Feature Extraction and Summarization of Recipes Using Flow Graph

Yoko Yamakata[1], Shinji Imahori[2], Yuichi Sugiyama[1], Shinsuke Mori[3], and Katsumi Tanaka[1]

[1] Graduate School of Informatics, Kyoto University
Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan
[2] Graduate School of Engineering, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan
[3] Academic Center for Computing and Media Studies, Kyoto University
Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan

Abstract. These days, there are more than a million recipes on the Web. When you search for a recipe with one query such as "*nikujaga*," the name of a typical Japanese food, you can find thousands of "*nikujaga*" recipes as the result. Even if you focus on only the top ten results, it is still difficult to find out the characteristic feature of each recipe because a cooking is a work-flow including parallel procedures. According to our survey, people place the most importance on the differences of cooking procedures when they compare the recipes. However, such differences are difficult to be extracted just by comparing the recipe texts as existing methods. Therefore, our system extracts (i) a general way to cook as a summary of cooking procedures and (ii) the characteristic features of each recipe by analyzing the work-flows of the top ten results. In the experiments, our method succeeded in extracting 54% of manually extracted features while the previous research addressed 37% of them.

1 Introduction

Cooking is one of the most fundamental activities of human social life. It is not only connected with the joy of eating but also deeply affects various aspects of human life such as health, dietary, culinary art, entertainment, human communication, and so on. Hene, the number of recipes on the Web has been increasing rapidly in recent years. In Japan, COOKPAD, the biggest recipe portal site, has more than 1.5 million recipes and 12 million users [14]. Rakuten-Recipe has more than 620,000 recipes. In the United States, Food.com has more than 475,000 recipes, while Allrecipe.com and FoodNetWork.com have more than ten million users. Google also offers a service for recipe search.

However, more is not always better. Even if you submit just one query, such as "*nikujaga*," to COOKPAD, you can find more than 5,600 "*nikujaga*" recipes. Of course, all of the recipes explain how to cook "*nikujaga*," but they are somewhat different. Some recipes fry meat in advance while others fry meat after an onion. Some recipes add a soy-source during frying while others mix it during stewing.

A. Jatowt et al. (Eds.): SocInfo 2013, LNCS 8238, pp. 241–254, 2013.

The title is not useful because it was given by the recipe authors freely and it reflects only his/her subjective evaluation. You try to find your favorite recipe by reading the text parts of several recipes. However, it is very hard because a cooking is a work-flow with parallel procedures and it requires much effort to understand, memorize, and compare these cooking procedures.

Because all of the recipes are obtained using the same query "*nikujaga*," the ways of cooking described by these recipes must be similar. Therefore, we propose a method for finding their general way to cook as the summary by extracting the common structure of the cooking flow-graphs from the top ten search results. Moreover, the system obtains the characteristic features of each recipe by comparing it with the generated general recipe.

As it is introduced in Section 3.1, we are researching a method that converts a recipe text to a flow-graph of cooking procedures. Therefore, we assume that all recipes had been converted to such flow-graphs using this method. Additionally in this paper, a recipe flow graph is assumed to be a tree structure because the most of the recipes' flow-graphs can be represented as tree-type graphs. Hereafter, we refer to it as a "recipe tree." The system conducts node-to-node mapping of all pairs of recipe trees and integrates the most similar pair of recipes. The system repeats this integration and finally obtains one general recipe. The characteristic features of each recipe can be extracted by mapping the recipe tree and the obtained general recipe tree and finding the differences.

2 Feature Types and Their Importances of Recipe

In this section, we analyses which type of feature should be extracted from a recipe in the purpose.

To analyze recipe feature types and their importances, we conducted a survey. We searched on the recipe portal site COOKPAD [14] with the four queries "*nikujaga*," "*carbonara*," "beef stew," and "*nigauri* (a name of an ingredient)," and collected top 10 recipes for each query. Therefore, these four recipe sets of 10 recipes respectively, 40 recipes in total, were obtained as a test data. We asked two annotators, who were undergraduate students, to extract characteristic features of each recipe comparing with the other nine recipes in each set manually. We also asked them to assign a rank to each feature according its importance when they found more than two features for one recipe. Consequently, 197 features with 29 duplication and 168 unique features were obtained from 40 recipes in total. We classified obtained 168 features into seven types as follows.

