook:substitutions`.
  4. *recipe*, when the AK is directly given by the original recipe when a choice between ingredients is mentioned (e.g. "100g butter or margarine"). This particular substitutions are taken into account by a wiki bot which runs through the wiki for automatically extracting them.

According to this definition, (*"My Strawberry Pie"*, *Strawberry*, *Raspberry*, Taaable), is an AK obtained from Taaable, meaning that strawberries can be replaced by raspberries in the *"My Strawberry Pie"* recipe. In WikiTaaable, each substitution is encoded as a wiki page like the one given in Figure 3.

In order to increase the acquisition of specific AK, a collaborative environment between users and automatic processes supported by machines has been implemented [6].

Figure 3. Example of a substitution page.

2.4 Objectives

The main objective of this work is to go beyond the discovery and the acquisition of specific AK, by discovering more generic AK. Generic AK are rules that could be applied in a larger number of situations, because the context of the AK will be a set of recipes (e.g. for cakes), instead being only one recipe (e.g. for the *"My Strawberry Pie"* recipe).

For that, an automatic KDD process can be used on cooking data and especially recipes. A crucial step of the KDD process [8] is that the results produced by the KDD process must be interpreted and validated by a (human) expert, in order to be considered as knowledge. The expert can also be involved for repairing some results that are not completely relevant. The next section details our KDD approach. In section 5, we show how the expert is involved in our approach.

Increasing the quantity and quality of AK in the wiki will improve the results of the reasoning system. However, this requires a smooth modification of the reasoning process for taking into account the AK for computing adaptation (this is a mid-term work). Besides, to ensure the non-regression of the system, the impact of the continuously new AK produced on the system results must be evaluated by test sets. In this paper, we focus on generic AK acquisition. A way for evaluating how knowledge evolution can be managed for improving the results of a reasoning process is proposed in [19].

3 Methodology for adaptation Knowledge discovery

AK discovery is based on the scheme of KDD [8]. The main steps of the KDD process are data preparation, datamining, and interpretation of the extracted units of information. Data preparation relies on formatting data for being used by datamining tools and on filtering operations for focusing on special subsets of objects and/or properties, according to the

objectives of KDD. Datamining tools are applied for extracting regularities into the data. These regularities have then to be interpreted; filtering operations may also be performed on this step because of the (often) huge size of the datamining results or of the noise included in these results. All the steps are managed by a computer science analyst guided by an expert of the domain.

The KDD process used in this paper is based on itemset extraction, which is introduced in 3.1. The preparation of data that will be used as entry of the itemset extraction is presented in 3.2 and the way for obtaining generic AK from itemset is detailed in 3.3. Experimentations and results are presented and discussed in 4.

3.1 Itemset extraction

Itemset extraction is a set of datamining methods for extracting regularities into data, by aggregating object items appearing together. Like FCA [10], itemset extraction algorithms start from a *formal context* K, defined by $K = (O, A, R)$, where O is a set of objects, A is a set of attributes, and R is the relation on $O \times A$ stating that an object is described by an attribute [10]. Table 1 shows an example of context, in which recipes are described by the ingredients they require: O is a set of 5 objects (r_1, r_2, r_3, r_4, and r_5) which are recipes, A is a set of 12 attributes (`Sugar`, `Water`, `Strawberry`, etc.) which are ingredients. A *cross* at the intersection of a recipe r (in line) and an ingredient i (in column) means that r requires i.

	Sugar	Water	Strawberry	PieCrust	Cornstarch	CoolWhip	Raspberry	Gelatin	Apple	AppleJuice	Cinnamon	PieShell
r_1	×	×	×	×	×	×						
r_2	×			×	×		×	×				
r_3	×	×		×			×					
r_4	×			×	×				×	×		
r_5	×	×							×		×	×

Table 1. Example of formal context representing ingredients used in recipes.

An *itemset* I is a set of attributes, and the *support* of I, *supp*(I), is the number of objects of the formal context having every item of I. I is *frequent*, with respect to a threshold σ, whenever $supp(I) \geq \sigma$. I is *closed* if it has no proper superset J ($I \subsetneq J$) with the same support. For example, {`Sugar`, `Raspberry`} is an itemset and $supp(\{\texttt{Sugar}, \texttt{Raspberry}\}) = 2$ because 2 recipes require both `Sugar` and `Raspberry`. However, {`Sugar`, `Raspberry`} is not a closed itemset, because {`Sugar`, `PieCrust`, `Raspberry`} has the same support. In the following, "CIs" stands for closed itemsets.For $\sigma = 3$, the frequent CIs of this context are {`Sugar`, `PieCrust`, `Cornstarch`}, {`Sugar`, `PieCrust`}, {`Sugar`, `Water`}, {`Sugar`} {`Water`} {`PieCrust`}, and {`Cornstarch`}.

The *stability* of I, *stab(I)*, is the probability that I endures despite the absence of an object of the formal context having every item of I [12]. In other words, a stable CI does not result from the existence of some particular objects. Formally, let X be a set of objects and X' the maximal set of properties describing X, with $X' = \{a \in A | \forall o \in X, oRa\}$. Let n be the cardinal of X. Let $\langle C \rangle_j$ be the equivalence class of X corresponding to the set of objects of size j included in X and having the same maximal set of properties that X.

Kuznetsov define the stability as [12]:

$$stab(X') = \frac{|\bigcup_{j=2}^{n-1} \langle C \rangle_j|}{2^n - n - 2}$$

Example. Let $X_1 = \{r_1, r_2, r_4\}$, $X_1' = \{$`Sugar`, `PieCrust`, `Cornstarch`$\}$ and $n = 3$. $\langle X_1 \rangle_2 = (\{r_1, r_2\}, \{r_1, r_4\}, \{r_2, r_4\})$

$$stab(X_1') = \frac{|\bigcup_{j=2}^{n-1} \langle X_1' \rangle_j|}{2^n - n - 2}$$

$$stab(X_1') = \frac{3}{2^3 - 3 - 2}$$

$$stab(X_1') = 1$$

Let $X_2 = \{r_1, r_2, r_3, r_4\}$, $X_2' = \{$`Sugar`, `PieCrust`$\}$ and $n = 4$. $\langle X_2 \rangle_2 = (\{r_3, r_4\})$ $\langle X_2 \rangle_3 = (\{r_1, r_2, r_3\}, \{r_1, r_3, r_4\}, \{r_2, r_3, r_4\})$

$$stab(X_2') = \frac{|\bigcup_{j=2}^{n-1} \langle C \rangle_j|}{2^n - n - 2}$$

$$stab(X_2') = \frac{1 + 3}{2^4 - 4 - 2}$$

$$stab(X_2') = 0.4$$

As $stab(\{\texttt{Sugar}, \texttt{PieCrust}, \texttt{Cornstarch}\}) = 1$ and $stab(\{\texttt{Sugar}, \texttt{PieCrust}\}) = 0.4$, {`Sugar`, `PieCrust`, `Cornstarch`} is more stable than {`Sugar`, `PieCrust`}. This means that the existence of {`Sugar`, `PieCrust`} is more dependant of its objects than {`Sugar`, `PieCrust`, `Cornstarch`}. If one of the objects r_1, r_2, r_4 is removed from the formal context, X_1 CI will always exist. This is not the case for X_2. If r_3 is removed from the formal context, X_2 will not continue to be a CI.

A CI I is said *stable*, with respect to a threshold γ, whenever $stab(I) \geq \gamma$. In the previous example, with $\gamma = 0.75$, {`Sugar`, `PieCrust`, `Cornstarch`} is stable, whereas {`Sugar`, `PieCrust`} is not stable.

For the following experiments, the CHARM algorithm [21] that efficiently computes the CIs is used. CHARM is implemented in CORON, a software platform implementing a rich set of algorithmic methods for symbolic data mining [20].

3.2 Data preparation for generic AK discovering

The first step when using a closed itemsets mining algorithm is building the formal context. In order to extract generic AK, specific AK, such as the ones stored in WIKITAAABLE, will be used.

Building the formal context is guided by the set of recipes for which generic AK want to be extracted (e.g. for cakes). The characterisation of the set of recipes is the context parameter of a substitution, representing in which case the substitution can be applied. In the following and for simplification, we choose to characterise a set of recipes only by a dish type T belonging to the hierarchy of dish types stored in WIKITAAABLE. But the KDD process described in this section

could also be run on a set of recipes characterised more precisely (e.g. for cakes containing apples).

Each specific AK, as the one presented in Figure 3 will be used for producing an object in the formal context. The set of properties will be composed of the ingredients that are replaced and the ingredients they are replaced with, extended by their most generic concepts in the food hierarchy of WikiTaaable. In the formal context, replaced ingredient(s) and their generics are prefixed by `R` and replacing ingredient(s) and their generics are prefixed by `W`. Moreover, the replaced (respectively replacing) ingredient(s) of a substitution is/are duplicated and prefixed by `R_ING` (respectively `W_ING`) for distinguish it/them from their generic concepts. The idea, when prefixing the properties of the formal context like this, is to facilitate the interpretation of the CIs (see hereafter).

For example, let S_1, S_2 and S_3 be 3 specific AK substitutions for cakes, with S_1 and S_2 consisting in replacing `Strawberry` with `Apple`, and S_3 consisting in replacing `Raspberry` with `Pear`. According to the food hierarchy which states that the generic concepts of `Strawberry` are `Fruit` and `Berry`, that the generic concepts of `Apple` are `Fruit` and `PomeFruit`, that the generic concepts of `Raspberry` are `Fruit` and `Berry`, and that the generic concepts of `Pear` are `Fruit` and `PomeFruit`, the formal context presented in Table 2 is built.

	R_ING_Strawberry	W_ING_Apple	R_Strawberry	W_Apple	R_Berry	R_Fruit	W_PomeFruit	W_Fruit	R_ING_Raspberry	W_ING_Pear	R_Raspberry	W_Pear
S_1	×	×	×	×	×	×	×	×				
S_2	×	×	×	×	×	×	×	×				
S_3					×	×	×	×	×	×	×	×

Table 2. Formal context for S_1, S_2 and S_3 substitutions.

For extracting knowledge for a given dish type T, the formal context $K = (O, A, R)$ will be composed as follow:

- O is the set of substitutions in recipes producing a dish of type T;
- A is the set of replaced ingredient(s), the replacing ingredient(s) and their generic concepts, prefixed with the role (`R`, `W`, `R_ING`, `W_ING`) they play in the substitution;
- R is the relation stating that the ingredient is involved in the substitution.

For obtaining concrete results, a large number of specific AK is required for the datamining process in order to extract generic AK. So, specific AK have been taken from the *Recipe Source* database (`http://www.recipesource.com`) from ingredient lines containing a choice between ingredients. For example, a recipe containing an ingredient line such as *"500g of strawberry or apple"* is equivalent to S_1, meaning that strawberry can be replaced by apple. However, a choice $\texttt{Ing}_1$ or $\texttt{Ing}_2$ represents in fact two substitutions: $\texttt{Ing}_1$ can be replaced by $\texttt{Ing}_2$ and conversely. That is why, a choice of ingredients will produce two lines in the formal context, one for each possible substitutions.

3.3 From closed itemsets to generic AK

The formal context allows to build CIs which have to be transformed and interpreted in order to acquire generic AK. A formal context as the one given in example in Table 2 will generate CIs with prefixed ingredients, grouping together ingredients or category of ingredients depending on their regularities of co-occurrence. Figure 4 shows the first lines of the raw results resulting from the datamining process applied on specific AK that can be applied in cake dishes, ranked by their support, and organised hierarchically according to their itemset inclusion (as it is done in concept lattices).

Two kinds of CIs can be distinguished depending if the CI contains ingredient prefixed by `R_ING`, `W_ING` or not:

- a CI composed only from ingredients which are not prefixed by `R_ING` or `W_ING` is the expression of a generalised substitution rules. The ingredients of the specific AK used for building the formal context have been generalised to some more generic class of ingredient. For example, the CI {`R_Berry`, `R_Fruit`, `W_PomeFruit`, `W_Fruit`} will be produced from Table 2. It can be interpreted as follow: berry can be replaced by pome fruit in cake dishes (starting from strawberry and raspberry replaced respectively by apple and pear). In the following, this kind of substitution will be referred as a *generalised* AK.
- a CI composed with ingredients prefixed by `R_ING` or `W_ING` is the expression of a part of substitution that really exists in specific AK. For example, the CI {`R_ING_Strawberry`, `W_ING_Apple`, `R_Strawberry`, `W_Apple`, `R_Berry`, `R_Fruit`, `W_PomeFruit`, `W_Fruit`} will be produced from Table 2. When keeping only the most specific `R_ING` and `W_ING`, it can be interpreted as follows: strawberry can be replaced by apple in cake dishes. In the following this kind of substitution will be referred as *instantiated* AK.

3.3.1 CI *interpretation*

For extracting substitution AK, a CI must contain a replaced part (prefixed by `R_`) and a replacing part (prefixed by `W_`). So CIs with only ingredients prefixed by `R_` or only ingredients prefixed by `W_` will not be interpreted.

Some other simplifications are required to transform a CI into an AK. First, only the most specific ingredients prefixed by `R_` and the most specific ingredients prefixed by `W_` are kept. As illustrated previously, {`R_Berry`, `R_Fruit`, `W_PomeFruit`, `W_Fruit`} will be simplified in {`R_Berry`, `W_PomeFruit`}, from which a substitution AK can be generated. Second, if a CI contains only two items related to the same ingredient, one prefixed by `R_` and one prefixed by `W_`, then the CI will not produce a useful AK. For example, {`R_Fruit`, `W_Fruit`} will generate an AK stating that fruit can be replaced by fruit, which is not really interesting.

Depending from how the ingredient are prefixed in a CI, different kinds of AK will be generated:

- `R_ING_x`, `W_ING_y` has to be interpreted like *"Replace the ingredient x by the ingredient y"*;
- `R_ING_x`, `W_y` has to be interpreted like *"Replace the ingredient x by ingredients of the class y"*;
- `R_x`, `W_ING_y` has to be interpreted like *"Replace ingredients of the class x by the ingredient y"*;

```
(1) R_Fat, W_Fat ; Stab :1; Supp : 1510
(2)    R_Fat, R_Butter, W_Fat ; Stab :0.99; Supp : 757
(3)    R_Fat, R_Butter, R_ING_butter, W_Fat ; Stab :0.99; Supp : 740
(4)     R_fat, R_Butter, R_ING_Butter, W_Fat, W_Margarine, W_ING_Margarine ; Stab :1; Supp : 699
(5)        R_fat, R_Butter, R_ING_Butter, W_Fat, W_Shortening ; Stab :0.49; Supp : 28
(6)            R_fat, R_Butter, R_ING_Butter, W_Fat, W_Shortening, W_ING_Shortening ; Stab :0.99; Supp : 27
(7)            R_fat, R_Butter, R_ING_Butter, W_Fat, W_Shortening, W_Crisco, W_ING_Crisco ; Stab :0.5; Supp : 1
...
(17)          R_Fat, W_Fat, W_Butter ; Stab :0.99; Supp : 757
(18)              R_Fat, W_Fat, W_Butter, W_ING_Butter ; Stab :0.99; Supp : 740
(19)                  R_Fat, W_Fat, W_Butter, W_ING_Butter, R_Margarine, R_ING_Margarine ; Stab :1; Supp : 699
...
```

Figure 4. First lines of the raw output resulting from the datamining process.

- `R_x, W_y` has to be interpreted like *"Replace ingredients of the class x by ingredients of the class y"*.

Each CI will be presented in the interface with replaced ingredients first (ingredients prefixed by `R_`) followed by replacing ingredients (ingredients prefixed by `W_`). Moreover, if two CIs I_1 and I_2 exist, such as $I_1 = \{\texttt{R\_x}, \texttt{W\_y}\}$ and $I_2 = \{\texttt{R\_y}, \texttt{W\_x}\}$, only the first one will be kept for proposing an AK involving `x` and `y` in the interface. A symmetric arrow between `x` and `y` will represent that `x` can be replaced with `y`, and conversely, `y` can be replaced with `x`.

After simplification, only lines (2), (3), (4), (6), (7) of Figure 4 are kept, the other lines being removed by one of the simplification rules given before.

3.3.2 CI *filtering and ranking*

As the number of CIs generated by the datamining process is (often) huge, some rules are required in order to limit the number of CIs that will be presented to the expert for evaluation. Usually, CIs are filtered thanks to their support and stability: CIs with support lower than a threshold σ or with a stability lower than a threshold γ could be removed. Support and stability are also generally used for ranking the CIs. In our approach, the expert sets support and stability values through the interface (which is presented in section 5).

4 Experimentations and results

In this work, some experimentations have been realised for showing first results and for discussing about the best way of validating AK coming from CIs. In all the tables of results presented in this section, the AK propositions are symmetric.

A first experiment shows, on an example, how many AK are generated starting from a set of specific AK. For *cake* dishes, the formal context contains 2556 objects described by 978 properties, and produces 1599 CIs. Among them, 321 CIs do not contain at the same time ingredients prefixed by `R_` and ingredients prefixed by `W_`, and can be removed. After merging two symmetric AK into one, 571 CIs still remain. 52 CIs composed from the same ingredient prefixed once by `R_` and then by `W_` are then removed. With a minimal support $\sigma = 3$ and a minimal stability $\gamma = 0.5$, only 131 CIs are kept, for $\gamma = 0.6$, only 113, for $\gamma = 0.7$, only 97, for $\gamma = 0.8$, only 81, for $\gamma = 0.9$, only 51, for $\gamma = 0.95$, only 37. So, increasing the stability (as well as the minimal support) is a way to limit the number of CIs that will be proposed to the expert. Unfortunately, there is no good heuristic to define automatically these thresholds. That is why these parameters may be modified in the interface. Moreover, ranking these AK by support or by stability provides results that are different, as shown in Table 3 and Table 4. Deciding between the support and the stability, which one could facilitate the choice of *good* CIs, is not easy. However, using the stability for ranking produces more instantiated AK (i.e. AK. involving ingredients prefixed by `R_ING` and `W_ING`).

Replaced	With	Supp	Stab
Butter	Fat	758	1.00
ING_Butter	Fat	741	1.00
Butter	ING_Margarine	716	1.00
ING_Butter	ING_Margarine	699	1.00
Cultured milk product	Dairy	92	1.00
ING_Pecan	Nut	55	1.00
Nut	Walnut	49	0.72
ING_Butter milk	Dairy	47	0.98
Nut	ING_Walnut	47	0.87
ING_Pecan	ING_Walnut	44	1.00
ING_Butter milk	Cultured milk product	41	0.97
Fruit	Citrus fruit	40	1.00
ING_Butter milk	ING_Sour milk	36	1.00
Coffee	Liquid	36	1.00
ING_Butter	ING_Shortening	27	1.00
Fruit	Orange	23	0.98
Fat	Dairy	21	0.99
Fruit	Berry	20	0.98
Dairy	Milk	20	0.98
Citrus fruit	Orange	17	1.00

Table 3. The 20 first AK propositions for cake dishes, ranked by decreasing support, with $\gamma = 0.7$.