- **Additional ingredient:** when a recipe has an uncommon ingredient.
- **Reduced ingredient:** when a recipe does not have a common ingredient.
- **Ingredient quantity:** when a quantity of an ingredient is significantly differ from the others.
- **Uncommon action:** when a recipe has an uncommon action.
  ex.) The potato is immersed in water after cutting.
- **Action order:** when an order of actions differs from the other recipes.
  ex.) Soy source is mixed when it fries ingredients in a recipe while soy source is mixed after adding water to the ingredients in other recipes.

– **Tool:** when an uncommon tool is used.
– **Writing type:** when it is different writing type from the others.

The number and the proportions of each type in the manually extracted features are shown in Table 1. As you see in the table, the most common feature is additional ingredient. Though the proposed method is able to extract this feature, even a simple method that compares the member of the ingredient list of the recipe with the others can also extract such as [10]. Reduced ingredient and quantity of ingredient are also able to be extracted in the same way. Meanwhile, the annotators considered the action order is the most important features even though the proportion of it was not very high. Uncommon action is also considered more important than additional ingredient. Since these two features cannot be extracted just by comparing the words of the instructions, it is said that sophisticated analysis is required to find important recipe features.

Table 1. Manually extracted features for each categories

Type	Ave. rank	*Nikujaga*	*Carbonara*	*Nigauri*	Beef stew	Total
Action order	**1.8**	5 (8%)	1 (2%)	1 (2%)	5 (11%)	12 (7%)
Quantity of ingredient	2.1	6 (9%)	0 (0%)	0 (0%)	1 (2%)	7 (4%)
Reduced ingredient	2.2	6 (9%)	1 (2%)	0 (0%)	1 (2%)	8 (5%)
Uncommon action	**2.4**	11 (17%)	2 (5%)	4 (9%)	7 (16%)	24 (14%)
Additional ingredient	2.5	28 (44%)	39 (89%)	3 (7%)	34 (77%)	104 (62%)
Writing style	3.7	1 (2%)	0 (0%)	0 (0%)	2 (5%)	3 (2%)
Cooking tool	3.9	7 (11%)	1 (2%)	0 (0%)	2 (5%)	10 (6%)
Total		64	44	8	52	168

3 Pre-processing for Recipes

3.1 Recipe Tree: The Work-Flow Format of a Recipe

We first convert recipe procedures written in a natural language into a tree-type work-flow graph. Fig. 1 shows an example of a recipe tree. In the recipe tree, each leaf node corresponds to an ingredient of the recipe such as “a potato,” “meat,” and “sugar.” Each intermediate node corresponds to a cooking action of eleven categories, “Mix,” “Cut,” “Fry,” “Roast,” “Boil,” “Cook in boiling water,” “Deep frying,” “Heat by instrument,” “Steam,” “Stop ongoing action” and“Others”. The root node corresponds to the completed dish which is ready to serve. The label of each node is a pair of a type, ingredient or cooking action, and a word sequence corresponding to the name. The root node is the only exception and has the dish name. For instance, a sentence “A potato is cut to larger bite-sized pieces, and is immersed in water” corresponds to the sub-tree in the dotted circled in Fig. 1.

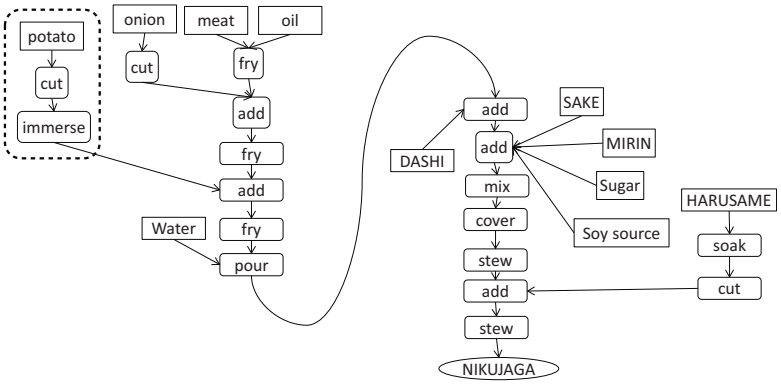


Fig. 1. A recipe tree of one "*nikujaga*"

Table 2. Named entity tags

NE Tag	Meaning	NE Tag	Meaning
F	Food	Ac	Action by the chef
T	Tool	Af	Action by foods
D	Duration	S	State of foods
Q	Quantity		

Such a recipe tree can be automatically converted from a recipe text using natural language processing (NLP) [3–5]. NLP for recipe texts proposed in [5] consists of two important parts based on machine learning methods. The first one is named entity (NE) recognition after word identification which extracts important word sequences shown in Table 2 appearing in the recipe text. The second one is predicate-argument structure (PAS) analysis after syntactic analysis which determine the subject, the direct object, and the indirect object (arguments) for a verb (predicate). In this paper we adopt the same named entity definition but only use F (Food) for ingredient nodes and Ac (Action by the chef) for cooking action nodes. We use PAS of the verbs marked as Ac which corresponds to the arcs in recipe trees.

It is reported that the NE recognition accuracy went up from 53.4% to 67.0% by only 5 hour annotation [5]. The NE recognition accuracy for the general tag set (person name, organization name, place, etc.) is around 80% $\sim$ 90% when enough large training data are available [7]. In addition, there are less variations for food names and cooking action names than for the general tag set. Therefore we can say that it is possible to achieve about 90% accuracy just by preparing practically large training data. Currently the accuracy of PAS analysis is less than NE recognition. In the recipe domain, however, the vocabulary is much more limited than in general domains of NLP such as newspaper articles. Thus a domain adaptation technique for PAS analysis [12] allows us to achieve an enough high accuracy with a practical size of the training data.

As we described above, the NLP community is working on recipe texts, as well as patent disclosures etc., as a domain adaptation example of NE recognition or PAS analysis. Since currently the accuracies are, however, not sufficiently high for the application we propose in this paper, in the experiments we use recipe trees manually converted from recipe texts.

3.2 Tree Mapping Algorithms

Tree is one of the most common and well-studied combinatorial structures in computer science. Comparison of two (or more) trees is a fundamental task in many applications such as computational biology, structured text databases and image analysis. Various measures have been proposed and studied for comparison of two trees. Among such measures, tree edit distance is the most common and well studied. For two labeled trees T and T', the edit distance from T to T' is measured by the minimum cost sequence of edit operations needed to transform T into T'. The edit operations are deletion, insertion, and substitution. For ordered labeled trees, efficient algorithms for computing the edit distance have been proposed in the literature. Tai [9] developed the first polynomial time algorithm for the problem, several improvements followed, and Demaine et al. [1] proposed an optimal algorithm that runs in $O(n^3)$ time for n-node trees.

For unordered labeled trees, including recipe trees, the problem of computing the edit distance between two trees is difficult (more precisely, the problem is known to be NP-hard [13]). Therefore, it is reasonable to try to develop heuristic algorithms for this case. Shasha et al. [8] proposed a simple heuristic algorithm by sorting and iterative improvement algorithms based on metaheuristics. However, they focused on only the number of child for ordering while label matching is required for our purpose. Do and Rahm [2] proposed a system called COMA, which provides an extensible library of simple and hybrid match algorithms, but the editing costs cannot be adjusted flexibly.

For a given set of trees, computing one tree that is similar to all the other trees is a challenging task and has been studied in the literature. Phillips and Warnow [6] showed the hardness of this problem and proposed a heuristic method for computing a tree called the asymmetric median tree. Their method works well for evolutionary trees (in which species label the leaves). However, it is hard to apply this method to our application and different heuristic methods are necessary to compute a general recipe tree.

4 Generation of General Recipe Tree

4.1 Framework

The procedures of the system are as follows:

[Step 1] Ten recipes are given as a search result. In advance, every recipe has been converted to a recipe tree, in the form of a rooted, labeled, and unordered tree. Set the weight w of a tree T to one for each recipe tree.

[Step 2] The system calculates an approximate edit distance $d(T, T')$ between every pair of trees (T, T'). For each calculation, two unordered trees are transformed into ordered trees so that the distance of them becomes closer.

[Step 3] The trees of the closest pair T and T' are integrated to one tree. Let T and T' be the two trees to be integrated, and w and w' their weights. The system generates a new tree with the properties that (i) the distance from T is around $d(T, T')w'/(w + w')$, (ii) the distance from T' is around $d(T, T')w/(w + w')$, and (iii) the weight is $w + w'$. After adding the new tree to the current set of trees and removing two trees T and T' from it, our algorithm returns to Step 2, if two or more trees are remaining. Go to Step 4 when the number of trees becomes one. The final integrated tree, the general tree T_{gen}, is the output of our system.