Table 3 gives the 20 first AK propositions for cake dishes, ranked by decreasing support, with $\gamma = 0.7$. The support fosters the emergence of generalised AK propositions. Some of the generalised AK propositions are too generic, e.g. `Fruit/Citrus fruit`, `Fat/Dairy`, `Coffee/Liquid`; they will not produce a relevant AK. For example, `Fat/Dairy` will not produce a relevant AK because not all the subconcepts of `Fat` can be replaced with anyone of the subconcepts of `Dairy` (for example, `Oil` cannot be replaced with `Mozzarella`). However, some symmetric generic AK propositions could be relevant, but not in a symmetric way. For example, for `Fat` $\rightarrow$ `Butter` seems a relevant AK (`Oil`, `Shortening` can be replaced with `Butter`), but the opposite is not relevant (`Butter` cannot be replaced in general with `Oil`, when melted with sugar, for example). Instantiated AK seems more relevant regarding a dish type, e.g. `ING_Butter/ING_Margarine`, `ING_Walnut/ING_Pecan`. However some of instantiated AK could not be applied to the type of dish for which the

Replaced	With	Supp	Stab
Butter	Fat	758	1.00
ING_Butter	Fat	741	1.00
Butter	ING_Margarine	716	1.00
ING_Butter	ING_Margarine	699	1.00
Cultured milk product	Dairy	92	1.00
ING_Pecan	Nut	55	1.00
ING_Pecan	ING_Walnut	44	1.00
Fruit	Citrus fruit	40	1.00
ING_Butter milk	ING_Sour milk	36	1.00
Coffee	Liquid	36	1.00
ING_Butter	ING_Shortening	27	1.00
Citrus fruit	Orange	17	1.00
ING_Margarine	ING_Unsalted butter	15	1.00
ING_Milk	ING_Water	9	1.00
ING_Egg	ING_Egg substitute	8	1.00
ING_Coffee	ING_Water	8	1.00
Fat	Dairy	21	0.99
ING_Espresso	Coffee	14	0.99
Fruit	Stone fruit	13	0.99
ING_Butter milk	Dairy	47	0.98

Table 4. The 20 first AK propositions for cake dishes with best stability.

KDD process has been run. For example, `ING_Sour milk` and `ING_Butter` are exchangeable in `Cheesecake` but not in all type of `cake`s.

Table 4 gives the 20 first AK propositions for cake dishes with the best stability. The same kind of analysis can be done for Table 3. The major difference is the number of instantiated AK propositions which is the double with a stability ranking, comparing to the support ranking. As instantiated AK are, in proportion, more relevant than generalised AK, a way to obtain a good number of generic AK is to filter instantiated AK, as presented in Table 5. In this table, CIs are filtered for only producing instantiated AK.

Replaced	With	Supp	Stab
ING_Butter	ING_Margarine	699	1.00
ING_Pecan	ING_Walnut	44	1.00
ING_Butter milk	ING_Sour milk	36	1.00
ING_Butter	ING_Shortening	27	1.00
ING_Margarine	ING_Unsalted butter	15	1.00
ING_Milk	ING_Water	9	1.00
ING_Egg	ING_Egg substitute	8	1.00
ING_Coffee	ING_Water	8	1.00
ING_Brandy	ING_Cognac	6	0.98
ING_Espresso	ING_Strong coffee	6	0.98
ING_Butter milk	ING_Milk	5	0.97
ING_Armagnac	ING_Cognac	5	0.97
ING_Coffee	ING_Espresso	5	0.97
ING_Butter	ING_Flour	5	0.97
ING_Cake mix	ING_Chocolate	5	0.97
ING_Brandy	ING_Rum	5	0.97
ING_Butter	ING_Sour milk	5	0.97
ING_Cream	ING_Evaporated milk	4	0.94
ING_Honey	ING_Maple syrup	4	0.94
ING_Apple	ING_Orange juice	4	0.94
ING_Cream	ING_Milk	4	0.94
ING_Almond	ING_Pecan	4	0.94
ING_Lemon rind	ING_Orange	4	0.94
ING_Cream cheese	ING_Mascarpone	4	0.94

Table 5. All instantiated AK propositions ranked by decreasing support with $sigma = 4$ and $\gamma = 0.7$.

Many experiments, on various dish types, produce interesting results. In many dish types, butter can be replaced with margarine (and vice versa), which is the most frequent substitution in cooking. Some other interesting AK can be mentioned for illustration, like, for example:

- in `Beverage`: `ING_Brandy` → `ING_Liquor`, `ING_Honey` → `ING_Sugar`, `ING_Lemon juice` → `ING_Lime juice`;
- in `Dinner pie`: `ING_Cream` → `ING_Milk`, `ING_Plain yogurt` → `ING_Sour cream`, `ING_Basil` → `ING_Thyme`;
- in `Salad`: `ING_Mayonnaise` → `ING_Salad dressing`, `ING_Lemon juice` → `ING_Vinegar`, `ING_Soy sauce` → `ING_Tamari`
- in `Rice dish`: `ING_Olive oil` → `ING_Vegetable oil`, `ING_Chicken` → `ING_Ham`,`ING_Cheddar` → `ING_Monterey jack`
- in `Soup`: `ING_Leak` → `ING_Onion`, `ING_Chicken stock` → `ING_Vegetable stock`

5 Dialogue with the expert to transform CI to generic AK

The process of CI extraction, filtering and ranking allows generating AK propositions. The number of AK propositions is huge. Besides, AK have different forms (general or specific AK).

To be exploited in the recipe adaptation process, AK propositions must be validated by an expert. However, the AK result from the extraction process can not be proposed to the expert as obtained at the output of the process (see Figure 4). To facilitate the work of the expert, we will offer an interface. This interface, presented in Figure 5, allows experts to validate or to correct AK propositions. It is composed of three parts.

The first part, on the top left hand side, allows choosing parameters for filtering AK. Parameters are:

- *Type* which corresponds to the type of dish for which AK propositions are extracted. This parameter also determines the context in which the AK could be applied;
- *Min supp* which is minimal support required for a CI;
- *Min stab* which is the minimal stability required for a CI;
- Maximal *Number* of AK propositions displayed;
- Ranking method: by decreasing stability or decreasing support;
- Type of AK: generalised and/or instantiated AK.

These parameters can be chosen by experts to define the scope of the rules they want to work on.

The second part, on the bottom left hand side, displays AK propositions satisfying the filtering parameters. Each line is of the form ``` ``Food-A Symbol Food-B'' ```, where `Food-A` and `Food-B` are lists of ingredients, and `Symbol` can be a right-facing arrow (→) indicating that `Food-A` can be replaced by `Food-B`, a left-facing arrow (←) to reverse the proposed AK and thus replace `Food-B` by `Food-A`, and finally, a symmetric arrow (↔) indicating that the rule is symmetric (`Food-A` can be replaced by `Food-B` and `Food-B` can be replaced by `Food-A`). In the illustration given in Figure 5, all AK are symmetric.

On the right of each AK, three actions are possible:

- validate the AK which will be stored in WikiTaaable;
- reject the proposed AK;
- display more details about the AK.

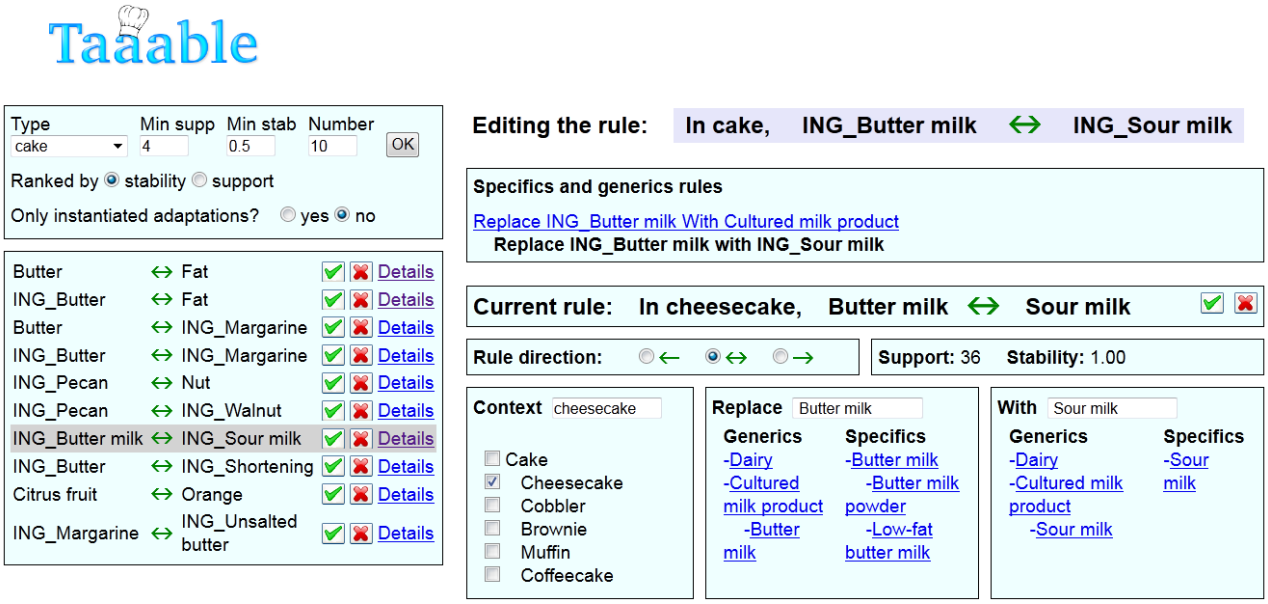

Figure 5. Expert validation interface.

The details of a AK are displayed on the third part of the interface, on the right-hand side. On this frame, the expert can make modifications in order to adapt the proposed AK.

The first line shows the proposed AK. The first frame allows the expert seeing the *"closed rules"* of the proposed AK and may navigate in the direct generics and direct specifics AK. If the expert select another AK in this list, the selected AK becomes the AK to validate and the interface is refreshed.

The second frame shows the AK which evolves according to the actions of the expert on the following frames. The *"Rule direction"* frame allows expert to modify the direction of the AK. The frame at its right indicates the support and the stability of the AK.

The *"Context"* frame allows changing the selected dish type by one or several more specific dish types in the ontology. This modification is necessary when the rule is too general and does not apply to an entire category of dish.

For example, we saw that the AK `ING_Sour milk` ↔ `ING_Butter milk` proposed for any cake cannot be validated as such because, actually, it cannot be applied to all types of cake. The expert can specify more specific categories of dishes for which the proposed AK is true, in this example, for `Cheesecake`.

The *"Replace"* and *"With"* frames allow changing the ingredients involved in the AK. This functionality is required in the case of a too general rule for involving more specific ingredients instead of too general category, or conversely. The expert can choose one of the generic or specific ingredients in the ontology.

For example, in the `Biscuit` category, the rule `ING_Margarine` → `Fat` cannot be validated because in the ingredients ontology of Taaable, `Fat` is more generic than `Bacon grease`, `Butter`, `Dripping`, `Duck fat`, `Lard`, `Margarine`, etc. Therefore, this AK indicates that margarine may be substituted by any of these ingredients or category of ingredients, which is not relevant.

After modifications, the expert can validate the adaptation rule or reject the proposed rule.

On Figure 5, the expert has selected the applicative context `cake`, *Min supp* $\sigma = 4$, *Min stab* $\gamma = 0.7$, and 10 rules to display and a ranking by stability. Then, the expert has clicked on *Details* for the rule `ING_Butter milk` ↔ `ING_Sour milk`. On the right, he has changed the applicative context `Cake` by `Cheescake`. He can now validate or reject this rule proposition.

6 Related work

Previous researches deal with man-machine collaboration where knowledge is obtained by a knowledge extraction process which is guided or/and validated by human. In this section, we present several systems based on KDD or on CBR or even both that demonstrate this collaboration.

6.1 KDD systems

The KDD process requires to be supervised by an expert, who can interact at various levels. One of the most usual problems in a KDD is to control the over-abundance of results generated by the KDD process. The expert could, for example, interact for better selecting the data that will be mined as it is described in [3] which proposes an approach for optimising the formulation of the problem to solve. Another approach consists in filtering and ranking of numerous results obtained by the datamining algorithms. For example,

[18] proposes subjective measures of interestingness for evaluating the datamining results. These measures depend on the user profile and a result is considered as interesting for a user according to two major reasons: unexpectedness (if the user is "surprised" by the results) and actionability (if the user can exploit the results). The paper focuses on unexpected results which are results in contradiction with beliefs of the user. So, a result which may revise beliefs of a user is relevant.

Another usual problem of the KDD process is the selection of relevant information among the large set of information produced, for transforming them into knowledge. Many approaches for taking into account this step of the KDD process have been proposed. For example, [2] presents a methodology for KDD in the context of building a semi-automatic ontology from heterogeneous textual resources. [2] uses formal concept analysis (FCA) [10] for classifying objects according to their properties which are extracted from various textual resources (e.g. thesaurus, full texts, dictionaries, etc.). The results of the FCA process is translated in description logics for representing the ontology concepts. Experts are involved at each step of the KDD process. For example, when the properties describing the objects are produced by an automatic extraction process, experts have to validate and filter the most representative properties, i.e. that described the best the objects. During the last step of the process, experts have to validate the formal concepts that have been produced, by selecting those which makes sense in their domain.

The integration of the AK Extractor follows the same principle. AK Extractor implements a KDD process which produces a set of substitution propositions. These substitutions have to be validated in order to be stored as AK.

6.2 CBR systems

CBR [16] is a method for solving new problems thanks to adaptation of previously solved problems. However, the step of adaptation may fail.

In this case, an expert must take part in the repair process. In the following, we present systems in which the expert is involved in an opportunistic way to repair solutions. Most of these systems take advantage of this opportunity to acquire new adaptation knowledge.

DIAL [13] focuses on an interactive acquisition of AK in the domain of disaster response planning. When the system returns a solution that is inconsistent, the response planning is returned with a description of the elements that need to be adjusted in the planning. For example, a response planning for an earthquake in Los Angeles indicates that National Guard must be called. When this plan is used for an earthquake in Indonesia, a problem arises because there is not National Guard in Indonesia. So, the response plan must be adapted. DIAL is composed of three kinds of adaptation process. Case-based adaptation and rule-based adaptation reuse knowledge already available in the system. However, manual adaptation involves the expert. In this third type of adaptation, the expert selects a generic transformation to apply and navigate in the knowledge base to search relevant knowledge for instantiating the generic transformation. It is an opportunistically triggered man-machine collaboration for AK acquisition.

In DIAL, adaptation is performed manually transformed if automatic adaptations fail. Conversely, WebAdapt [14] proposes two independent adaptations modes: one automatic and one manual. Thus, WebAdapt allows users to choose the way they want their solutions to be adapted depending on their own needs. This is useful when users know the results they want to achieve. WebAdapt is a system for enhancing user experience when navigating on websites. It is used in the domain of sightseeing and itinerary planning. Depending on user's goals and preferences, the system builds adapted itineraries. If the user wants a highly customised itinerary, he can choose the manual adaptation that allows him to interact directly with the system and thus to refine the process of adaptation.

The FRAKAS system [4] is also an opportunistic system. During the reasoning process, FRAKAS triggers interactions with an expert, on the fly, for acquiring missing domain knowledge. Knowledge is said to be missing when an inconsistency appears in the proposed solution. On a dedicated interface, the expert can highlight inconsistent knowledge. A reasoning mechanism processes this inconsistency and proposes possible ways of solving them to the expert. Depending on the expert answer, new knowledge is acquired by the system.

Like [5], [1] uses an opportunistic approach for AK acquisition. In the second version of Taaable, which implements the approach proposed in [1], users may give some feedback on substitutions for adapting recipes. If a proposition of substitution is judged irrelevant by the user, an interface allows the user to guide the system for repairing the adaptation. The user may indicate that some ingredient(s) is/are missing when adding an ingredient, or that some ingredients of the recipe are not compatible with an ingredient that must be added. In the case where is ingredient(s) missing, a system of AK acquisition, called Cabamaka [7] based on KDD, is triggered. This last step allows to repair the bad adaptation and memorise the AK. The AK is a rule composed of ingredient(s) that have to be removed and ingredient(s) that have to be added, similar to a *substitution* AK used in WikiTaaable.

In each of the previous systems, knowledge acquisition is triggered when an adaptation fails. The originality of our approach is that AK can be triggered in parallel of the adaptation process and that this knowledge can be acquired at any time in a semantic wiki collaborative space.

7 Conclusion and ongoing work

In this paper, we described an approach for interactive acquisition of AK. We implemented this approach in Taaable, a case-based reasoning tool for adapting cooking recipes. Our approach is based on the discovery of closed itemsets (CIs), a datamining technique. First, we define a formal context in which the CIs are mined. Then, we transform these CIs in substitutions. Because a huge number of CIs is produced, we use support and stability to filter and rank the CIs before presenting them to the expert.

We have conducted several experiments to measure the influence of support and stability thresholds on the nature of the AK presented to the expert. We found that instantiated AK are, in proportion, more relevant than generalised AK. A way to obtain a good number of generic AK is to filter instantiated AK, and to adjust support and stability consequently. However, depending on the expert expectations, values for support and stability may vary. This is why we let the expert set these values himself within the interface. The interface al-

lows the expert not only to define the context in which CIs will be discovered, but also to validate and/or modify AK found by the automatic process. Therefore, this interface allows the expert to refine the quality of knowledge.

A future work to improve user interaction is to further integrate our generic AK acquisition interface in TAAABLE. By doing so, we want to allow experts to use at any time (e.g. to acquire additional AK while adapting a specific recipe). We believe that a better integration in the interface, and a connection with the TAAABLE inference engine will help experts by enabling them to perform AK acquisition in "context". For example, we could show to expert the consequences of some AK on actual recipes by providing him actual examples. By asking him "Are you sure you want to replace butter with bacon grease in this sweet cake recipe?", we help him to become aware of the applicability of the acquired rule. Therefore, this will help the expert to judge more easily the relevance of a rule.

Another future work concerns the extension of the interface by some functionalities that will help the expert to better determine the context of application of a generic AK. Datamining techniques like FCA can help again for discovering regularities in recipes linked to the set of specific AK that produce a generic AK. Indeed, the properties of these recipes, e.g. the ingredients they use, the dish types they produce, their origin, can be used as entry of a new datamining process. On the example presented in Figure 5 about the substitution of `Butter milk` with `Sour milk` with a support of 36, such a process will show that for 32 of these 36 recipes are `Cheese cake` recipes, the 4 others being `Cake` recipes. Expected results are also about some more precise context of application of generic AK, according to ingredients used in recipe, e.g. in `Salad` containing `Fish`, `Vinegar` can be replaced with `Lemon juice`.

8 Acknowledgements

This work is supported by French National Agency for Research (ANR), program Contint 2011 (ANR-10-CONTINT-025). More information about Kolflow is available on the project website: `http://kolflow.univ-nantes.fr/`.

REFERENCES

[1] F. Badra, A. Cordier, and J. Lieber, 'Opportunistic Adaptation Knowledge Discovery', in *8th International Conference on Case-Based Reasoning - ICCBR 2009*, eds., Lorraine McGinty and David C. Wilson, volume 5650 of *Lecture Notes in Computer Science*, pp. 60–74, Seattle, United States, (July 2009). Springer.

[2] R. Bendaoud, A. Napoli, and Y. Toussaint, 'Formal Concept Analysis: A unified framework for building and refining ontologies', in *16th International Conference on Knowledge Engineering and Knowledge Management - EKAW 2008*, eds., A. Gangemi and J. Euzenat, volume 5268 of *Lecture Notes in Artificial Intelligence*, pp. 156–171, Acitrezza, Catania, Italie, (2008). Springer Berlin / Heidelberg.

[3] Z. Chen and Q. Zhu, 'Query construction for user-guided knowledge discovery in databases', *Inf. Sci.*, **109**, 49–64, (August 1998).

[4] A. Cordier, *Interactive and Opportunistic Knowledge Acquisition in Case-Based Reasoning*, Thèse de doctorat en informatique, Universite Lyon 1, November 2008.

[5] A. Cordier, B. Fuchs, J. Lieber, and A. Mille, 'Interactive Knowledge Acquisition in Case Based Reasoning', in *Workshop on Knowledge Discovery and Similarity, a workshop of the seventh International Conference on Case-Based Reasoning (ICCBR-07)*, eds., D. Wilson and D. Khemani, Belfast, United Kingdom, (August 2007).

[6] A. Cordier, E. Gaillard, and E. Nauer, 'Man-Machine Collaboration to Acquire Cooking Adaptation Knowledge for the TAAABLE Case-Based Reasoning System', *WWW 2012 - SWCS'12 Workshop*, 1113–1120, (April 2012).

[7] M. d'Aquin, F. Badra, S. Lafrogne, J. Lieber, A. Napoli, and L. Szathmary, 'Adaptation Knowledge Discovery from a Case Base', in *17th European Conference on Artificial Intelligence ECAI06*, ed., Traverso, Trento, Italy, (August 2006). IOS Press.

[8] *Advances in Knowledge Discovery and Data Mining*, eds., U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, AAAI/MIT Press, Menlo Park, CA, USA, 1996.

[9] E. Gaillard, J. Lieber, and E. Nauer, 'Adaptation knowledge discovery for cooking using closed itemset extraction', in *The Eighth International Conference on Concept Lattices and their Applications - CLA 2011*, pp. 87–99, Nancy, France, (October 2011).

[10] B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*, Springer, 1999.

[11] M. Krötzsch, D. Vrandecic, and M. Völkel, 'Semantic mediawiki', in *International Semantic Web Conference*, eds., I. F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, volume 4273 of *Lecture Notes in Computer Science*, pp. 935–942. Springer, (2006).

[12] S. O. Kuznetsov, 'On stability of a formal concept', *Annals of Mathematics and Artificial Intelligence*, **49**(1-4), 101–115, (April 2007).

[13] D. Leake, A. Kinley, and D. C. Wilson, 'Acquiring Case Adaptation Knowledge: A Hybrid Approach', in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 684–689, Belfast,Portland, Oregon, United States, (August 1996). AAAI Press.

[14] D. Leake and J. Powell, 'Mining Large-Scale Knowledge Sources for Case Adaptation Knowledge', in *Proceedings of the Seventh International Conference on Case-Based Reasoning*, eds., R Weber and M Richter, pp. 209–223, Belfast, United-Kingdom, (August 2007). Springer Verlag.

[15] R. Lopez De Mantaras, D. McSherry, D. Bridge, D Leake, B. Smyth, S. Craw, B. Faltings, M. L. Maher, M. T. Cox, K. Forbus, M. Keane, A. Aamodt, and I. Watson, 'Retrieval, reuse, revision and retention in case-based reasoning', *Knowl. Eng. Rev.*, **20**, 215–240, (September 2005).

[16] C. K. Riesbeck and R. C. Schank, *Inside Case-Based Reasoning*, Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey, 1989.

[17] Y. Shidochi, T. Takahashi, I. Ide, and H. Murase, 'Finding replaceable materials in cooking recipe texts considering characteristic cooking actions', in *ACM multimedia 2009 workshop on Multimedia for cooking and eating activities*, pp. 9–14, (2009).

[18] A. Silberschatz and A. Tuzhilin, 'On subjective measures of interestingness in knowledge discovery.', in *In Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pp. 275–281, Montreal, Quebec, Canada, (August 1995).

[19] H. Skaf-Molli, P. Molli, E. Desmontils, E. Nauer, Y. Toussaint, G. canals, A. Cordier, and M. Lefevre, 'Knowledge Continuous Integration Process (K-CIP)', in *WWW2012 Workshop on Semantic Web Collaborative Spaces (SWCS2012)*, (2012).

[20] L. Szathmary and A. Napoli, 'CORON: A Framework for Levelwise Itemset Mining Algorithms', *Supplementary Proc. of The Third International Conference on Formal Concept Analysis (ICFCA '05), Lens, France*, 110–113, (2005).

[21] M. J. Zaki and C.-J. Hsiao, 'CHARM: An efficient algorithm for closed itemset mining', in *SIAM International Conference on Data Mining SDM'02*, pp. 33–43, (2002).

A Machine Learning Approach to Recipe Text Processing

Shinsuke Mori and **Tetsuro Sasada** and **Yoko Yamakata** and **Koichiro Yoshino** [1]

Abstract. We propose a machine learning approach to recipe text processing problem aiming at converting a recipe text to a work flow. In this paper, we focus on the natural language processing (NLP) such as word identification, named entity recognition, and syntactic analysis to extract predicate-argument structures (tuples of a verbal expression and its arguments) from a sentence in a recipe text. Predicate-argument structures are subgraphs of the work flow of a recipe.

We solve these problems by methods based on machine learning techniques. The recipe domain is, however, different from the general domain in which many language resources are available. And we have to adapt NLP systems to the recipe texts by preparing annotated data in the recipe domain. To reduce the cost of the adaptation, we adopt a pointwise framework allowing to train analyzers from partially annotated data.

The experimental results showed that an adaptation works well for each NLP and with all the adaptations the accuracy of the entire system increased. We can conclude that more adaptation work helps develop an accurate recipe-text-to-flow system.

1 Introduction

Cooking is one of the most important and complicated tasks in daily life. Because of the variety and creativity of Japanese cooking, many Japanese people cooking at home are interested in learning new recipes. Many portal sites offer millions of recipes created by not only professional chefs but also by people cooking at home. For example, COOKPAD[2] provides more than two millions of recipes submitted by house chefs, and the number is increasing.

However, having a multitude of recipes is not always a good thing. Because consumer created recipe texts use a variety of expressions and writing styles, there are many recipes which are different at the word comparison level but describe the same cooking process. For example, we found 4,529 recipes for "*nikujaga*" (a typical Japanese dish)" on COOKPAD, but nobody knows how many unique cooking processes these describe. This problem prevents users from discovering a new cooking processes for a dish. Thus it is important to abstract recipe texts and make it possible to measure the similarity between them.

One of the best abstract representations for measuring the similarity of recipes is a graph [16]. Since cooking is a sequence of semi-ordered actions, we can represent it as a directed acyclic graph (DAG), called a work flow, where nodes correspond to the cooking actions and the final nodes correspond to dishes to be served. In fact, some recipe portal sites provide a work flow for their recipes [3].

There have been some attempts at converting the texts of recipes into a work flow. One of the pioneering works [6] proposed a semi-automatic method for converting recipe texts in Japanese into work flows. This work is, however, not mature from the viewpoint of natural language processing (NLP) and graph theory. The authors just use NLP tools designed for general domain texts such as newspaper articles with simply add some words in the recipe domain. For this reason their NLP part is not good at the recipe writing style, orthographical variants, and others[4]. In addition, their work flow construction process is based on manually created rules. Thus it worth renewing this method by adopting state-of-the-art NLP technologies and graph theory.

On this background, we propose a machine learning approach aimed at converting recipe texts into work flows. In this paper, we focus on NLPs and describe the procedure used to extract predicate-argument structures (tuples of a verbal expression and its arguments) from sentences in a recipe text. Predicate-argument structures are subgraphs of the whole work flow of the recipe. We leave the work flow construction for future work.

A machine learning approach provides following two advantages. The first is that we can steadily increase the accuracy just by increasing the amount of training data. After constructing the system, we only need to spend our time on generating annotated data. In addition, our framework allows us to increase the accuracy with minimal cost. The second advantage is robustness. Rule-based methods always suffer from the inability to handle exceptions such as orthographical variants, unusual wording, etc. Our approach allows us to divide the recipe analysis procedures into algorithms based on machine learning and annotated data provided by annotators.

The recipe domain is, however, different from the general domain in which many language resources are available. And we have to adapt NLP modules to the recipe texts by preparing annotated data in the recipe domain. To reduce the cost of the adaptation, we adopt a pointwise framework which allows us to train analyzers from partially annotated data. Text processing systems in this framework do not require whole sentences to be annotated (full annotation), but can use sentences in which only domain specific words and expressions are annotated with linguistic behavior (partial annotation).

In the evaluation, we report the accuracies of each NLP procedure and the effects of the adaptation to the recipe domain, as well as the accuracy of the entire system.

The design we describe in this paper is not limited to the recipe

[1] Kyoto University, Japan, email: forest@i.kyoto-u.ac.jp

[2] http://cookpad.com/ (accessed in June, 2012).

[3] For example http://www.cookingforengineers.com/ (accessed in June, 2012).

[4] In Japanese for example, onion can be written in both ideograms or phonetic alphabets.

domain. It is much more general and allows us to develop a text processing system in a specific domain very quickly with low cost.

2 Recipe Text Analysis

In this section, we describe the details of our method for extracting tuples of a predicate and its arguments, called a predicate-argument structure, from sentences in recipe texts. Our method is divide into three natural language processing (NLP) tasks: word recognition, named entity recognition, and syntactic analysis. Since all of them are based on machine learning, their accuracy is high as far as the training data is available. In addition, we adopt a pointwise approach which enables the tools for each NLP module to be trained from partially annotated data [12].

2.1 Recipe Text

A recipe text is composed of "the list of foods used as ingredients" and "text describing the step-by-step instructions on how to cook the dish." In this paper, we focus on the text part from which the work flow is constructed.

The text part consists of several short sentences describing chef actions and food state transitions, which we call events. Sometimes a sentence includes more than one event. From a linguistic viewpoint, they are mostly straightforward because there is no modality problems, less use of the passive form, and less tense variance. Thus, almost all events can be represented by tuples of a predicate and its arguments. For example, "cut an apple with a fruit knife" becomes

cut(*obj.*: an apple., with: a fruit knife),

where the predicate is "cut" and the arguments are "an apple" as the object and "a fruit knife" connected to the predicate with a preposition "with."

Differences between this kind of text and the general text used to train the NLP modules, such as newspaper articles or dictionary example sentences, causes problems. Thus, it is important to perform domain adaptation for each NLP module.

2.2 Word segmentation

The first step of text processing is to identify words in sentences. For languages such as Japanese and Chinese which do not have obvious word boundary, the word identification problem is solved by segmenting a sentence into a word sequence. For inflective languages such as English and French, this problem is solved by estimating the canonical (dictionary) form of each word. In this step, we also estimate part-of-speech (POS) tags for each word which are used during the syntactic analysis.

The recipe texts we use in the experiment are written in Japanese. Thus the first problem is word segmentation. The input is a sentence as follows:

水４００ｃｃを鍋で煮立て、沸騰したら中華スープの素を加えてよく溶かす。
(Heat 400 cc of water in a pot, and when it boils, add Chinese soup powder and dissolve it well.)

And the output is a word sequence as follows:

水|４-０-０|ｃ-ｃ|を|鍋|で|煮-立-て|、|沸-騰|し|た-ら|中-華|ス-ー-プ|の|素|を|加-え|て|よ-く|溶-か|す|。

where "|" and "-" mean existence and non-existence of a word boundary, respectively. Note that we divide inflectional endings from the stem and we do not need stemming.

To solve the word segmentation problem in the recipe domain, we adopt the pointwise approach [13]. The main reason is that this approach allows us to train a model by referring to partially annotated sentences, in which some parts between characters are annotated with word boundary information and some are not, as in the following example:

水␣４␣０␣０␣ｃ␣ｃ␣を|鍋|で|煮-立-て|る
(Heat 400 cc of water in a pot)

where "␣" means the lack of word boundary information. In the pointwise approach, we can focus our resources on the annotation of difficult parts, for example on domain specific words and expressions.

We formulate word segmentation as a binary classification problem as in [14], Our word segmenter, given a sentence $x_1x_2\cdots x_h$, estimates boundary tag b_i between characters x_i and x_{i+1}. Tag b_i = 1 indicates that a word boundary exists, while b_i = 0 indicates that a word boundary does not exist. This classification problem can be solved by support vector machines [4].

We use information about the surrounding characters (character and character-type n-grams), as well as the presence or absence of words in the dictionary as features (see Table 1). Specifically dictionary features for word segmentation l_s and r_s are active if a string of length s included in the dictionary is present directly to the left or right of the present word boundary, and i_s is active if the present word boundary is included in a dictionary word of length s.

Table 1. Features for word segmentation.

Type	Feature strings
Character n-gram	x_l, x_r, $x_{l-1}x_l$, x_lx_r, x_rx_{r+1}, $x_{l-1}x_lx_r$, $x_lx_rx_{r+1}$
Character type n-gram	$c(x_l)$, $c(x_r)$, $c(x_{l-1}x_l)$, $c(x_lx_r)$, $c(x_rx_{r+1})$, $c(x_{l-2}x_{l-1}x_l)$, $c(x_{l-1}x_lx_r)$, $c(x_lx_rx_{r+1})$, $c(x_rx_{r+1}x_{r+2})$
dictionary	l_s, r_s, i_s

x_l and x_r indicate the characters to the left and right of the word boundary in question. The function $c(\cdot)$ converts a character sequence into the character type sequence. l_s, r_s, and i_s represent the left, right, and inside dictionary features.

2.3 Named Entity Recognition

A single word does not always correspond to an object or an action in the real world, but a word sequence does. Thus the next step of our system is to recognize such word sequences, which are called named entities (NE). For recipe text recognition, we adopt the following NE types: Food (F), Quantity (Q), Tool (T), Duration (D), State (S), chef's action (Ac), or foods' action (Af).

NE recognition is normally solved as a sequence labeling problem for each word based on the IOB2 tagging system. So the NE tags are extended by adding B and I to denote the beginning and the continuation of a named entity. In addition, words which is not a part of any NE are annotated with O. Thus the tag set is $\mathcal{T}$ = { F, Q, T, D, S, Ac, Af } × { B, I } ∪ O. For example, the following annotation means

		w			
$P(y\|w)$	水	４００	ｃｃ	を	$\cdots$
F-B	0.62	0.00	0.00	0.00	$\cdots$
F-I	0.37	0.00	0.00	0.00	$\cdots$
Q-B	0.00	0.82	0.01	0.00	$\cdots$
y Q-I	0.00	0.17	0.99	0.00	$\cdots$
T-B	0.00	0.00	0.00	0.00	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$
O	0.01	0.01	0.00	1.00	

Figure 1. Best path search in named entity recognition.

that the word "水" (water) is a food, the word sequence "４００ ｃｃ" is a quantity, and the word "を" (the case marker for an object) is not an NE.

水/F-B ４００/Q-B ｃｃ/Q-I を/O

For NE recognition we extend the pointwise approach to allow a partially annotated corpus as a training data. First we estimate the parameters of a classifier based on logistic regression [4] from fully and/or partially annotated data. Then, given a word sequence, the classifier enumerates all possible tags for each word with their probabilities (see Figure 1). Finally our NE recognizer searches for the tag sequence of the highest probability satisfying the constraints[5].

2.4 Syntactic Analysis

The final disambiguation process used to extract predicate-argument tuples is to determine the syntactic structure of words and NEs in a sentence. This paper follows the standard setting of recent work on dependency parsing. Each word in a sentence syntactically modifies only one other word, called its head, except for the head word of the sentence [3]. Thus the output is a tree where the nodes are words and the arcs express a dependency relationship.

Formally given as input a sequence of words, $\boldsymbol{w} = \langle w_1, w_2, \ldots, w_n \rangle$, the goal of syntactic analysis is to output a dependency tree $\boldsymbol{d} = \langle d_1, d_2, \ldots, d_n \rangle$, where $d_i = j$ when the head of w_i is w_j. We assume that $d_i = 0$ for some word w_i in a sentence, which indicates that w_i is the head of the sentence.

State-of-the-art syntactic analyzers (parser) are based on machine learning. The parameters are estimated from an annotated corpus in the general domain. Thus when we think of applying the parser to recipe texts, once again domain adaptability is an important point. For this reason, we adopt a pointwise approach [5]. A parser based on this approach allows us to estimate the parameters from partially annotated data in addition to normal fully annotated data. This parser is one of several based on the maximum spanning tree (MST) framework [9].

At the step of analysis, first the parser assigns a score $\sigma(d_i)$ to each edge (i.e. dependency) d_i, then finds a dependency tree, $\hat{\boldsymbol{d}}$, that maximizes the sum of the scores of all the edges.

$$\hat{\boldsymbol{d}} = \operatorname*{\mathbf{argmax}}_{\boldsymbol{d}} \sum_{d \in \boldsymbol{d}} \sigma(d). \tag{1}$$

In the training phase $\sigma(d_i)$ is estimated for each w_i independently by a log-linear model [1]. Contrary to the original MST parser that estimates $\sigma(d)$ with a perceptron-like algorithm that optimizes the score of entire dependency trees, a pointwise parser calculates the probability of a dependency labeling $p(d_i = j)$ for a word w_i from its context, which is a tuple $x = \langle \boldsymbol{w}, \boldsymbol{t}, i \rangle$, where $\boldsymbol{t} = \langle t_1, t_2, \ldots, t_n \rangle$ is a sequence of POS tags assigned to $\boldsymbol{w}$ by a POS tagger. The conditional probability $p(j|x)$ is given by the following equation:

$$p(j|x, \boldsymbol{\theta}) = \frac{\exp(\boldsymbol{\theta} \cdot \boldsymbol{\phi}(x, j))}{\sum_{j' \in \mathcal{J}} \exp(\boldsymbol{\theta} \cdot \boldsymbol{\phi}(x, j'))}. \tag{2}$$

The feature vector $\boldsymbol{\phi} = \langle \phi_1, \phi_2, \ldots, \phi_m \rangle$ is a vector of non-negative values calculated from features on pairs (x, j), with corresponding weights given by the parameter vector $\boldsymbol{\theta} = \langle \theta_1, \theta_2, \ldots, \theta_m \rangle$. The features are listed in Figure 2. We estimate $\boldsymbol{\theta}$ from sentences annotated with dependencies. It should be noted that the probability $p(d_i)$ depends only on i, j, and the inputs $\boldsymbol{w}$, $\boldsymbol{t}$, which ensures that it is estimated independently for each w_i. The pointwise approach enjoys greater flexibility, which allows for training from partially annotated corpora. Because parameter estimation does not involve computing $\hat{\boldsymbol{d}}$, the parser does not apply the maximum spanning tree algorithm in training.

F1 The distance between a dependent word and its candidate head.
F2 The surface forms of the dependent and head words.
F3 The parts-of-speech of the dependent and head words.
F4 The surface forms of up to three words to the left of the dependent and head words.
F5 The surface forms of up to three words to the right of the dependent and head words.
F6 The parts-of-speech of the words selected for F4.
F7 The parts-of-speech of the words selected for F5.

Figure 2. Features for syntactic analysis.

2.5 Predicate-Argument Structure Analysis

After the three disambiguation procedures described above, the input sentence is transformed into a dependency tree where nodes are words and some subtrees are annotated with an NE tag. Then we execute the following steps from the beginning of the text to the end to extract tuples of a predicate and its arguments, called predicate-argument structure, from it.

1. Find the next NE tagged with Ac or Af.

   煮立て/Ac (boil)

2. Set that NE as the predicate with unknown arguments.

   煮立て (??, ??, ...)

3. Enumerate all the NE sequences depending on the predicate by referring to the dependency tree. Note that many of them are connected indirectly to the predicate with a case marker which is apparent from the POS in Japanese.

   /水/F (water) /４００ ｃｃ/Q を (*obj.*, case marker)
   /鍋/T (pot) で (by, case marker)

4. Construct a predicate-argument structure using the predicate and these sequences. Note that we add a case marker for each argument tagged with F (food) or T (tool) to clarify the semantic role (subject, object, etc.) of the NE in regards to the predicate.

[5] For example, "F-B S-I" is invalid.

Table 2. Fully annotated corpora.

corpus name	#sentences	#words	#characters	#NEs	#dependencies
BCCWJ	53,899	1,275,135	1,834,784	–	–
recipe	242	4,704	7,023	1,523	–
Dict. sentences	11,700	147,809	197,941	–	136,109
Newspaper art.	9,023	263,425	398,569	–	254,402
recipe	724	13,150	19,966	3,797	12,426

煮立て (*obj.*:水-４００-ｃｃ, で:鍋)
boil(*obj.*:water 400cc, by:pot)

The procedure described above does not cover some linguistic phenomena such as zero-anaphora, causative form, relative clause, etc. These phenomena require some additional disambiguation because they span more than one sentences. The text processing component could output possible candidates with probability so that the work flow construction component can execute disambiguation as an optimization problem. We leave this part as future work.

3 Evaluation

We developed a recipe text analysis system based on the framework we explained in Section 2 In this section, we present experimental results on real recipe texts and evaluate our framework.