[Step 4] Extract the characteristic features of each recipe tree by mapping the recipe tree with the general tree T_{gen} and finding the differences.

4.2 Transformation to Ordered Tree

As stated in Section 3.2, it is difficult to compute the accurate edit distance between unordered labeled trees. Therefore, the system converts each unordered tree into an ordered tree so that an approximate edit distance between them becomes small.

In our heuristic method, the system decides the order of children for each node from the root node to leaf nodes. At the beginning, the system finds the node that is closest to the root node and that has more than two or more children for each tree. Let u and v be the found nodes of two trees, and $u_1, u_2, \ldots, u_p$ and $v_1, v_2, \ldots, v_q$ be the child nodes of u and v. To decide the orders of these child nodes, the system solves the following problem.

$$
\begin{aligned}
\text{Maximize } & \sum_{i,j} c(i,j)x(i,j) \\
\text{Subject to } & y(i) = \sum_{j} x(i,j) \geq 0 \ (i = 1, 2, \ldots, p) \\
& z(j) = \sum_{i} x(i,j) \geq 0 \ (j = 1, 2, \ldots, q) \\
& x(i,j)(y(i) - 1)(z(j) - 1) = 0 \ (i = 1, \ldots, p, \ j = 1, \ldots, q) \\
& x(i,j) \in \{0, 1\},
\end{aligned}
$$

where $x(i, j)$ is the decision variable whose value is one if and only if node u_i is mapped to node v_j, and $c(i, j)$ denotes the number of common ingredients which appear in both sub-trees whose root nodes are u_i and v_j. Note that the system does not define node-to-node mapping at this point. One reason is that the number of children p and q may be different, and another reason is that a one-to-many mapping can be suitable for some cases (see Fig. 2 as an example). After one-to-one or one-to-many mappings are obtained by solving this problem at nodes u and v, the procedure goes to their descendants (i.e., solving similar

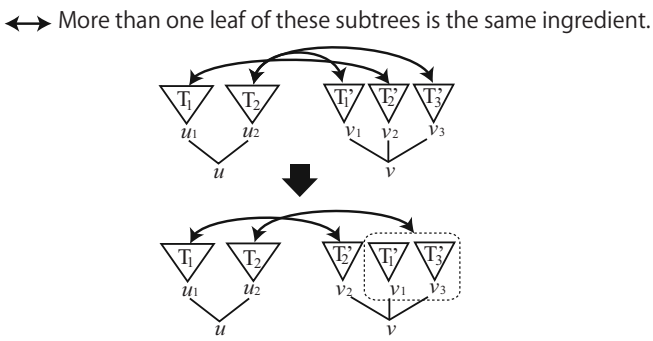


Fig. 2. Example for transformation to ordered trees

problems for each corresponding sub-tree). The procedure stops when it cannot find a descendant.

4.3 Node-to-Node Mapping between Two Trees

For two rooted, labeled and ordered trees T and T', we compute the minimum cost sequence of edit operations needed to transform T into T'. The edit operations are (i) deletion: deleting a node from a tree, (ii) insertion: inserting a node into a tree, and (iii) substitution: changing one node of a tree into another node. Each operation has its cost $c_{del}(u), c_{ins}(u)$, and $c_{sub}(u, v)$, respectively.

Now, let consider such case that "Stir the onion and add the carrot." Note a chef keeps stirring the vegetables when he/she add the carrot. It means that "Mix" including "add", "throw in", "put" and "pour" can be given to the other cooking action. "Other" is also in the same way. Therefore, we set $c_{sub}(u, v)$ depend on combination of u and v' as followings.

When u is a leaf:
$c_{sub}(u, v) = 0$ if the labels of u and v are the same.
$c_{sub}(u, v) = \infty$ otherwise.
When u is a root:
$c_{sub}(u, v) = 0$ if v is a root.
$c_{sub}(u, v) = \infty$ otherwise.
When u is an intermediate node:
$c_{sub}(u, v) = 0$ if the labels of u and v are the same.
$c_{sub}(u, v) = C_{sub1}$ if at least one of the labels of u or v is "Mix" or "Others".
$c_{sub}(u, v) = C_{sub2}$ if the labels of u and v differ.

The cost of $c_{del}(u)$ and $c_{ins}(u)$ are set a constant number C_{del_ins} for all nodes.

The minimum cost sequence of edit operations required to transform T into T' can be computed using the algorithm of Tai [9]. Although this algorithm runs fast enough for our data set, more sophisticated algorithms (e.g., [1]) will be useful for complicated recipes with many ingredients and cooking actions.

4.4 Recipe Tree Integration

To suggest the general way of cooking in a given recipe set, the generated general tree should be almost equally close to each of the recipes. The general tree should have the common characteristics of the given set. That is, ingredients or cooking actions that appear often in the given set must be extracted into the general tree. Moreover, sequences of cooking actions are also important for recipes and should be stored in the general recipe. Therefore, when the system integrates two trees, it counts how many recipes are integrated into each tree and generates a new tree that is affected by each tree in proportion as its integration counts.

For each pair of recipe trees, edit distance is computed using the methods explained in Sections 4.2 and 4.3. The system integrates the closest pair of recipes into one intermediate recipe tree. Let T and T' be the two ordered trees to be integrated, and w and w' their weights. The edit distance $d(T, T')$ and a set of edit operations transforming T into T' are computed. Our system generates a new intermediate tree whose distance from T (resp., T') is around $d(T, T')w'/(w + w')$ (resp., $d(T, T')w/(w + w')$). Concretely, the system adopts n insertion/deletion/substitution operations, where n is calculated as

$$n = d(T, T')w'/(w + w') \times m$$

and m is the number of insertion/deletion/substitution operations. The order of preference in this adoption is as follows:

- The deletion operation is adopted if
  - the deleted node is an ingredient and the number of its occurrences in the ten recipes is fewer than two.
  - the deleted node is a "Mix" or "Other" action.
- The insertion operation is adopted if
  - the inserted node is an ingredient and the number of occurrences in the ten recipes is two or more.
  - the inserted node is not a "Mix", "Other", or "Stop ongoing action" action.

There is no order of preference for substitution operations. After such integration procedures, the generated new tree T'' could have an action node as a leaf, because the connected leaf node of an ingredient was removed. In such cases, the system removes a sub-tree that has no ingredient as its leaf.

Then, T and T' are removed from the current set of trees and T'' with the weight $(w + w')$ is added to the set. The system repeats this integration and finally obtains one general recipe tree T_{gen}.

4.5 Characteristic Feature Extraction

The features of each recipe are extracted by comparing the recipe with the general recipe T_{gen}. Concretely, T is mapped with T_{gen} and deletion/insertion/substitution operations corresponding to the characteristic features of T.

5 Experiments and Results

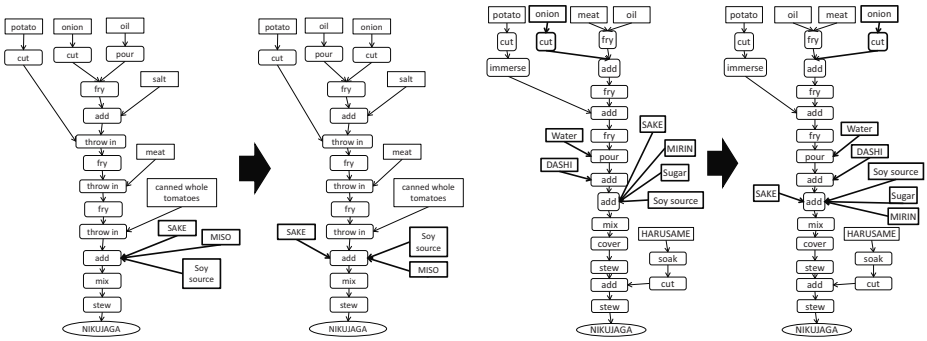
5.1 Recipe Data Set

The given data set was the top ten results of searching with a query "*Nikujaga*" at COOKPAD [14]. The recipe IDs of obtained recipes were A) 1487670, B) 1485091, C) 1499546, D) 1519874, E) 1521946, F) 1524200, G) 1531094, H) 1531503, I) 1531751, and J) 1531880 (you can find these recipes at COOKPAD by searching with these IDs as a query). Then, we converted them into unordered recipe trees manually, this process is possible to automatized as we stated in section 3.1.