3.1 Experimental Settings

There are various language resources available in the general domain. But are not suitable for recipe text analysis because of domain differences. But they can be used for parameter estimation of baseline NLP systems. For the word segmentation step we use the Balanced Corpus of Contemporary Written Japanese (BCCWJ) [8] which contains sentences annotated with word boundary information and words annotated with POS tags. We also used a dictionary UniDic (version 1.3.12) [6] and the list of arabic numbers, first names, family names, and signs. The total number of entries is 423,489 words. For the syntactic analysis step we use sentences extracted from a dictionary [7] and *Nikkei* newspaper articles[7]. These sentences are annotated with word boundary information and the dependency structure. NEs to be recognized are highly domain dependent. NEs in the general domain such as names of people, names of organizations, and dates, etc. are not useful in recipe text processing. But we need NEs specific to the recipe domain as enumerated in Subsection 2.3. So we prepared a small recipe corpus annotated with these NEs. Table 2 shows the specifications of these fully annotated corpus. In addition we prepared a partially annotated corpus in the recipe domain for each procedure. The details are described in each subsection. The test data is randomly selected 100 recipes taken from COOKPAD in all the evaluations.

3.2 Word Segmentation

The first step is word segmentation. The baseline system, KyTea[8] [13], is based on a linear SVM [4], which decides if there is a word boundary or not at each point. The parameters are estimated from the BCCWJ, the dictionary sentences, and newspaper articles.

[6] Available at `http://www.tokuteicorpus.jp/dist/` (accessed in June, 2012).
[7] `http://e.nikkei.com/` (accessed in June, 2012)
[8] Available at `http://www.phontron.com/kytea/` (accessed in June, 2012).

煮立て (Freq=1497)
中␣火␣で|煮-立-て|、␣（␣１␣）␣の␣ほ␣う␣れ␣ん ···
Ａ␣を|煮-立-て|、␣（␣１␣）␣の␣し␣い␣た␣け␣・ ···
鍋␣に␣Ｂ␣を␣加␣え|煮-立-て|る␣。

Figure 3. Partial annotation for word segmenter adaptation. An annotator checks if a string “煮立て” (boil) is a word in the context. The meaning of the symbols between characters are explained in Subsection 2.2.

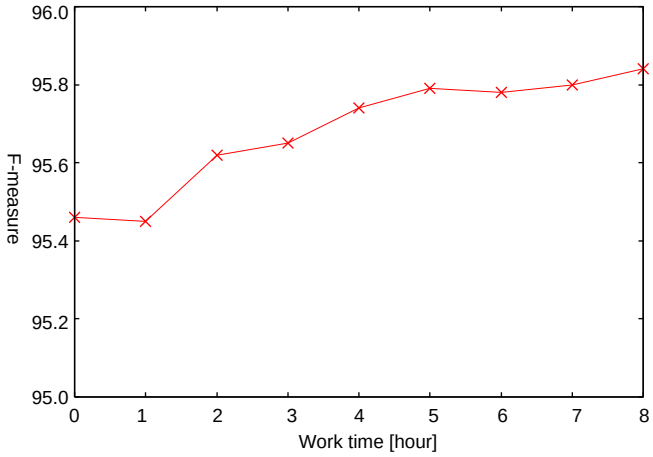

Figure 4. Learning curve of word segmentation.

For the domain adaptation, we first extracted unknown word candidates from a large raw recipe text by the distributional analysis [10]. Then we showed an annotator three occurrences for each unknown word candidate with its contexts in keyword in context (KWIC) style shown in Figure 3. Then the annotator checked if these strings are a word in that context and changed the word boundary information if necessary. The total work time was eight hours. We measured the word segmentation accuracy after each hour.

As an evaluation measure, we use word F-measure following [11]. Roughly speaking, the F-measure represents the ratio of the correctly recognized words over all the words.

Figure 4 shows the learning curve of word segmentation. The accuracy of the baseline is lower than the accuracy in the general domain (98.13%) reported in [13]. With adding a partially annotated corpus by checking the unknown word candidates, the accuracy increases gradually. From the curve, it can be said that the accuracy has not been saturated and more annotation work contributes to a further

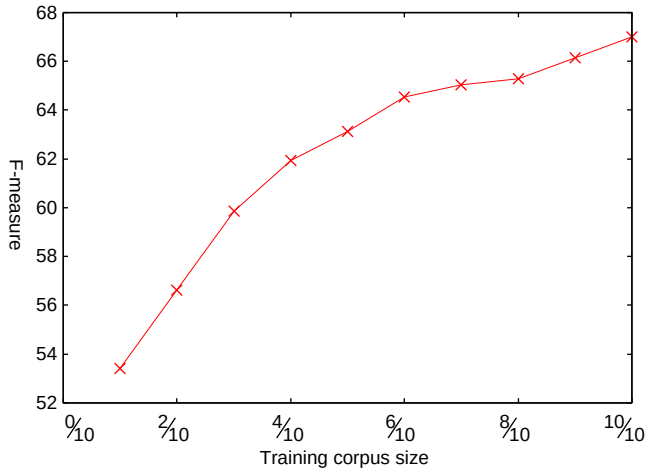

Figure 5. Learning curve of named entity recognition.

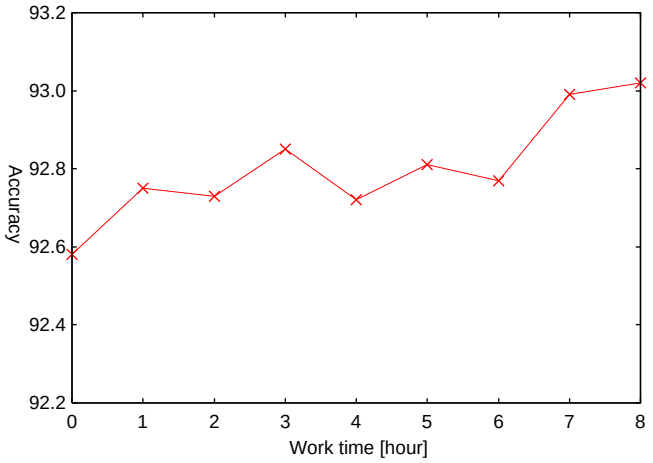

Figure 6. Learning curve of syntactic analysis.

improvement.

3.3 Named Entity Recognition

The next step is named entity recognition. The baseline system first enumerates all possible tags for each word with their probabilities using logistic regression. The parameters are estimated from 1/10 of the recipe in Table 2 corpus whose words are annotated with NE tags. Then our NE recognizer searches for the best tag sequence.

For the domain adaptation, we simply increased the training data size from 1/10 to 10/10. The total annotation time was about five hours. We measured the named entity recognition accuracy for the training size of 1/10, 2/10, ..., 10/10.

The evaluation measurement is F-measure, the harmonic mean of the precision and the recall [2]. The precision is the ratio of the NEs correctly recognized by the system over all the recognized NEs and the recall is the ratio of the correctly recognized over all the NEs in the test corpus.

Figure 5 shows the learning curve of named entity recognition. The accuracy of the baseline, the left most point in the graph, is very low compared with the F-measure of 80.17 reported in [15] for named entity recognition in the general domain. The reason is that the training data size of the baseline is much smaller than 12,000 sentences used in [15]. By adding annotated sentences, the accuracy increases steadily. But it is still lower than the F-measure of 80.17 reported in the general domain. This shows that we need more training data.

3.4 Syntactic Analysis

The last disambiguation is syntactic analysis. The parser, EDA[9] [5], requires words annotated with POS tags. But the training corpus and the test corpus (see Table 2) are not annotated with POS tags, so we used a Japanese POS tagger, KyTea [13], trained on the BCCWJ. Then we built the baseline parser from the dictionary sentences and newspaper articles.

[9] Available at `http://www.ar.media.kyoto-u.ac.jp/members/flannery/eda/index_en.html` (accessed in June, 2012).

For the domain adaptation, first we constructed a dictionary containing all the sequences of a noun and a postposition appearing in the training corpus. Then we searched for new noun-postposition sequences in the raw recipe text from the beginning and showed them to an annotator. Then he/she annotated them with the dependency destination (normally the verb). The total work time was 8 hours. We measured the syntactic analysis accuracy after each hour.

The evaluation measurement is the accuracy defined as the ratio of words with the correct dependency destination over all the words. We discarded the last word in the sentences because in written Japanese it is always the sentence head.

Figure 6 shows the learning curve of syntactic analysis. The accuracy of the baseline, the left most point in the graph, is lower than the accuracy 96.83%[10] reported in [5]. The reason is the difference in domain between the training data and the test data. By adding a partially annotated corpus prepared by checking new sequences of a noun and a postposition appearing in the raw recipe text, the accuracy increased gradually. From the curve, it can be said that we can expect further improvements just by continuing the annotation work.

3.5 Overall System

Finally we measured the accuracy of the overall system before and after all the adaptations. In this experiment the input is a sentence. The sentence is automatically segmented into words, named entities are automatically extracted, and the sentence is converted automatically into a dependency tree.

Similar to named entity recognition the evaluation measure for the overall system is F-measure. The only difference lies in the definition of the units to be recognized. Here they are the predicate-argument pairs. For example, the predicate-argument structure shown in the end of Section 2 has two following pairs:

1. ⟨煮立て, *obj.*:水-４００-ｃｃ⟩ ⟨boil, *obj.*:water 400cc⟩
2. ⟨煮立て, で:鍋⟩ ⟨boil, by:pot⟩

A pair is correct if and only if the predicate and its argument are the same including the case markers.

[10] The parser was trained on the dictionary sentences and tested on different sentences in the same domain

The F-measure of the cascade combination of the baseline systems was 42.01. After the adaptations of all the procedures, the F-measure increased to be 58.27. The baseline accuracy is low but after the adaptations the performance improved drastically by 28.0% error elimination. The accuracy after the adaptations, still, may not be sufficient for the succeeding work flow construction process. The total work time for corpus annotation is only 8+5+8 = 21 hours. We can easily double or triple the work time to have a further improvement.

The framework of all the procedures we propose in this paper requires only annotations to words or expressions specific in the recipe domain and allows us to improve the overall system performance very easily. From the comparison among the learning curves of all the procedures (Figure 4, 5, 6), it may be a good strategy to spend our annotation work more on NE, since the baseline accuracy of named entity extraction is much lower than the other procedures and the accuracy gain realized by the adaptation is much larger.

4 Conclusion

In this paper, we presented a machine learning approach to recipe text processing aiming at converting a recipe text to a work flow. We focused on NLPs and describe the procedures to extract predicate-argument structures from a sentence in a recipe text. In the evaluation, we reported the accuracies of each NLPs and the effect of the adaptation to the recipe domain as well as the accuracy of the entire system. The design we describe in this paper is general and allows us to develop a text processing system in a certain domain very quickly with low cost.

For recipe work flow construction, the problems of some linguistic phenomena and the connection of predicate-argument structures are still remains. We will give a solution to these problems with the same design philosophy.

Acknowledgement

This work was supported by Grant-in-Aid for Scientific Research of the government of Japan (KAKENHI 23500177 and 23700144). The authors are also thankful to Mr. Yuichi Sugiyama for his annotation work.

REFERENCES

[1] Vincent J. Della Pietra Adam L. Berger, Stephen A. Della Pietra, 'A maximum entropy approach to natural language processing', *Computational Linguistics*, **22**(1), (1996).

[2] Andrew Borthwick, *A Maximum Entropy Approach to Named Entity Recognition*, Ph.D. dissertation, New York University, 1999.

[3] Sabine Buchholz and Erwin Marsi, 'Conll-x shared task on multilingual dependency parsing', in *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pp. 149–164, (2006).

[4] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin, 'LIBLINEAR: A library for large linear classification', *Journal of Machine Learning Research*, **9**, 1871–1874, (2008).

[5] Daniel Flannery, Yusuke Miyao, Graham Neubig, and Shinsuke Mori, 'Training dependency parsers from partially annotated corpora', in *Proceedings of the Fifth International Joint Conference on Natural Language Processing*, (2011).

[6] Reiko Hamada, Ichiro Ide, Shuichi Sakai, and Hidehiko Tanaka, 'Structural analysis of cooking preparation steps in japanese', in *Proceedings of the fifth international workshop on Information retrieval with Asian languages*, number 8 in IRAL '00, pp. 157–164, (2000).

[7] Donald Keene, Hiroyoshi Hatori, Haruko Yamada, and Shouko Irabu, *Japanese-English Sentence Equivalents*, Asahi Press, Electronic book edn., 1992.

[8] Kikuo Maekawa, 'Balanced corpus of contemporary written japanese', in *Proceedings of the 6th Workshop on Asian Language Resources*, pp. 101–102, (2008).

[9] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič, 'Non-projective dependency parsing using spanning tree algorithms', in *Conference on Empirical Methods in Natural Language Processing*, pp. 523–530, (2005).

[10] Shinsuke Mori and Makoto Nagao, 'Word extraction from corpora and its part-of-speech estimation using distributional analysis', in *Proceedings of the 16th International Conference on Computational Linguistics*, (1996).

[11] Masaaki Nagata, 'A stochastic japanese morphological analyzer using a forward-dp backward-a* n-best search algorithm', in *Proceedings of the 15th International Conference on Computational Linguistics*, pp. 201–207, (1994).

[12] Graham Neubig and Shinsuke Mori, 'Word-based partial annotation for efficient corpus construction', in *Proceedings of the Seventh International Conference on Language Resources and Evaluation*, (2010).

[13] Graham Neubig, Yosuke Nakata, and Shinsuke Mori, 'Pointwise prediction for robust, adaptable japanese morphological analysis', in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, (2011).

[14] Manabu Sassano, 'An empirical study of active learning with support vector machines for japanese word segmentation', in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 505–512, (2002).

[15] Kiyotaka Uchimoto, Qing Ma, Masaki Murata, Hiromi Ozaku, and Hitoshi Isahara, 'Named entity extraction based on a maximum entropy model and transformation rules', in *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pp. 326–335, (2000).

[16] Liping Wang, Qing Li, Na Li, Guozhu Dong, and Yu Yang, 'Substructure similarity measurement in chinese recipes', in *Proceedings of the 17th international conference on World Wide Web*, number 10, pp. 978–988, (2008).

Semi-automatic annotation process for procedural texts: An application on cooking recipes

Valmi Dufour-Lussier[123] and **Florence Le Ber**[1234] and **Jean Lieber**[123] and **Thomas Meilender**[1235] and **Emmanuel Nauer**[123]

Abstract. TAAABLE is a case-based reasoning system that adapts cooking recipes to user constraints. Within it, the preparation part of recipes is formalised as a graph. This graph is a semantic representation of the sequence of instructions composing the cooking process and is used to compute the procedure adaptation, conjointly with the textual adaptation. It is composed of cooking actions and ingredients, among others, represented as vertices, and semantic relations between those, shown as arcs, and is built automatically thanks to natural language processing.

The results of the automatic annotation process is often a disconnected graph, representing an incomplete annotation, or may contain errors. Therefore, a validating and correcting step is required. In this paper, we present an existing graphic tool named KCATOS, conceived for representing and editing decision trees, and show how it has been adapted and integrated in WIKITAAABLE, the semantic wiki in which the knowledge used by TAAABLE is stored. This interface provides the wiki users with a way to correct the case representation of the cooking process, improving at the same time the quality of the knowledge about cooking procedures stored in WIKITAAABLE.

Keywords: cooking, natural language processing, procedural texts, semantic annotation, semantic wiki.

1 Introduction

This paper presents how an automatic textual annotation process of procedural texts, like cooking recipes, can be improved, using a graphical interface plugin in a wiki and by involving wiki users for correcting and completing the result of the automatic annotation. This work has been done in the framework of TAAABLE, a case-based reasoning (CBR) system which adapts cooking recipes to user constraints. According to the user constraints, TAAABLE looks up, in the recipe base, whether some recipes satisfy these constraints. Such recipes, if they exist, are returned to the user; otherwise the system is able to retrieve similar recipes (i.e. recipes that match the target query partially), called source cases in the context of a CBR application, and to adapt these recipes, creating new ones. The knowledge required by TAAABLE is stored in semantic wiki named WIKITAAABLE. Adaptation consists first in substituting some ingredients of the source cases by the ones required by the user. Then, two additional adaptations are computed: the adaptation of ingredient quantities and the adaptation of the text of the preparation procedure [3]. This last adaptation requires to represent semantically the sequence of instructions composing the cooking process. Previous work, based on natural language processing (NLP), has been realised in order to transform the textual procedure into a semantic annotation automatically. This semantic annotation takes the form of a directed graph in which cooking actions and ingredients, among others, are represented as vertices, and the semantic relations between those are represented as arcs.

The main objective of this work is to make it possible for wiki users to edit the graph that is automatically generated for each recipe in WIKITAAABLE, in order to correct and eventually complete it. In this paper, we show how KCATOS, an existing graphic tool designed to display and edit decision trees, has been adapted and integrated in WIKITAAABLE.

The paper is organised as follows: Section 2 specifies the problem in its whole context and introduces TAAABLE and WIKITAAABLE. Section 3 introduces the graph representation used for recipes, the NLP methods used to extract those graphs, and textual adaptation. Section 4 explains our approach for editing and correcting the graphs using KCATOS. Section 5 concludes the paper.

2 Context and motivation

TAAABLE is a CBR system that has been designed to take part in the Computer Cooking Contest[6], an international contest which aims at comparing CBR system results on a common domain: cooking. Several challenges are proposed in this contest. Among them, two challenges (won by TAAABLE in 2010) are of specific interest to this work:

- the *main challenge*, in which CBR systems must return recipes satisfying a set of constraints given by the user, such as inclusion or rejection of ingredients, the type or the

[1] Université de Lorraine, LORIA, UMR 7503 — Vandœuvre-lès-Nancy, F-54506, France
[2] CNRS, LORIA, UMR 7503 — Vandœuvre-lès-Nancy, F-54506, France
[3] Inria — Villers-lès-Nancy, F-54602, France
[4] Université de Strasbourg/ENGEES, LHYGES, UMR 7517 — Strasbourg, F-67000, France
[5] A2ZI — Commercy, F-55200, France

[6] `http://computercookingcontest.net/`

origin of the dish, or its compatibility with some diets (vegetarian, nut-free, etc.). For example: a user may ask for "a dessert, with rice and fig", as illustrated in Fig. 1. Systems have to search into a limited set of (ca. 1500) recipes for recipes satisfying the constraints, and if there is no recipe satisfying all the constraints, the systems have to adapt existing recipes into new ones.

- the *adaptation challenge*, in which CBR systems must adapt a given recipe to specific constraints. For example, "adapt the 'My strawberry pie' recipe because I do not have strawberries".

2.1 Principles of TAAABLE

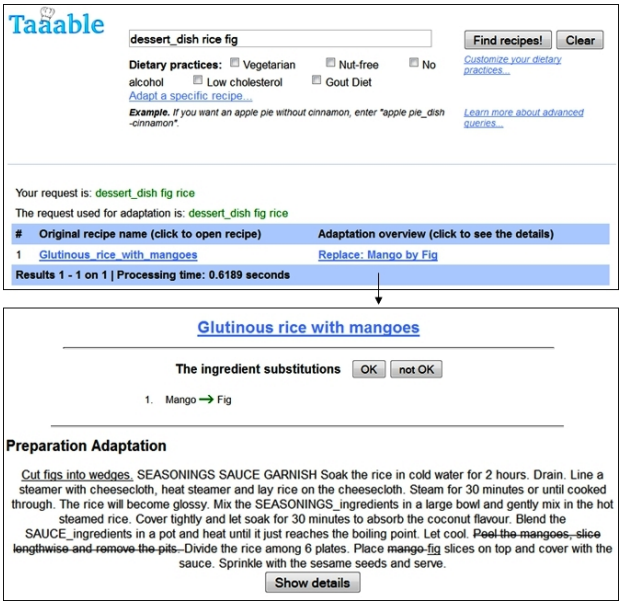

Figure 1: The TAAABLE interface. Queried for a dessert dish, with rice and fig, TAAABLE proposes to replace mango by fig in the "Glutinous rice with mangoes" recipe. After viewing the adapted recipe, the user can give feedback about the substitution ("OK" or "not OK").

Like many CBR systems [12], TAAABLE uses an ontology to retrieve the source cases that are the most similar to a target case (i.e. the query). TAAABLE retrieves and creates cooking recipes by adaptation. According to the user constraints, the system looks up, in the recipe base (which is a case base), whether some recipes satisfy these constraints. Recipes, if they exist, are returned to the user; otherwise the system is able to retrieve similar recipes (i.e. recipes that match the target query partially) and adapts these recipes, creating new ones. Searching similar recipes is guided by several ontologies, i.e. hierarchies of classes (ingredient hierarchy, dish type hierarchy, etc.), in order to relax constraints by generalising the user query. The goal is to find the most specific generalisation (with the minimal cost) for which recipes exist in the case base. Adaptation consists in substituting some ingredients of the source cases by the ones required by the user.

To deal with the adaptation of a specific recipe (which is the *adaptation challenge* problem), TAAABLE uses the same hierarchy based generalisation/specialisation mechanism on a recipe base containing only the recipe that has to be adapted. For example, when adapting the "My Strawberry Pie" recipe (in which strawberries are required) to the constraint "no strawberry", `Strawberry` is generalised on `Berry`, which is then specialised in another berry (`Raspberry`, `Blueberry`, `Blackberry`, etc.). Substitutions (e.g substitute `Strawberry` by `Raspberry`) are proposed to the user.

2.2 WIKITAAABLE

WIKITAAABLE[7] is a semantic wiki that uses Semantic MediaWiki [8] as support for encoding knowledge associated to wiki pages. WIKITAAABLE contains the set of resources required by the TAAABLE reasoning system, in particular an ontology of the domain of cooking, and recipes.

The cooking ontology is composed of 6 hierarchies: a *food* hierarchy (related to ingredients used in recipes, e.g. `Berry`, `Meat`, etc.), a *dish type* hierarchy (related to the types of recipes, e.g. `PieDish`, `Salad`, etc.), a *dish moment* hierarchy (related to the time for eating a dish, e.g. `Snack`, `Starter`, `Dessert`, etc.), a *location* hierarchy (related to the origins of recipes, e.g. `France`, `Asia`, etc.), a *diet* hierarchy (related to food allowed or not for a specific diet, e.g `Vegetarian`, `NutFree`, etc.), an *action* hierarchy (related to cooking actions used for preparing ingredients, `ToCut`, `ToPeel`, etc.). In the semantic wiki, each concept of a hierarchy is encoded as a category page `Category: <concept name>`. Each concept is described by a short description, lexical variants (used by the annotation bot, for searching concepts in the full text of recipes), its sub-categories and super-categories. For berries, the wiki page indicates that `Berry` is a sub-concept of `Fruit` (corresponding to the `Category:Fruit` page), and sub-categories of `Berry` (e.g. `Raspberry`, `Blueberry`, etc.) are listed. Each action has properties, both syntactic and semantic, associated to it, which makes the automatic case acquisition process described hereafter possible.

The set of recipes contained in WIKITAAABLE are those provided by the Computer Cooking Contest, that have been semantically annotated according to the domain ontology. Each recipe, as the one given in the example of Fig. 2, is encoded as a wiki page, composed of several sections: a title, which is the name of the recipe, an "*Ingredients*" section containing the list of ingredients used in the recipe, each ingredient being linked to its corresponding *Category* page in the food hierarchy, a "*Textual Preparation*" section describing the preparation process, some possible "*substitutions*" which are adaptation knowledge, and "*other information*" like the dish type, for example.

Additionally, for each recipe, there is a graph representing formally the preparation process, which constitutes a highly structured case representation usable by a CBR engine. For instance, TAAABLE uses this to propose a fully adapted recipe text to the user. Previously, WIKITAAABLE used a simple tree representation as shown in Fig. 3. This representation is limited and has not been designed to be edited, which is why it is being replaced by the more complete yet easier to edit representation shown in Fig. 4.

[7] `http://wikitaaable.loria.fr`

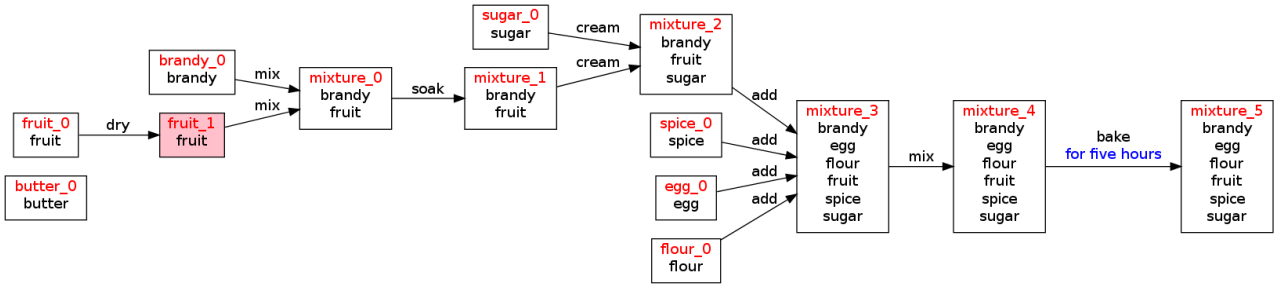

Figure 3: Simple tree representation of the recipe shown in Fig. 2.

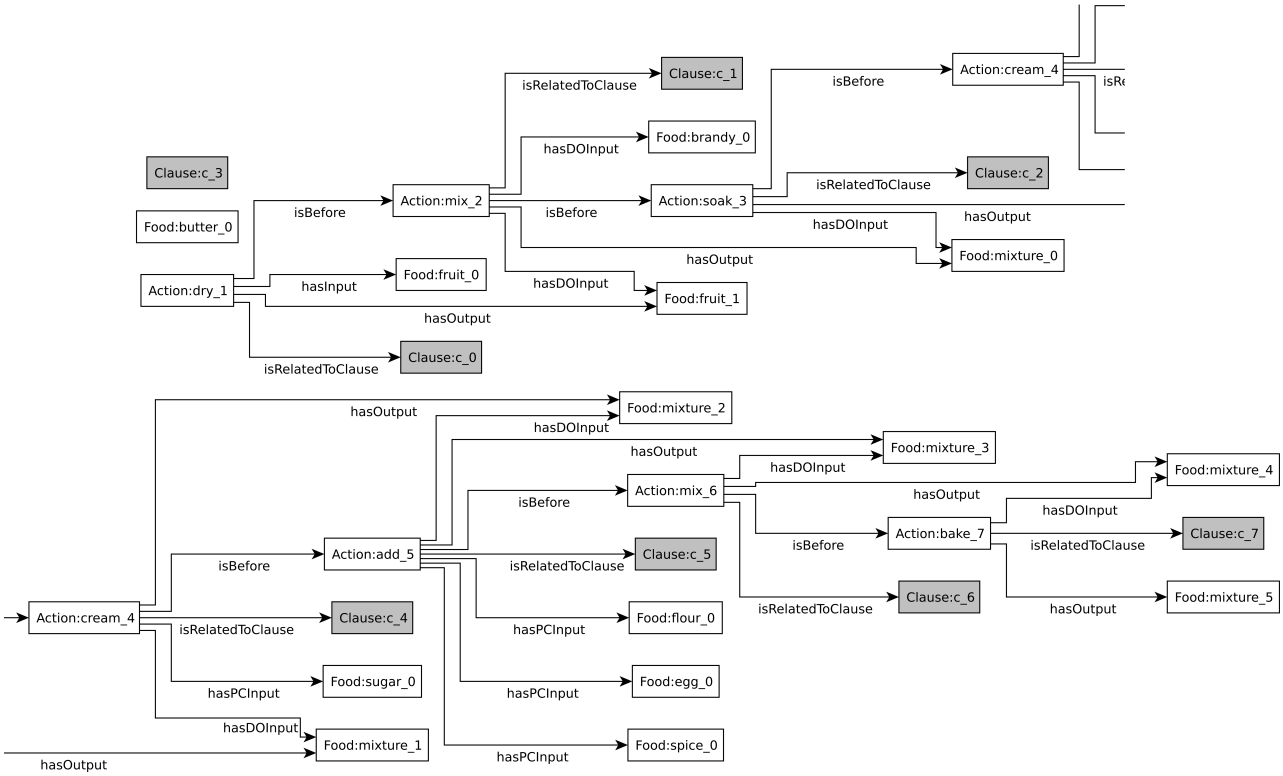

Figure 4: Recipe graph corresponding to the preparation of the recipe given Fig. 2 in KCatoS (split horizontally for readability).

Initially, this graph is acquired from the ingredients and the textual preparation parts through an automatic process making use of NLP methods, which will be presented in the next section.

However, NLP never gives perfect results, and this case acquisition application is no exception. While efforts are being invested in making the acquisition more accurate, if higher quality cases are required in the short term, user intervention is required.

An interface is proposed to make it possible for users to edit the graphs. Graph edition is accomplished in interaction with the automatic acquisition application, such that for each change entered by a user, the system is able automatically to suggest further changes that seem to be required.

Increasing the quantity and quality of the worklfows in the wiki will improve the results of the reasoning system.

The next section details our approach for automatic case acquisition and illustrates how the case representations are being used currently in Taaable.

Figure 2: Example of a WikiTaaable recipe.

3 Textual CBR adaptation

3.1 Case representation

Two important methods that are used in CBR to represent processes, including recipes, are workflows [11] and qualitative algebras [5]. A tree formalism presented earlier was also used in [6]. It is important that the representation of a recipe in WikiTaaable be general enough so that it could be translated to any of those formalisms.

To make semi-automatic annotation by users easier, the representation that is used is presented as a graph. However, each recipe graph is directly translatable to an equivalent knowledge base expressed for instance in description logics. In recipe graphs, each vertex represents the instantiation of a WikiTaaable category, for instance an action or a food item. An arc expresses a semantic link between two vertices. For one thing, each action vertex is related to a certain amount of food vertices through relations of type `hasDOInput` and/or `hasPCInput`, and to (generally) one food vertex through the relation `hasOutput`. `hasDOInput` refers to an input linguistically specified as a direct object, and `hasPCInput` as a prepositional complement: e.g. in "add milk to the batter", "milk" is a direct object and "batter" is a prepositional complement. Action vertices are also related to each others through relations such as `isBefore` or `isDuring`.

Normally, the preparation of a recipe comes to a point where all ingredients are mixed together to form a new whole. This indicates that a correct annotation of a recipe will normally be a connected graph.

3.2 NLP for procedural case acquisition

The case acquisition process proposed uses a combination of classical NLP tools which have been slightly adapted to give better results within the framework of procedural texts, as well as some new ideas that were implemented specifically for this type of texts. Other approaches are possible, e.g. [13] discusses two different methods based on information extraction to acquire cases as workflows.

A text is first tokenized (split in words), in order to make further linguistic analysis possible. Then, a part-of-speech tag are assigned to each word, indicating whether it is a verb, a noun, etc. This makes is possible to analyse sentences syntax to physically identify, for instance, actions and their arguments. For each action, an action vertex is added to the graph, as well as one (exceptionally, more than one) new food vertex corresponding to its output.

Those steps are not trivial, but effective tools exist, which are either based on stochastic machine learning (e.g. the Brill tagger [4] used for part-of-speech tagging) or rule-based (e.g. the chunker [1] used with hand-crafted grammar rules for syntactic analysis). But this is not sufficient, e.g. to identify properly which food items an action takes as input.

For instance, in a sentence such as "Peel the mangoes, slice lengthwise and remove the pits", it is easy as humans to understand that mangoes are being sliced and pitted, but some heuristic is needed is order for a computer to realise that. This is where the property in the action hierarchy of WikiTaaable comes in handy. Each action has an arity which makes it possible to tell when an argument is missing. In the sentence above, for instance, "slice" requires a direct object which is missing, and "remove", a prepositional complement. Whenever this happens, it is taken that the missing argument is in fact the output of the last action, in this example, the mangoes. In that way, we are able to deal with anaphora, the phenomenon wherein a different word, or no word at all as it might be, is used to represent an object.

Other types of anaphora appear in recipes. The expression "seasonings ingredients" clearly refer to some set of food components, so the ingredient hierarchy is used to find all the nodes of the tree that fit under the "seasonings" category. A phrase such as "cover with sauce" is trickier because there is no obvious clue either in the text or in the ontology as to which food the word "sauce" may refer to. We built, from the analysis of thousands of recipes, "target sets" of ingredients that usually appear in the food components being referred to by word such as "sauce" or, say, "batter". This allows for probabilist association of the word with the proper food item with respect to the ingredients of this food.

3.3 Textual adaptation

One thing having a high structured case representation of recipes is useful for is to suggest to the user an adaptation at the formal representation level, or even at the text level.

For instance, in [6], an adaptation method that consists in replacing a sequence of actions applicable to an ingredient α with a sequence more suited to a substitution ingredient β taken from another recipe was introduced. Compared with,

for instance, complex and error-prone text generation, this method is economical in terms of textual adaptation, because it makes it possible to prune a piece of text and replace it with a piece from a different text, modulo minor linguistic adjustments (fixing capitalisation and punctuation).

Whatever the case representation formalism, the effect of any adaptation method will be to modify the formal representation of the retrieved case to make it suitable as a solution to the target problem. If the solution is to be presented to the user in text format, it is necessary that these modifications be applied to the text at the same time as to its formalised form.

This requires some annotation in the text to establish a mapping between the text and its formal representation. Minimally, clause or sentence segmentation must be marked in the text, so that each clause or sentence can be mapped to the action(s) it expresses. In order simply to replace whatever α-specific preparation steps in the recipe with β-specific preparation steps, this is sufficient.

Fig. 5 shows an example of this adaptation method: the tree branch corresponding to mangoes is being pruned at the arc shown in dotted style, and a fig branch from another recipe is being grated in its place. The corresponding adaptation is done in text at the same time. The result of the textual adaptation appears in Fig. 1.

In the new representation format used in WikiTaaable, the text–formal representation mapping is materialised in the graph by one vertex associated each clause from the text. Each action vertex is connected to exactly one clause vertex with an arc labelled with `isRelatedToClause`.

4 Editing graphs using KCatoS

4.1 KCatoS

Many processes or decision makings can be represented using decision trees. Unfortunately, Semantic Mediawiki does not provide any decision tree editor. That is why KCatoS has been created. KCatoS is a semantic decision tree editor, which provides a collaborative tool to simplify knowledge acquisition. Using a simple graphical language, KCatoS allows exporting decision trees to formalised knowledge, by proposing an original algorithm export to OWL [10].

KCatoS is initially a part of a larger work about collaborative editing of clinical guidelines. KCatoS has been created as a SMW extension for Oncologik[8] [9], a semantic wiki that shares clinical practice guideline in oncology. Indeed, to represent decision making, guidelines use visual representations that can mostly be viewed as decision trees from which a meaning can be extracted.

KCatoS proposes various features. Among these features, a syntactic module can be used to check if the edited tree respects decision tree rules. Included in the interface, the module allows to validate trees step by step while drawing, by identifying shapes with mistakes. As an output, different formats are proposed: bitmap (PNG and JPG), vector graphics (SVG), and ontologies (OWL). Moreover, KCatoS includes its own versioning systems. As each tree is kept on a distant server, modifications are saved. Currently, only few functions dealing with history are available: previous versions of a tree can be viewed and restored with some information about authors and dates. However, some pieces of information are saved into XML files that will allow to add functionalities such as the comparison of versions and merging algorithms. Those improvements are planned to be integrated at a future time.

KCatoS decision tree editor is a web-based application using Google Web Toolkit[9] (GWT) that allows to create complex Ajax applications. A few additional APIs dedicated to GWT are used to manage the interface. Drawing capabilities rely on SVG and JavaScript technologies while OWL export is done thanks to OWL API [7]. Thus, KCatoS is open to collaborative work and web services. Its framework can be integrated in most of content management systems.

4.2 KCatoS for recipes

KCatoS cannot be used without modifications in WikiTaaable. One reason is the size of the graphs. KCatoS was made to edit small decision trees, whereas a recipe with i ingredients and a actions will have at least $3a + i$ vertices (each action normally has one for vertex itself, one for its output, and one for its clause).

Is is therefore necessary to present the users with features designed to handle the size and complexity of the graphs, such as the "intelligent zoom" shown in Fig. 6. When the user selects an action for which the annotation needs to be corrected, the editor can automatically centre on this action and show only relevant vertices, such as the foods known to be available for use at the time when the action is executed.

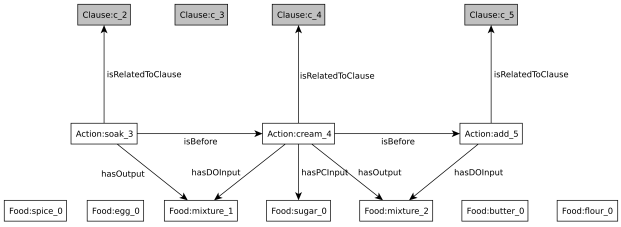

Figure 6: "Intelligent zoom" on vertex `Action:cream_4`.

Another difficulty is that one simple error in the case acquisition process can result in numerous errors down the road. For instance, considering that ingredient names are not usually repeated more than once in the text, if one input for an action was missed, it will remain missing from the input of all the following actions. Correcting all these mistakes is bound to be a time-consuming and error-prone process. For this reason, KCatoS was modified to allow for semi-automatic annotation. The user is asked to obey to the text order when correcting mistakes. In exchange, the application is capable of using the human-validated part of the annotation as an additional input and propose a new representation of the recipe, hopefully correcting all the mistakes that were caused by the original error corrected by the user.

By looking closely at the graph of Fig. 4, it can be seen that the "molten" action is missing. This causes the vertices for the ingredient "butter" and the clause `c_3` (visible in Fig. 6) to be isolated. This further causes butter to be missing from

[8] `http://www.oncologik.fr`

[9] `http://code.google.com/intl/en/webtoolkit/`

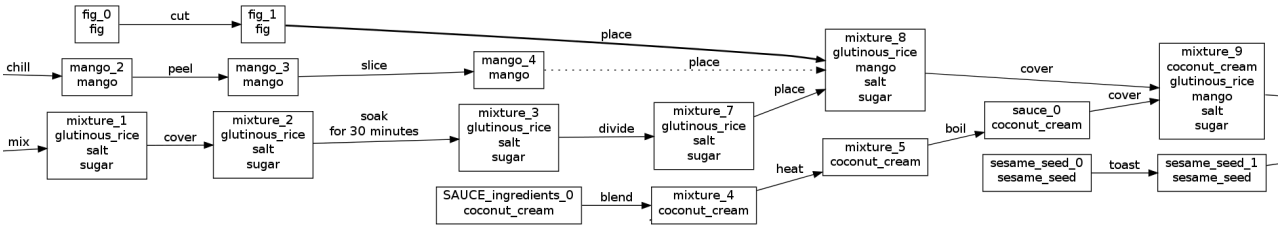

Figure 5: Adaptation by action sequence replacement: a branch related to mango preparation is pruned (dashed arc) and a branch related to fig preparation is grafted (bold arc).

the input of the action "cream", and from the remainder of the recipe. If the user simply adds the missing action and connecting it to the correct input and clause, as shown in Fig. 7, the system is able to correctly repair the rest of the graph.

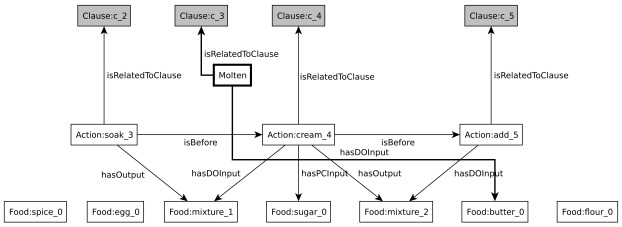

Figure 7: Adding a missing action in KCatoS.

5 Conclusion

In order to adapt recipes properly, Taaable uses a highly structured case representation for cooking recipes. This representation is extracted automatically from recipe texts, using an application based on natural language processing methods, but the results are not perfect. This paper shows how a graph editing application for semantic wikis, KCatoS, can be integrated in WikiTaaable, the semantic wiki in which the knowledge required by Taaable is stored, in order to provide wiki users with a way to correct the case representation of the cooking process, thus improving the quality of the knowledge about cooking procedures stored in WikiTaaable. Making those tools available to the public will make it possible for us to gain feedback about the usability of our proposal and its ability to generate correct annotations.