5.2 Examples of Transformation to Ordered Trees

The system calculated the mapping score for all combinations of two of the ten recipe trees. For each pair, the trees were transformed to ordered trees so that these trees could be mapped with lower cost in accordance with the algorithm explained in Section 4.2.

Fig. 3 (a) and (b) show examples of transformation from an unordered tree to an ordered tree when mapping recipe D) to recipe G). As shown in these figures, the trees on the right are closer to each other than the trees on the left. The subtrees indicated by thick lines were reordered in this procedure.



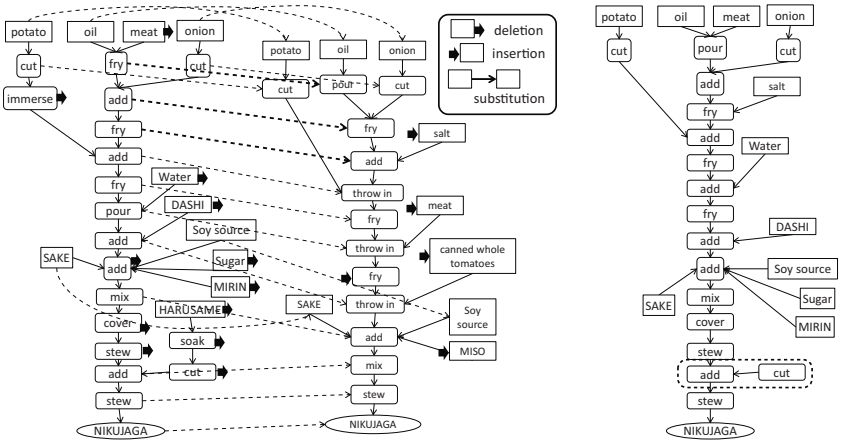
(a) Recipe transformation of D) for mapping with G)

(b) Recipe transformation of G) for mapping with D)

Fig. 3. Results of transforming from unordered to ordered tree

5.3 Node-to-node Mapping

The system calculated the mapping cost of editing distance. In this experiments, we set C_{del_ins}, C_{sub1}, and C_{sub2} as 7, 5, and 10, respectively. The mapping result between recipes D) and G) is shown in Fig. 4. Since it has 12 deletion operations, 5 insertion operations, and 3 substitution operations, the mapping cost was 134.



(a) Mapping result from recipe D) to G)

(b) Integrated recipe tree from recipes D) and G)

Fig. 4. Mapping and integration results

Table 3. Edit distances of all combinations of two of ten recipes

	A	B	C	D	E	F	G	H	I	J	Ave.
A	-	148	213	157	166	214	197	211	152	200	184.2
B	148	-	155	150	140	215	152	171	146	183	162.2
C	213	155	-	140	140	230	172	144	155	182	170.1
D	157	150	140	-	140	192	134	162	159	179	157.0
E	166	140	140	140	-	201	143	153	145	182	156.7
F	214	215	230	192	201	-	175	213	234	234	212.0
G	197	152	172	**134**	143	175	-	160	143	204	164.4
H	211	171	144	162	153	213	160	-	204	237	183.9
I	152	146	155	159	145	234	143	204	-	193	170.1
J	200	183	182	179	182	234	204	237	193	-	199.3

The mapping costs of all combinations of pairs of ten recipes are shown in Table 3. The first line on the right of this table shows the average distance from each recipe to the others. In this line, recipe E) gets the closest average distance. This means that recipe E) is the most general of these ten recipes.

5.4 Recipe Tree Integration

Since the pair of lowest cost was the combination of recipes D) and G), the system integrated these recipes and generated one tree, in accordance with the algorithm introduced in Section 4.4. Fig. 4 (b) shows the integrated recipe tree. Because one leaf node is not an ingredient but an action, "cut," the system removed that node and the "add" node for that leaf, as indicated by the dotted circle in Fig. 4 (b). When more than two or more nodes of the same type are