Acknowledgements

KCatoS has been conceived and developed in the context of Kasimir project, gathering the company A2ZI, the health network Oncolor, and Université de Lorraine.

REFERENCES

[1] S. P. Abney, 'Parsing by chunks', in *Principle-Based Parsing: Computation and Psycholinguistics*, ed., R. C. Berwick, volume 44, 257–278, Kluwer, (1991).

[2] I. Bichindaritz and S. Montani, eds. *Case-Based Reasoning Research and Development (ICCBR 2010)*, volume 6176, Alessandria, Italy, July 2010. Springer-Verlag.

[3] A. Blansché, J. Cojan, V. Dufour-Lussier, J. Lieber, P. Molli, E. Nauer, H. Skaf-Molli, and Y. Toussaint, 'Taaable 3: adaptation of ingredient quantities and of textual preparations.', in *Workshop Proceedings of the Eighteenth International Conference on Case-Based Reasoning (ICCBR 2010)*, ed., C. Marling, pp. 189–198, Alessandria, Italy, (2010).

[4] E. Brill, 'A simple rule-based part of speech tagger', in *Proceedings of the workshop on Speech and Natural Language*, HLT '91, pp. 112–116, Stroudsburg, PA, USA, (1992). Association for Computational Linguistics.

[5] V. Dufour-Lussier, J. Lieber, F. Le Ber, and L. Martin, 'Adapting spatial and temporal cases', in *International Conference on Case-Based Reasoning (ICCBR)*, Lyon, France, (September 2012). Accepted.

[6] V. Dufour-Lussier, J. Lieber, E. Nauer, and Y. Toussaint, 'Text adaptation using formal concept analysis', In Bichindaritz and Montani [2], pp. 96–110.

[7] M. Horridge and S. Bechhofer, 'The OWL API: A Java API for OWL ontologies', *Semantic Web*, **2**(1), 11–21, (2011).

[8] M. Krötzsch, D. Vrandecic, and M. Völkel, 'Semantic mediawiki', in *International Semantic Web Conference*, eds., I. F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, volume 4273 of *Lecture Notes in Computer Science*, pp. 935–942. Springer, (2006).

[9] T. Meilender, J. Lieber, F. Palomares, and N. Jay, 'From web 1.0 to social semantic web: Lessons learnt from a migration to a medical semantic wiki', in *ESWC*, eds., E. Simperl, P. Cimiano, Axel Polleres, Ó. Corcho, and V. Presutti, volume 7295 of *Lecture Notes in Computer Science*, pp. 618–632. Springer, (2012).

[10] T. Meilender, J. Lieber, F. Palomares, and N. Jay, 'Semantic decision trees editing for decision support with KCATOS - Application in oncology', in *ESWC Workshops, SIMI 2012 : Semantic Interoperability in Medical Informatics, Heraklion, Greece, May 27-28, 2012, To be published*, (2012).

[11] M. Minor, R. Bergmann, S. Görg, and K. Walter, 'Towards case-based adaptation of workflows', In Bichindaritz and Montani [2], pp. 421–435.

[12] C. K. Riesbeck and R. C. Schank, *Inside Case-Based Reasoning*, Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey, 1989.

[13] P. Schumacher, M. Minor, K. Walter, and R. Bergmann, 'Extraction of procedural knowledge from the web: A comparison of two workflow extraction approaches', in *Proceedings of the 21st international conference companion on World Wide Web*, WWW '12 Companion, pp. 739–747, New York, NY, USA, (2012). ACM.

ACook: Recipe adaptation using ontologies, case-based reasoning systems and knowledge discovery

Sergio Gutiérrez Mota and **Belen Diaz Agudo** [1]

Abstract.

This paper presents ACook, a program that addresses the problem of adapting the ingredients of a cooking recipe using a variety of techniques like ontologies, CBR systems and knowledge discovery. Using this technologies, we developed three different versions of the tool: Onto-ACook that only uses an ingredient ontology, CBR-ACook which is a case-based-reasoning system, and AKD-ACook based on adaptation knowledge discovery. An adaptation example is used to evaluate the results of the three subsystems and the conclusions are presented. Finally we propose some improvements for each version of the program.

1 Introduction

In the last years of artificial intelligence a lot of tools have been developed in order to solve problems which are increasingly complex. This includes the use of ontologies to formalize knowledge in a particular domain and reason about it. The development of case-based reasoning systems provides a model which we can use to solve problems using older solutions stored in a case-base. Finally, knowledge discovery is a field that describes methods to extract knowledge of a database automatically.

This paper describes ACook, a tool that addresses the challenge of adapt a cooking recipe using specific constraints over the ingredients.

This paper presents and compares three different ways to approach the problem of adapting cooking recipes. The first system, Onto-ACook, works using an ontology of ingredients and only takes into account each banned ingredient separately. The second version, CBR-ACook, uses a case-based reasoning system which needs an expert to add the knowledge of the domain into the program. The last implementation, AKD-ACook, extracts automatically ingredient adaptation patterns using a recipe database and uses them to build a consistent solution ,i.e., one completely based on an unique recipe.

Finally, this document compares the three different systems and outlines their strengths and weaknesses in order to suggest improvements for each one.

2 Technologies

In this section, we present the more basic concepts of the technologies used in ACook and how are being used in the program.

2.1 Ontologies

An ontology is a representation of knowledge into a concrete domain. It contains a set of concepts and the relationships between those concepts. It is used to reason about these concepts and extract properties about the entities within the domain. Figure 1 shows the hierarchy of the main ontology used in ACook that links ingredients by their origin.

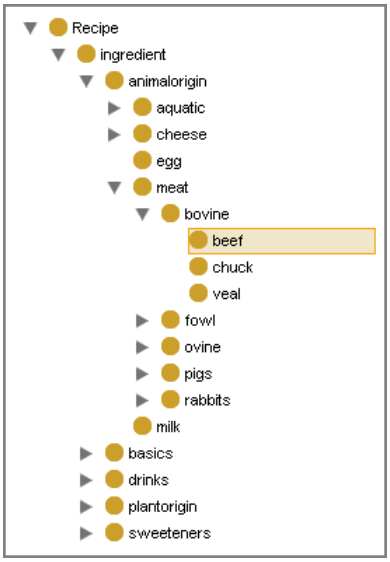

Figure 1: Ingredients ontology used in ACook

2.2 Case-Based Reasoning

Case-Based reasoning (CBR) is a model used to solve problems of artificial intelligence. The main component of the CBR systems are the **cases**, which are a description of a particular problem and its solution. Given a new description of a problem, a CBR system offers a solution based on the similarity to already solved problems stored in the system. The main difference between CBR systems and rules-based systems are that in the first ones it is not needed to formalize the knowledge from an expert, it is just needed to fill the program with descriptions of previous events.

Figure 2 shows an example of a case used in ACook to adapt recipes. Each case consists of the description of the problem (recipe name, used ingredients, preparation and banned ingredients) and its solution. The details of the cases are explained in section 3.2.

2.3 Knowledge Discovery

Knowledge Discovery (KD) is a large field in computation that describes methods to extract knowledge by searching patterns in large

[1] School of Computing. Complutense University of Madrid. Spain, email:belend@sip.ucm.es

Case	
Recipe	*Roasted peaches with brown sugar and sherry*
Ingredients	Butter Peaches Sherry Brown Sugar
Preparation	Cut nectarines in half, remove stones and place, cut side up, on baking sheet. Divide butter among halves, placing little in each cavity. Add 1 teaspoon each Sherry and brown sugar to cavity of each nectarine half. Broil 4 to 6 inches from heat source until topping is bubbly and fruit softened, about 5 minutes.
Banned ingredients	Brown Sugar
Solution	Sugar+

Figure 2: Case example of the CBR system used in ACook

databases. The techniques included in KD depends strongly on the domain of the processed data and the way the programmer wants to show the solutions. The most used branch of knowledge discovery is the one used in databases (KDD) which defines a number of steps to get the knowledge from the data: selection, pre-processing, transformation, data mining and interpretation/evaluation.

ACook uses KD [10] to adapt cooking recipes as it is shown in section 3.3. To understand how it works we need to introduce the basics and some new concepts.

We define a formal context K as a tuple of the form (G, M, r), where G is a set of objects, M is a set of items and r is the relation $G \times M$ meaning that an object is described by an item.

An itemset I is defined as a subset of items of M. Also, the support of I, or support(I) is the number of objects of G having every item of I. Finally, I is called frequent whenever his support is larger than a threshold σ and it is called closed whenever it has no proper superset J with the same support. The idea behind the frequent closed itemsets (FCIs) is that FCIs are the biggest patterns (with more items) used in larges groups of objects.

Figure 3 shows a table with an example of formal context. G is the set of recipes and M is the set of ingredients. Every check in the table means that the recipe uses the ingredient. The only FCI with support 4 is {Onion, Oil}, in other words, there are 4 recipes including onion and oil.

3 Implementation

The following sections presents different versions of ACook and how they use the technologies in the preceding section. In order to make things clear we use the same adaptation query for the three systems: Adapt the recipe called *Greens with Vinaigrette* which uses the ingredients {Lettuce, Vinegar, Pepper, Onion, Oil, Salt, Sugar} and with the following banned ingredients: {Lettuce, Onion}.

Figure 4 shows the application interface, in the first menu the user can choose a recipe from the list and in the second one the user can

	Lettuce	Vinegar	Pepper	Onion	Oil	Salt	Sugar	Garlic	Parmesan	Cherry	Lemon juice	Chicken
Recipe 1	X	X	X	X	X	X	X					
Recipe 2	X	X		X	X	X		X	X			
Recipe 3	X		X	X	X	X			X	X	X	
Recipe 4			X	X	X		X				X	X

Figure 3: Example of formal context with recipes and ingredients

check the banned ingredients. Each button from the bottom image executes one of the three adaptation subsystems that are described in the following sections.

3.1 Onto-ACook

The first version of ACook uses an ontology of ingredients to build the best adaptation for a recipe. The ontology is an extended version of the one used in JaDaCook [2]. It forms an ingredient hierarchy that contains 221 ingredients arranged in 54 classes by its source and type. The ontology does not take advantage of other features like attributes or restrictions in order to simplify its construction. Figure 1 shows the ontology used in ACook. The algorithm 1 shows how Onto-ACook adapts a recipe using the ontology. For each banned ingredient, the system searches between its closer brothers in the ontology a valid substitution (not already banned or used) and chooses one randomly. When we query the system with the example recipe, *Greens with Vinaigrette*, it informs us that we can make the following replacements:

- Add {Romaine, Asparagus}
- Remove {Lettuce, Onion}

Onto-ACook only gives simple substitutions, where an ingredient can only be replaced by another one and does not take into account more complex possibilities like removing or replacing other non banned ingredients in order to build a better recipe. This is because the substitution system only consider each ingredient separately and does not know anything about the recipe it is used in. In other words, it is unable to handle combinations of ingredients. Furthermore, this version of ACook is very ontology dependent, therefore, the larger and complex the ontology is, the better solutions that Onto-ACook offers. In addition, there is other features like ingredients flavor or quantities that are impossible to capture in ontologies and therefore in Onto-ACook.

3.2 CBR-ACook

The second version of ACook has been fully developed with JColibri [3], a Java framework to create CBR systems. Each case, as shown in figure 2, has the description of a recipe (its name and a list of used ingredients) and a list of banned ingredients. Proposed solutions by CBR-ACook are a list of ingredients that should be added (written with a $+$ symbol) and a list of ingredients that should be removed (written with a $-$ symbol) from the original recipe. It is assumed that banned ingredients are already chosen to be removed from the original recipe so they are not shown in the solution. This new way of expressing solutions gives us more complex adaptations since we can remove ingredients which have not been banned or add more than one ingredient for each substitution.

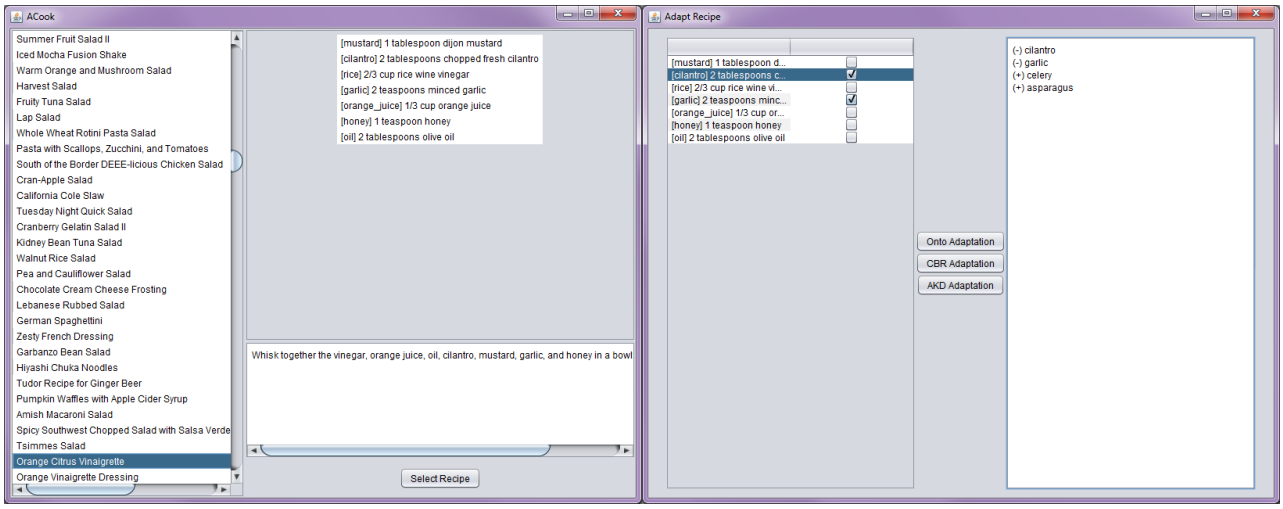

Figure 4: ACook gui

Algorithm 1 Adaptation of Onto-ACook

```
function ONTOACOOK-ADAPT(banned_ingredients)
    for all ing ∈ banned_ingredients do
        pad ← ontology.getParent(ing)
        replaced ← false
        while not replaced do
            search for a valid substitution (not already banned or used) in pad parents and updates the replaced one
            if not replaced then
                pad ← ontology.getParent(pad)
            end if
        end while
    end for
end function
```

Algorithm 2 shows how the substitution of CBR-ACook works. One of the keys of CBR systems is the function used to compute the similarity between cases, CBR-ACook does the mean between two values: the similarity between the list of used ingredients and the list of banned ingredients. The similarity between lists of ingredients uses the *deep basic* [6] algorithm, that returns a similarity value of two ingredients according to the distance between them in an ontology. The similarity function is based on the ontology used in Onto-ACook. Algorithm 3 shows a simplified version of the similarity function used to compare lists of ingredients.

Algorithm 2 Adaptation of CBRACook

```
function CBRACOOK-ADAPT(recipe, banned_ingredients)
    query ← recipe + banned_ingredients
    cbr_cycle(query)
    return most_similar_case_solution
end function
```

When we query CBR-ACook with the example recipe, the system suggests the following solution based on previous cases filled by an expert:

- Add {Romaine, Endive, Scallion}
- Remove {Lettuce, Onion}

Algorithm 3 Similarity function of CBRACook

```
function CBRACOOK-SIMILARITY(query_ingredients, case_ingredients)
    similarity ← 0
    for all query_ing ∈ query_ingredients do
        max_similarity ← 0
        for all case_ing ∈ case_ingredients do
            max_similarity ← max(max_similarity, deep basic(query_ing, case_ing)
        end for
        similarity ← similarity + max_similarity
    end for
    return similarity / number_of_query_ingredients
end function
```

As we can see, this solution is more consistent that the one suggested by the first version of the tool, this is because Onto-ACook only uses each ingredient separately, however, CBR-ACook uses the recipe as a complete entity. Furthermore, the solution contains three new ingredients and only two removed ingredients. This version of the system returns a good adaptation because the expert responsible for filling the cases only worked with salads. It was unfeasible to obtain a complete set of cases for all the recipes and this is one of the main weakness of this type of tools.

3.3 AKD-ACook

The last version of ACook rely on KD techniques applied to adaptation and is completely based on the tool TAAABLE [5]. In order to understand the way AKD-ACook works is necessary to introduce the concept of **recipe variations**. Given two recipes (R_i, R_j), the variations of R_i y R_j are a set of tuples $(\text{ingredient}, \text{modifier})$ called **ingredients variations**. There is only three valid modifiers and its meaning is:

- (*ing*,+) *ing* is used in R_j but not in R_i.
- (*ing*,−) *ing* is used in R_i but not in R_j.
- (*ing*,=) *ing* is used in both R_i and R_j.

Using the two following recipes:

- R_i = {Lettuce, Vinegar, Pepper, Onion, Oil, Salt, Sugar}
- R_j = {Lettuce, Vinegar, Onion, Oil, Salt, Garlic, Parmesan}

variations of (R_i, R_j) are {(Lettuce,=), (Vinegar, =), (Onion, =), (Oil, =), (Salt, =), (Pepper, −), (Sugar, −), (Garlic, +), (Parmesan, +)}. The idea behind this new concept is that the variations of (R_i, R_j) contains the changes that are needed in order to transform R_i into R_j.

AKD-ACook uses the formal context $K = (G, M, r)$ where the set G contains tuples (R, R_i), being R the recipe we want to adapt and each R_i are the other recipes stored in the system. The M set contains every possible ingredient variation and r links G objects with M items following the definition of recipe variations. Figure 5 shows a simple formal context used in AKD-ACook.

As we saw in section 2.3, pattern extraction is the same as FCI extraction. Because this approach is computationally expensive it is common to filter the data in order to handle smaller groups of recipes. In AKD-ACook, the filtering is applied to the recipes of the database which does not satisfy some criteria. Particularly, the recipes used in the FCI extraction process are the ones which satisfy the following tests:

- Does not use any of the banned ingredients.
- Share 50% of ingredients with the original recipe.
- There is no more than 50% of added ingredients compared with the original recipe.

The purpose of this tests is to exclude recipes that are not similar enough to the original one, increasing the amount of potential substitutions that can be considered valid from a cooking point of view.

Using the example recipe and thanks to the filtering process, the number of used recipes in the FCI extraction is reduced from 3385 to 70 and the number of ingredient variations is also reduced from 206 to 51. Once the system has filtered the recipes it creates the formal context and calls the CORON software platform [4] in order to extract FCIs. After this, the program sorts the patterns using five different criteria listed by priority:

1. FCIs with at least one added ingredient.
2. FCIs with more shared ingredients with the original recipe.
3. FCIs with less removed ingredients compared to the original recipe.
4. FCIs with less added ingredients compared to the original recipe.
5. FCIs with higher support.

Finally, the system computes the recipes responsible for the most valued FCI (the ones that use all the ingredients of the pattern). It is needed to use these recipes because the computed FCI is just a pattern of the variations between the original recipe and the others stored in the database, therefore, we cannot guarantee that it forms a consistent recipe. Each recipe involved in the best FCI creates a list of adaptation rules which are displayed to the user. Algorithm 4 shows a simplified version of how the adaptation system of AKD-ACook works.

When the system adapts the example recipe it creates 78 FCIs and chooses the following one as the best valued: {(Vinegar, =), (Pepper, =), (Sugar, =), (Salt, =), (Oil, =), (Celery, +), (Lettuce, −), (Onion, −)} which have a support of 3. The three recipe variations involved in the extraction of the FCI are shown in figure 5. Any of these variations can be used to build the substitution rules and AKD-ACook chooses one of them randomly. In this case it shows the following solution to the user:

Algorithm 4 Adaptation of AKDACook

```
function AKDACOOK-ADAPT(recipe, banned_ingredients, recipes)
    filtered_recipes ← filter(recipes)
    formal_context ← create_formal_context(recipe, banned_ingredients, filtered_recipes)
    fcis ← compute_fcis(formal_context)
    sort(fcis)
    used_recipes ← get_used_recipes(fcis)
    final_recipe ← get_random(used_recipes)
    return build_substitution_rules(final_recipe)
end function
```

- Add {Mustard, Cabbage, Celery}
- Remove {Lettuce, Onion}

Watching the solution obtained with AKD-ACook we can see that the substitution is completely valid because it used real recipes to generate the final adaptation. Furthermore, the solutions of this version of the system are as expressive as the ones computed with CBR-ACook, there can be more than one added ingredient for each banned ingredient. The main advantage of this system compared with the second version is that we did not need an expert responsible for creating the knowledge of the recipes domain since it was obtained automatically. Even the large recipe database was created using a simple program to parse some web pages with all the content. It is also important to note that even using recipes of other types –not only salads– the filtering and sorting steps decrease the possibilities of obtaining misleading results. Finally, it is essential to keep in mind that the adaptation is based on one unique recipe but the choice of this recipe is not made comparing the number of shared ingredients but using the knowledge extracted from the recipe database.

	(Vinegar, =)	(Pepper, =)	(Sugar, =)	(Salt, =)	(Oil, =)	(Mustard, +)	(Dried, +)	(Cabbage, +)	(Celery, +)	(Clove, +)	(Lettuce, -)	(Onion, -)
(R, Recipe 1)	X	X	X	X	X	X		X	X		X	X
(R, Recipe 2)	X	X	X	X	X				X		X	X
(R, Recipe 3)	X	X	X	X	X		X		X	X	X	X

Figure 5: Formal context that contains the recipes used in the best extracted FCI for the example query

4 Conclusions and related work

This paper outlines how the techniques of artificial intelligence are very effective in adaptation problems. With ACook we can see how these techniques can be applied to the same problem and shows the strengths and weaknesses of each one.

It is also sketched that the knowledge about the domain that the system is able to handle is a key point. In Onto-ACook the knowledge was restricted to the ingredients, therefore its recommendations were simpler. CBR-ACook use more knowledge, is able to handle recipes as complete entities, however, needing an expert to complete the knowledge of the system is inviable if we are using a domain as big as the used in this project. The last version of the program, AKD-ACook, is also able to handle recipes as complete entities, therefor its solutions are more complex compared to the first version, but unlike CBR-ACook it is not needed an expert to get all the knowledge, instead, it is extracted automatically.

Each one of the systems can be improved, the ontology can be completed, filling it with more ingredients either adding more attributes to every ingredient. Other works [5] show how the use of several hierarchies is also a viable way to improve the solutions obtained. The CBR system could be improved adding some module to process natural language in order to extract knowledge of the preparation of each recipe. It is also possible to categorize recipes in families so the CBR engine does not have to compare the query with every recipe in the casebase. The last version of the system, AKDACook, can use an extended formal context using the ingredients ontology in order to link ingredients, e.g. ingredients (Lettuce,$-$) and (Romain,$+$) entail the variation (Vegetable,$=$).

REFERENCES

[1] Emmanuelle Gaillard, Jean Lieber, Emmanuel Nauer. Adaptation knowledge discovery for cooking using closed itemset extraction. In *The Eighth International Conference on Concept Lattices and their Applications - CLA 2011*, pages 87-99, Nancy, France, Oct. 2011.

[2] P. Javier Herrera, Pablo Iglesias, David Romero, Ignacio Rubio, Belén Díaz-Agudo. JaDaCook: Java Application Developed and Cooked Over Ontological Knowledge. In Schaaf [Sch08], pages 209-218, 2008.

[3] JColibri: Case-Based Reasoning Framework. *http://gaia.fdi.ucm.es/research/colibri/jcolibri*.

[4] L. Szathmary y A. Napoli. CORON: A Framework for Levelwise Itemset Mining Algorithms. *Supplementary Proc. of The Third International Conference on Formal Concept Analysis (ICFCA '05), Lens, France*, pages 110-113, 2005.

[5] F. Badra et al. Taaable: Text Mining, Ontology Engineering, and Hierarchical Classification for Textual Case-Based Cooking. In *ECCBR Workshops, Workshop of the First Computer Cooking Contest*, pages 219-228, 2008.

[6] J. A. Recio-García, B. Díz-Agudo, P. A. González-Calero, and A. Sánchez-Ruiz-Granados. Ontology based CBR with JColibri. In R. Ellis, T. Allen, and A. Tuson, editors, *Applications and Innovations in Intelligent Systems XIV. Proceedings of AI-2006, the Twenty-sixth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 149-162, Cambridge, United Kingdom, December 2006. Springer.

[7] Ribeiro, R., Batista, F., Pardal, J. P., Mamede, N. J., and Pinto, H. S., Cooking an Ontology. In *Artificial Intelligence: Methodology, Systems, and Applications*, ser. Lecture Notes in Computer Science, no. 4183. Rua Alves Redol, 9: Springer Berlin / Heidelberg, pp. 213-221, Sep. 2006.

[8] F. Badra, A. Cordier, and J. Lieber. Opportunistic Adaptation Knowledge Discovery. In Lorraine McGinty and David C. Wilson, editors, *8th International Conference on Case-Based Reasoning - ICCBR 2009*, volume 5650 of *Lecture Notes in Computer Science*, pages 60-74, Seattle, États-Unis, July 2009.

[9] Aamodt, A. and Plaza, E. Case-Based Reasoning: Foundational Issues, Methodological Variations, and Systems Approaches. In *AI Communications* 7, pp. 39-59, 1994.

[10] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. AI Magazine, pages 37–54, 1996.

Knowledge Acquisition with Natural Language Processing in the Food Domain: Potential and Challenges

Michael Wiegand and **Benjamin Roth** and **Dietrich Klakow**[1]

Abstract. In this paper, we present an outlook on the effectiveness of natural language processing (NLP) in extracting knowledge for the food domain. We identify potential scenarios that we think are particularly suitable for NLP techniques. As a source for extracting knowledge we will highlight the benefits of textual content from social media. Typical methods that we think would be suitable will be discussed. We will also address potential problems and limits that the application of NLP methods may yield.

1 Introduction

Food plays an essential role in each of our lives. We do not only need it to survive but it has also significant social and cultural aspects. Within the last fifty years, research in artificial intelligence (AI) has brought immense achievements for human society with the result that, nowadays, AI technology is available in many parts of our life. Due to the importance of food in our society and the general applicability of AI methods, it is only a natural consequence that research in the area of AI has also addressed tasks in the food domain.

In this paper, we focus on one specific branch in artificial intelligence, namely natural language processing (NLP). NLP can be defined as the task of extracting meaningful content from natural language utterances. Research in artificial intelligence addressing food-related tasks up to now focused on human-computer interaction [2, 4, 11, 13, 14, 16, 17, 22], knowledge engineering [1, 7] and image/video processing [21, 28]. Since there has only been very little research examining the usefulness of NLP in tasks related to the food domain, we outline some directions of research that given the current state of the art we envisage to yield some potential. More precisely, we want to describe some scenarios in which NLP can be leveraged in order to extract *knowledge*. The resulting roadmap is the main contribution of this paper.

The basic task that all these scenarios involving NLP underlie is the conversion of some written natural language text, i.e. some unstructured data, to some structured text. For example, a text, such as Sentence 1, should be transformed to some relation (similar to a logic formula), such as Example 2.

1. I use shortcrust pastry for my apple pie.
2. *Ingredient-of(shortcrust pastry, apple pie)*

It is then the task of other disciplines, such as knowledge engineering, to incorporate these data into an information system that supports a user in their decision making. (This step will not be covered in this paper.) In this paper, we exclusively focus on knowledge extraction from written text. This work expands on our preliminary findings presented in [29] which describes empirical work of knowledge extraction in the food domain for German.

[1] Spoken Language Systems, Saarland University, Germany, email: Michael.Wiegand, Benjamin.Roth, Dietrich.Klakow@lsv.uni-saarland.de

2 The Main Purpose of Artificial Intelligence in the Food Domain

If one categorizes existing research in artificial intelligence dealing with food according to their purposes, one actually finds that most of them serve the same purpose. This research proposes technologies that attempt to fix some undesirable behavior. In [12], such methods are called *corrective technologies*. Much of previous work supports a user in cooking a meal [13, 14, 22, 25]. In these cases, the undesirable behavior can be considered the uncertainty or inexperience of how to prepare a meal. Another line of research deals with supporting people with health-related problems, e.g. following a specific diet [2, 4, 11, 17]. In these works, the undesirable behavior can be considered some improper diet. Some of previous work may not seem to address the fixing of an undesirable behavior, but in most of these cases they *indirectly* address this issue as some intermediate problem is solved. For example, the task of detecting how much food has been consumed from a plate [16], the task of analyzing drinking activity [28] or mastication [21] can be seen as intermediate steps that need to be dealt with in order to fully support humans in performing a diet.

We assume that a viable task in the food domain in which NLP can be applied should also address the fixing of some undesirable behavior. We even think that those two major scenarios presented above (i.e. *preparing a meal* and *following a health-related diet*) are actually ideal scenarios for applying NLP methods. We will motivate this in detail in Section 5.

3 Benefit of State-of-the-art NLP in General

With today's hardware capacity, a prominent advantage of NLP is that it can process text at a speed that significantly exceeds human performance and hence larger amounts of texts can be processed.

The type of information that can be extracted is usually restricted to content that can be detected with the help of some surface patterns. Surface patterns usually comprise lexical knowledge, but it may also include syntactic and semantic knowledge. We will illustrate this with an example. For a relation instance, such as *Can-be-Substituted-with(butter, margarine)*, there are many different ways of how this relation instance can be expressed in natural language text as exemplified by Sentences 3-5.

3. I use margarine instead of butter.
4. Butter is often substituted by margarine.
5. For the apple pie we used margarine; I forgot to buy butter at the supermarket.

Sentences 3 and 4 can be recognized with the help of NLP. This will be explained in more detail in Section 6.3.2. To infer this relation from Sentence 5, however, extra-linguistic (pragmatic) knowledge would be required as there are no explicit lexical cues indicating the given relation. The food items *margarine* and *butter* are in different clauses and there is no syntactic relationship between the words that could indicate some relation. Only by knowing that having forgotten butter at the supermarket is a justification of using margarine, one can infer that the speaker would normally have taken butter. From this we can conclude that butter and margarine can be exchanged with each other. This is some domain-specific knowledge that is extremely hard, if not impossible, to acquire. In other words, with state-of-the-art NLP it is not possible to fully comprehend an entire text. A deeper understanding of text can only be obtained if an ontology encoding word knowledge complements the linguistic analysis. Such endeavours only work for extremely closed-domain scenarios. Moreover, they are much less efficient.

4 Social Media as the Source for Natural Language Processing

When using NLP for a new domain, one also needs to answer what text source should be used for extracting content. Of course, not any arbitrary text source is applicable. In order to qualify as a source, the text type needs to meet the two following criteria:

Firstly, the text type needs to contain sufficient domain knowledge. In other words, if we choose a text type that only infrequently contains content regarding food, then we are not very likely to extract any significant amount of knowledge. In the past, most research in NLP has been carried out on news corpora [15]. The topic that is predominant on this text type are political affairs rather than food-related issues. Consequently, this text type would be of little value for knowledge extraction of food relations.

Secondly, the text type should not only contain knowledge about food that is already widely available in structured format (such as databases). Otherwise, there would hardly be any point in extracting knowledge from those texts as it would already be available.

Given these requirements, we argue that one particularly promising text type for the extraction of food-related knowledge are *social media*. By social media, one understands those types of media (not necessarily only textual data) that allow some interaction between the people who produce information and the people who consume it. Moreover, in social media the same person can assume both of these roles; a recipient of some information can be the producer of some other information in a different situation. The person who produces content can be any arbitrary *user*. This has led to the coinage of the term *user generated-content*.

The texts from the social-media domain that we are primarily interested in are *internet forums* and *weblogs*. Apart from the fact that large amounts of such texts are actually publicly available, e.g. they can be downloaded via web crawlers, there is also a significant proportion dealing with food-related issues. This is because food is a central issue in everyday life and, nowadays, almost every part of everyday life is reflected in social media in some way or the other. Furthermore, we assume that the kind of food-related information that can be found in social media is, to a large extent, complementary to what is found in existing resources. (In Section 6.2, we will give a typical example of the type of available resources that contains information regarding food items.) The reason for that is that existing resources contain (uncontroversial) factual content regarding food items. For instance, there are ontologies that arrange food items in a hyponymy (*is-a*) relationship. (Thus one can read off which food items are a subtype or supertype of another item.) Social media, on the other hand, also contain much subjective information. On the web, users may not only exchange recipes but also discuss which combination of food items they *think* is appropriate or which items can be used instead of each other. In addition to that, they may also exchange their *experience* with certain types of food, in particular, if they are on a diet or have certain health conditions (such as *allergies*, *diabetes* or *irritable bowel syndrome*).

In our first preliminary work on knowledge acquisition in the food domain [29], a crawl from an internet forum has successfully been used. Since that work is done on German, the largest German website dealing with food-related issues, namely *chefkoch.de*[2], has been crawled. The resulting data collection comprises $418,558$ webpages.

In the following section, we show that the information potentially contained in these data (described above) would also be extremely valuable for real-life scenarios.

5 Scenarios

One possible scenario that could make use of NLP and that also motivates our previous work in the food domain [29] is virtual customer advice. We will now describe this setting and highlight what benefits NLP would bring about in this task. Moreover, we will also outline possible extensions to this scenario. We will focus on this single scenario because it has many interesting facets that yield many possibilities of applying different NLP methods. Moreover, this scenario has an obvious commercial potential. Commercial viability is important for many new technologies to be developed, as it may foster cooperation between academia and industry.

The specific use case that is presented in [29] is assisting a customer in a supermarket in doing their shopping. Typical situations that could arrive are that

a) a customer wants to organize an event and does not know what food items or dishes are appropriate for that occasion;
b) a customer wants to prepare a meal but does not know what ingredients are necessary;
c) a customer wants to purchase a product that is currently out of stock and does not know what suitable substitutes there are;
d) a customer has a certain health disposition (e.g. may be suffering from diabetes) and does not know which products are most suitable for them.

All these cases are typical everyday life situations, all of which exhibit a user need that cannot be immediately satisfied by information that is available in a supermarket.[3] In principle, these problems could be solved by a large knowledge base containing relevant relations. For a), a relation table listing food items for diverse events would be required; for b), it would be a list of ingredients of different dishes; for c), it would be a table containing pairs of food items that can be substituted with each other; and for d), it would be a table listing food items recommended for people with a particular health disposition. Social media cover those everyday-life problems but, to a large extent, this information is only available in unstructured natural language text (e.g. as entries in an internet forum) rather than structured relation tables. Since we already stated in Section 3 that the speed of processing natural language text with NLP software can

[2] `www.chefkoch.de`

[3] Of course, a shopping assistant could be consulted but most supermarkets will not have sufficient human resources to assist every customer with their individual problems.

significantly exceed human performance, the choice of using NLP on extracting this knowledge from those weblogs or internet forums seems self-evident.

While [29] focuses on extracting structured knowledge, we also think that it would also be worthwhile providing entire (natural language) text passages in which particular relations have been found. The resulting applications may not be necessarily linked to the shopping scenario mentioned above, though. As already outlined in Section 2, health-related issues play a major role when it comes to the topic of food. Instead of building applications that just contain the knowledge of what types of food are recommendable for people with a certain disposition or the information about which food items are healthy or unhealthy, we assume that providing contextual information might be beneficial in several respects. Contextual information helps a user to understand how a system has arrived at some specific information. Thus, a user gains some trust in the knowledge-extraction system. In the ideal case, the context actually provides some explanation or justification for a specific claim. This additional information is in particular important if a claim that has been found is controversial or, at least, not immediately comprehensible. For instance, if a system extracts the knowledge *Is-Healthy(chocolate)*, a user would remain unsatisfied with this unexpected claim unless they are given some further background information as Sentence 5 does.

5. Chocolate is healthy because it's high in magnesium and provides vitamin E and vitamin B.

In particular, recent advances in shallow discourse processing might help to retrieve those passages which do not only contain a specific relation but also some justification [27] for it.

6 Methods

We will now describe a generic architecture which needs to be implemented in order to extract the type of knowledge from the food domain that we previously described. This architecture (illustrated in Figure 1) is a generalization of the system presented in [29].

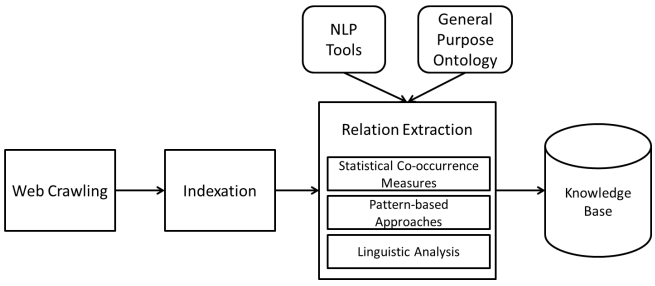

Figure 1. Generic architecture for knowledge acquisition using NLP.

6.1 Creating an Offline Index

In order to extract knowledge for the type of scenarios that we presented in Section 5, text processing needs to be carried out on large amounts of data, i.e. texts comprising several million words. Texts must first be assembled from the web. For such a purpose, publicly available web crawlers, such as *Heritrix* [23], can be used. Processing these text documents in a naive way (e.g. iterating through all files line by line) is not feasible as it would take too much time to complete the process. Imagine, a system is asked to find evidence for *Is-Healthy(broccoli)*. The first step would be finding passages (or sentences) in which the two linguistic entities *broccoli* and *healthy* co-occur. In order to obtain such text passages in real time, the text documents need to be converted into an *index data structure* that allows for efficient retrieval. For example, a widely used toolkit that carries out such a conversion and also enables the retrieval of text passages using that representation format is *Lucene* [18]. The algorithms that these tools implement are conceptually very similar to web search engines, such as *Google*, but these tools can be very flexibly tailored to a specific application. For example, one can determine how the index representing the data collection is going to be arranged. Moreover, much more sophisticated queries can be formulated in order to retrieve specific text passages.

6.2 Resources for Detecting Relevant Entities

Even though we want to extract knowledge from textual data, we also need some initial knowledge about our task domain. For instance, if we want to extract the knowledge of what types of food are usually consumed at a particular event, one needs to know the set of possible events and a list of all kinds of food. The most appropriate way to obtain such information is by incorporating general-purpose ontologies. For English, for example, one could make use of *WordNet* [19] which is a lexical database that lists semantic relations, such as *hyponymy* (*is-a* relation) or *meronymy* (*part-of* relation). These relations are not formulated between words but concepts which are groups of words with a similar meaning, i.e. *synonyms*. To obtain all words denoting food items one merely has to collect the words associated with the concepts that are hyponyms of *food*. The advantage of using such ontology instead of a mere list of food items is that it allows some inferences which might be useful for knowledge extraction. Imagine, for example, one is able to extract from text the knowledge *Suits-to(cheese, picnic)* (i.e. cheese is an appropriate type of food that can be consumed on a picnic). From this knowledge, one could also derive that this information also holds for a particular subtype of cheese, e.g. *cheddar*. Moreover, there can also be situations in which the knowledge of synonyms is beneficial. For instance, if the knowledge *Can-be-Substituted-by(zucchini, eggplant)* is extracted (i.e. zucchini is a good substitute for eggplant), and a user wants to obtain substitutes for *aubergine*, then the knowledge from an ontology that *aubergine* and *eggplant* refer to the same type of vegetable helps to infer that *Can-be-Substituted-by(zucchini, aubergine)*.

6.3 Relation Extraction

Once a text passage has been found in which two different target entities, for instance, the two food items *zucchini* and *aubergine* occur, one further needs to determine whether a particular relation holds between those items (e.g. *Can-be-Substituted-by(zucchini, aubergine)*). This is the task of *relation extraction*.

6.3.1 Statistical Co-occurrence

The simplest way to establish a relation is by measuring the statistical co-occurrence of entities between which there is potential relation. Imagine, for instance, we want to extract the food items that

are typically consumed at a given event, i.e. *Suits-to(FOOD-ITEM, EVENT)*. One possible way of extracting that knowledge is by measuring for each possible event which of the entire set of food items co-occurs with it. The more often two expressions co-occur with each other, the more likely there holds a specific relation between them. For example, *roast goose* will more often co-occur with *Christmas* than *banana* will co-occur with it, as *roast goose* is a dish typically consumed at *Christmas*. The strength of co-occurrence can be determined by applying standard measures, such as *point-wise mutual information* [6]. Statistical co-occurrence is particularly suitable for extracting relations which are difficult to grasp by means of textual patterns. For instance, in [29] it was found that lexical cues or phrases (see Section 6.3.2) to indicate the relation type *Suits-to* (e.g. cues of the form *X is usually consumed/eaten at/on (event) Y*) were less effective than co-occurrence measures. In particular, if the entities involved in a relation do not appear close to each other, a statistical co-occurrence method may be suitable.

The major shortcoming of this method is that it is completely oblivious of the context in which the entity pairs appear. This is, in particular, insufficient if there can be more than one relation holding between an entity pair. For example, if we applied this very method in order to extract relation instances of the type *Can-be-Substituted-by(FOOD-ITEM, FOOD-ITEM)* as in *Can-be-Substituted-by(fish fingers, fish cake)*, then this would mean that we would have to consider all potential pairs of food items. Unfortunately, a frequent co-occurrence between two food items does not necessarily mean that this particular relation type, i.e. *Can-be-Substituted-by*, holds between those items. This is because there could also be another relation holding between these two items, e.g. *Can-be-Served-with(FOOD-ITEM, FOOD-ITEM)* as in *Can-be-Served-with(fish fingers, mashed potatoes)*. With regard to an entity pair of type <*FOOD-ITEM, EVENT*>, there is actually only one likely relation type, namely *Suits-to*. Therefore, in order to decide whether there holds such a relation between a specific event and a specific food item, one just needs to measure the degree of co-occurrence. For extracting relations, such as *Can-be-Substituted-by(FOOD-ITEM, FOOD-ITEM)* or *Can-be-Served-with(FOOD-ITEM, FOOD-ITEM)*, on the other hand, more complex processing involving a context-based analysis is required.

6.3.2 Pattern-based Approaches

As motivated in the previous section, for some relation types, more context-aware methods, so-called *pattern-based* approaches, are necessary in order to extract instances from text.

One obvious solution to obtain such patterns is by manually writing them as it has been done in [29]. The advantage of this acquisition method is that it usually yields very precise patterns. The disadvantages are that the patterns are expensive to produce as they require expert knowledge and, moreover, tend to have a limited coverage. Of course, by considering levels of representation going beyond the mere lexical surface form (as in our preliminary work [29]) and using some linguistic resources, one could achieve some generalization. For example, consider the simple sequential surface pattern an expert may come up with, such as *replace X by Y*, in order to extract relation instances of type *Can-be-Substituted-by(FOOD-ITEM, FOOD-ITEM)*. This pattern would match Sentence 6. Sentences 7 and 8, on the other hand, would be missed.

6. You can replace butter by margarine.
7. Butter is often replaced by margarine.
8. Butter is often substituted by margarine.

With more sophisticated levels of representation that are available by state-of-the-art technology, these two sentences could also be matched. By using a pattern that does not only employ lexical information but also syntactic information, such as $X \uparrow$`logical-object` *replace* $\downarrow$`by-object` Y, Sentence 7 could also be matched. This pattern normalizes constructions, such as passive voice. The pattern says that the relation holds between *X* and *Y* if there is the verb *replace* and its logical object (this corresponds to the direct object in Sentence 6 but to the syntactic subject in Sentence 7 – both constituents are *butter*) is *X*, while its by-object is *Y* (this corresponds to *margarine*). In order to be able to match also Sentence 8 with a pattern, one would additionally need to know that *replace* and *substitute* are synonyms. General-purpose ontologies, such as *WordNet* [19], can provide such knowledge.

Another method to obtain patterns is to learn them from text. In order to do so, one needs labeled contexts, e.g. if patterns for relation type *Can-be-Substituted-by(FOOD-ITEM, FOOD-ITEM)* are to be learnt, sentences where instances of that relation are expressed have to be collected. A sufficiently large amount of such labeled data enables state-of-the-art supervised machine learning methods, such as *Support Vector Machines* [26], to be applied. A model produced by such a classifier is a weighted set of features which allows relation instances to be extracted. In principle, the features can be similar to the manually designed patterns. However, one typically uses a much larger set of features (patterns). One does not need to include exactly those features that are predictive. This is usually learnt by the classifier, i.e. highly weighted features roughly correspond to the predictive patterns. The features that are chosen as input for the learning algorithm can consequently be much more generic than manually designed surface patterns. Typical examples are words or word sequences between the arguments of a relation in a training sentence or the syntactic relationship between the arguments (as shown above). Since the feature space is usually fairly large, the resulting models that are learnt can be much more expressive than a set of manually defined surface patterns. In particular, the coverage of those models may be much higher.

The downside of this method is of course the time effort involved in labeling the data. A standard way of acquiring such data would be annotating large amounts of textual data, sentence by sentence, and mark the entities (in the text) between which there holds the target relation type, e.g. *Can-be-Substituted-by*. Fortunately, there are alternative methods that try to reduce that annotation effort. The class of methods commonly referred to as *distant supervision* [20] is a fairly recent methodology that could be applicable. It makes certain (simplifying) assumptions about the realization of relations that can drastically speed up the annotation process. Instead of annotating texts from scratch, one could, for instance, define a set of prototypical arguments of the target relation type, e.g. <*fish fingers*, *fish cake*> or <*margarine*, *butter*> for *Can-be-Substituted-by*, and then consider sentences in which those entities co-occur, for example Sentence 9, as positive training data.

9. I often use *margarine* instead of *butter*.

So, instead of *directly* labeling sentences, one just needs to formulate argument pairs. The remaining steps of this method can be done fully automatically. This annotation is much less time consuming since common argument pairs can have quite many matches within a large corpus. Moreover, the gold standard used in our preliminary work [29] introduced in [30] could be used for that very purpose.

Of course, there are limitations to this approach. One assumes that the co-occurrence of two entity pairs will always represent the target

relation type. However, in some sentence their occurrence could be co-incidental, such as the co-occurrence of *margarine* and *butter* in Sentence 10 (although if the chosen argument pairs are good prototypes, this situation will rarely occur and thus not critically mar the quality of the training data).

10. I just went to the supermarket to buy some *margarine*, *butter*, cheese, vegetables and potatoes.

6.3.3 Beyond Simple Patterns – Why further linguistic analysis might be helpful

Most pattern-based approaches focus on a propositional level of how a relation is expressed. However, we observed that for some relation types it is vital to consider the *embedding* of those propositions as it may discard the (general) validity of the proposition. For instance, consider the relation instance *Is-Healthy(beans)*. With a pattern-based approach, it would be easy to detect a typical occurrence, such as Sentence 11. Imagine, for example, that a pattern sequence *X BE*[4] *healthy* has been acquired. However, this sequence would also fire in Sentences 12-16 even though none of these sentences entails that statement.[5]

11. *Beans are healthy*.
12. I don't think that *beans are healthy*.
13. I really wonder whether *beans are healthy*.
14. My aunt claims that *beans are healthy*. (But this is wrong!)
15. *Beans are healthier* than chocolate.
16. It could be that *beans are healthy*.

Sentence 12 is negated, Sentence 13 is an indirect question, Sentence 14 reports somebody else's belief, Sentence 15 is a comparison, while in Sentence 16 there is a modal embedding. Some appropriate linguistic analysis (even with current state-of-the-art NLP technology) should be able to detect these types of embeddings. It involves common tasks, such as *negation detection* [3, 24] (Sentence 12) or *opinion holder extraction* [5] (Sentence 14) that are mostly dependent on lexical and syntactic information. This linguistic analysis could be used as an additional rule that undoes an erroneous detection of a relation by a pattern-based approach.

7 Difficulties and How They can be Solved

We already pointed out in Section 3 that for state-of-the-art NLP it is not possible to achieve a full textual understanding. In particular, relations that require some pragmatic knowledge cannot be extracted. In this section, we will not discuss the difficulties of NLP with regard to that particular problem but focus on difficulties that typically arise with those types of methods that we proposed in the previous sections. We believe that these difficulties are more imminent problems to the task and that they are also more likely to solved (at least partially) in the near future.

As stated previously in this paper, the text type we think is most suitable from which to extract knowledge regarding food is user-generated content from the web. Irrespective of the concrete task that is to be carried out on these data, the text type itself already entails a significant problem. User-generated content is typically not subject to any checking that the texts that are produced are suitable. As a consequence of that, texts may contain errors on various levels. Words may be misspelt, sentences may be ungrammatical, wording may be inaccurate or misleading, and even complete statements may be incomprehensible. Moreover, statements may be off-topic or offensive (e.g. flames). Of course, this has a negative impact on NLP methods as the largest part of research in NLP assumes that the texts contain no errors. If words are misspelt, they cannot be properly recognized. WordNet (Section 6.2), for example, cannot anticipate incorrect spelling since it only contains correctly spellt entries. In the previous section we stated that some methods to extract relations require some linguistic analysis. These analyses are typically produced by a parser. Not only ungrammatical sentences may negatively affect the analyses made by such parsers. Most automatic syntactic analyses require that all words of a sentence have been recognized and that both the wording and the syntactic constructions resemble those data on which the parser has been developed. In spite of deviations, a parser may produce an analysis but the analysis may be very wrong. As the majority of NLP tools are developed on regular (*tidy*) newswire texts, one has to expect a significant *domain mismatch* when using those tools on other text types.

Only until recently, the necessity of adapting common NLP tools to other domains, in particular noisy text types as can be found in social media, has been addressed in research. Already initial experiments on that task yield promising results [8, 9, 10]. What this line of research mostly attempts is capturing systematics behind misspelling words and training parsers on those sentences that are more representative of the target domain than traditional newswire texts. (Thus, to some extent, systematic ungrammaticality can be learnt from those data.) As this line of research is still in its infancy, up to now, there are no NLP tools publicly available that have been tuned to these special data.

As a consequence, the question, of course, may arise whether any research on social media is premature and should be carried out despite the lack of NLP tools that work sufficiently well on the user-generated content. Moreover, these adaptation efforts will only be able to solve some of those problems that are inherent to that domain. Inaccurate wording or incomprehensible statements will still remain a problem. Fortunately, not every sentence in user-generated texts contains these errors. After all, even with our preliminary work [29], we could show that some knowledge can be extracted. However, more research needs to be carried out in order to quantify the impact of those errors.

Irrespective of the technical problems that may occur during the automatic extraction of knowledge, one may also wonder how much knowledge is actually encoded in the data available. After all, the text corpora on which data are extracted can only be finite. Moreover, we just mentioned that in some way we rely on relations to be mentioned several times within our text collection. In the worst case, we would only be able to extract frequently occurring relations that are already common knowledge (e.g. relations of the type *Can-be-Substituted-by(butter, margarine)*). In other words, we would extract only that information that is not worth to be included in a specially built knowledge base since every ordinary person already has that knowledge. At this point in time, without some thorough empirical analysis, no definite judgement can be rendered. There is, however, one insight that may support the usefulness of the approach sketched in this paper, which is that social media are rapidly and steadily growing. A natural consequence of this is that the knowledge that can be extracted by state-of-the-art NLP methods may increase. So, a relation that cannot be extracted from textual data today because it is either not contained in those data or occurs too infrequently does not mean that it cannot be extracted from similar domain data in a few years' time. By then, there is much more text available that may al-

[4] By BE all inflectional forms of the verb *to be* are meant, i.e. *am*, *is*, *are*, *was*, *were* etc.

[5] We assume that Sentence 15 may also match as one usually normalizes word forms, so *healthier* would be reduced to the positive *healthy*.

low automatic NLP techniques to successfully extract this relation.

8 Conclusion

In this paper, we presented an outlook on the effectiveness of NLP in applications in the food domain. We identified two potential scenarios, namely advice on preparing meals and health-related issues, that predominate research in the food domain with regard to artificial intelligence and found that these scenarios are also quite suitable for NLP techniques. As a source for extracting knowledge we outlined the benefits of social media. Different extraction methods, ranging from co-occurrence measures to more complex linguistic analyses, were discussed. Finally, we also addressed potential problems that NLP methods may cause on the tasks we have proposed.

Acknowledgements

This work was funded by the German Federal Ministry of Education and Research (Software-Cluster) under grant no. “01IC10S01”.

REFERENCES

[1] Fadi Badra, Rokia Bendaoud, Rim Bentebibel, Pierre-Antoine Champin, Julien Cojan, Amélie Cordier, Sylvie Després, Stéphanie Jean-Daubias, Jean Lieber, Thomas Meilender, Alain Mille, Emmanuel Nauer, Amedeo Napoli, and Yannick Toussaint, ‘TAAABLE: Text Mining, Ontology Engineering, and Hierarchical Classification for Textual Case-Based Cooking’, in *European Conference on Case-Based Reasoning - ECCBR 2008, Workshop Proceedings*, (2008).

[2] Brandon Brown, Marshini Chetty, Andrea Grimes, and Ellie Harmon, ‘Reflecting on health: A system for students to monitor diet and exercise’, in *Proceedings of CHI Extended Abstracts*, (2006).

[3] Wendy Webber Chapman, Will Bridewell, Paul Hanbury, Gregory F. Cooper, and Bruce G. Buchanan, ‘A Simple Algorithm for Identifying Negated Findings and Diseases in Discharge Summaries’, *Journal of Biomedical Informatics*, **34**, 301 – 310, (2001).

[4] Pei-Yu Chi, Jen-Hao Chen, Hao-Hua Chu, and Bing-Yu Chen, ‘Enabling Nutrition-Aware Cooking in a Smart Kitchen’, in *Proceedings of CHI Extended Abstracts*, (2007).

[5] Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan, ‘Identifying Sources of Opinions with Conditional Random Fields and Extraction Patterns’, in *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*, Vancouver, BC, Canada, (2005).

[6] Kenneth Ward Church and Patrick Hanks, ‘Word Association Norms, Mutual Information, and Lexicography’, *Computational Linguistics*, **16**(1), 22–29, (1990).

[7] Amélie Cordier, Jean Lieber, Pascal Molli, Emmanuel Nauer, Hala Skaf-Molli, and Yannick Toussaint, ‘Knowledge Acquisition and Discovery for the Textual Case-Based Cooking system WIKITAABLE’, in *International Conference on Case-Based Reasoning - ICCBR 2009, Workshop Proceedings*, (2009).

[8] Jennifer Foster, “‘cba to check the spelling”: Investigating Parser Performance on Discussion Forum Posts’, in *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (HLT/NAACL)*, (2010).

[9] Jennifer Foster, Ozlem Cetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef van Genabith, ‘#hardtoparse: POS Tagging and Parsing the Twitterverse’, in *Proceedings of AAAI Workshop on Analysing Microtext*, (2011).

[10] Jennifer Foster, Ozlem Cetinoglu, Joachim Wagner, Joseph Le Roux, Joakim Nivre, Deirdre Hogan, and Josef van Genabith, ‘From News to Comment: Resources and Benchmarks for Parsing the Language of Web 2.0’, in *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*, (2011).

[11] Jeana Frost and Brian K. Smith, ‘Visualizing Health: Imagery in Diabetes Education’, in *Proceedings of Designing for User Experiences (DUX)*, (2003).

[12] Andrea Grimes and Richard Harper, ‘Celebratory Technology: New Directions for Food Research in HCI’, in *Proceedings of the Annual SIGCHI Conference on Human Factors in Computing Systems (CHI)*, (2008).

[13] Reiko Hamada, Jun Okabe, Ichiro Ide, Shin’ichi Satoh, Shuichi Sakai, and Hidehiko Tanaka, ‘Cooking Navi: Assistant for Daily Cooking in Kitchen’, in *Proceedings of the annual ACM international conference on Multimedia (MULTIMEDIA)*, (2005).

[14] Ichiro Ide, Yuka Shidochi, Yuichi Nakamura, Daisuke Deguchi, Tomokazu Takahashi, and Hiroshi Murase, ‘Multimedia Supplementation to a Cooking Recipe Text for Facilitating Its Understanding to Inexperienced Users’, in *Proceedings of the Workshop on Multimedia for Cooking and Eating Activities (CEA)*, (2010).

[15] Nancy Ide and Catherine Macleod, ‘The American National Corpus: A Standardized Resource of American English’, in *Proceedings of Corpus Linguistics*, (2001).

[16] Nitin Khann, Carol J. Boushey, Deborah Kerr, Martin Okos, David S. Ebert, and Edward J. Delp, ‘An Overview of The Technology Assisted Dietary Assessment Project at Purdue University’, in *Proceedings of the Workshop on Multimedia for Cooking and Eating Activities (CEA)*, (2010).

[17] Jennifer Mankoff, Gary Hsieh, Ho Chak Hung, Sharon Lee, and Elizabeth Nitao, ‘Using Low-Cost Sensing to Support Nutritional Awareness’, in *Proceedings of Ubicomp*, (2002).

[18] Michael McCandless, Erik Hatcher, and Otis Gospodnetić, *Lucene in Action*, Manning Publications, 2nd edn., 2010.

[19] George Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller, ‘Introduction to WordNet: An On-line Lexical Database’, *International Journal of Lexicography*, **3**, 235–244, (1990).

[20] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky, ‘Distant Supervision for Relation Extraction without Labeled Data’, in *Proceedings of the Joint Conference of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL/IJCNLP)*, Singapore, (2009).

[21] Kenzaburo Miyawaki, Satoshi Nishiguchi, and Mutsuo Sano, ‘Extraction of Mastication in Diet Based on Facial Deformation Pattern Descriptor’, in *Proceedings of the Workshop on Multimedia for Cooking and Eating Activities (CEA)*, (2010).

[22] Kenzaburo Miyawaki and Mutsuo Sano, ‘A Cooking Support System for People with Higher Brain Dysfunction’, in *Proceedings of the ACM Multimedia 2009 Workshop on Multimedia for Cooking and Eating Activities (CEA)*, (2009).

[23] Gordon Mohr, Michael Stack, Igor Ranitovic, Dan Avery, and Michele Kimpton, ‘An Introduction to Heritrix, an open source archival quality web crawler’, in *Proceedings of the International Web Archiving Workshop (IWAW)*, (2004).

[24] Pradeep G. Mutalik, Aniruddha Deshpande, and Prakash M. Nadkarni, ‘Use of General-purpose Negation Detection to Augment Concept Indexing of Medical Documents – A Quantitative Study Using the UMLS’, *Journal of the American Medical Informatics Association*, **8**(6), 598 – 609, (2001).

[25] Yuka Shidochi, Tomokazu Takahashi, Ichiro Ide, and Hiroshi Murase, ‘Finding Replacable Materials in Cooking Recipe Texts Considering Characteristic Cooking Actions’, in *Proceedings of the ACM Multimedia 2009 Workshop on Multimedia for Cooking and Eating Activities (CEA)*, (2009).

[26] Vladimir N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 1995.

[27] Suzan Verberne, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen, ‘What is not in the Bag of Words for Why-QA?’, *Computational Linguistics*, **36**(2), 229 – 245, (2010).

[28] Qing Wang and Kiduk Jie Yang, ‘Drinking Activity Analysis from Fast Food Eating Video Using Gernative Models’, in *Proceedings of the ACM Multimedia 2009 Workshop on Multimedia for Cooking and Eating Activities (CEA)*, (2009).

[29] Michael Wiegand, Benjamin Roth, and Dietrich Klakow, ‘Web-based Relation Extraction for the Food Domain’, in *Proceeding of the International Conference on Applications of Natural Language Processing to Information Systems (NLDB)*. Springer-Verlag, (2012).

[30] Michael Wiegand, Benjamin Roth, Eva Lasarcyk, Stephanie Köser, and Dietrich Klakow, ‘A Gold Standard for Relation Extraction in the Food Domain’, in *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, (2012).