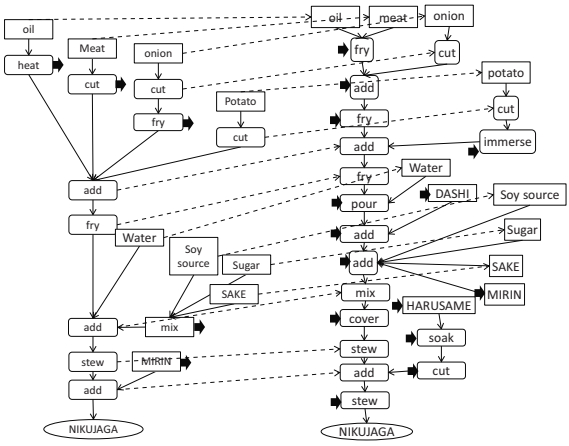


Fig. 5. Mapping result of the general tree (on the left) with recipe D) (on the right)

directly connected and have no other branch, these nodes are combined into one node. In this example, after removing the circled part, two "stew" nodes were directly connected and the system removed one of them.

5.5 General Recipe Tree of Ten Recipes

The system repeated the integration until the ten recipe trees became one. The finally integrated recipe tree is shown on the left side of Fig. 5. The edit distances between the general tree and each recipe tree are shown in Table 4. The average distance of the general tree is 133.7, while the average distance between each recipe with the others is greater than 157, as shown in Table 3. This means that the general recipe is much closer to all of the recipes than any one of them.

Table 4. The edit distances between the general and each recipe tree

	A	B	C	D	E	F	G	H	I	J	Ave.
General	159	63	138	133	87	162	127	131	152	185	**133.7**

5.6 Characteristic Features of Each Recipe

The characteristic features of recipe D) were extracted by mapping it with the general tree. The mapping result is shown in Fig. 5. According to the editing operation of the mapping, the characteristic features of recipe D) can be extracted as follows: (i) "MIRIN is added at the same timing with other seasoners," (ii) "place a small lid directly on the food," (iii) "It uses HARUSAME and DASHI," (iv) "meat is fried without cutting," and (v) "Potato is immersed in water after cutting." (ii), (iii), and (v) were matched with manually extracted features. However such manually extracted features as "use mince," "use sesame oil," and

"use a small amount of sugar" could not be extracted because the recipe tree did not include such data. On the other hand, (iv) was extracted incorrectly because the action of cutting meat might just be omitted in the instruction.

We adopted the feature extraction method to all of the ten recipes respectively and obtained 85 features in total. Then we compared them with the manually extracted features that were introduced in Section 2. The number of consistences between the manually extracted features and automatically extracted features, and precisions and recalls of our method are shown in Table 5.

As shown in the table, the features of "Action order", that was evaluated the most important features by the annotators and is difficult to be extracted by the previous researches, were extracted with 12% precision and 60% accuracy. Since 84% of the incorrectly extracted features could also be agreed as "Action order" feature when we recheck the recipe text, one of the reasons for these false positive results could be that the annotators could not find these features. If so, it means that the proposed methods has higher performance than human for extracting such features. The features of "Action type", which is also difficult to be extracted by previous researches, could be extracted with 45% precision and 45% recall. In total, our method succeeded in extracting 54% of manually extracted features while the previous researches address only 37% of them.

We also conducted the experiment on the recipes of "Carbonara" and it achieved precision of 47% with recall of 60%. Speaking about "Action order" and "Uncommon action", two of three features were successfully extracted.

Table 5. Feature type of recipe

Type	Ave. ranking	Manually extracted	Automatically extracted	# of consistence	Precision	Recall
Action order	**1.8**	5 (8%)	25 (29%)	3	12%	60%
Quantity of ingredient	2.1	6 (9%)	0 (0%)	0	-%	0%
Reduced ingredient	2.1	6 (9%)	14 (16%)	6	43%	100%
Action type	**2.4**	11 (17%)	11 (13%)	5	45%	45%
Additional ingredient	2.5	28 (44%)	29 (34%)	17	59%	61%
Writing style	3.7	1 (2%)	0 (0%)	0	-	0%
Cooking tool	3.9	7 (11%)	6 (7%)	3	50%	43%
Total		63	85	34	40%	54%

6 Discussions

Mapping Cost. In the proposed method, a cost of node-to-node mapping is set according to their types so that ingredients or actions of the same broad categories are treated as the same. However, such a difference is sometimes very meaningful for finding characteristic features of a recipe. Moreover, not all actions are equally important. For example, a washing action on a potato is abbreviated very often, because it goes without saying that a potato should be

washed. As future work, we will make the mapping cost of insertion, deletion, and substitution operations depend on the name of the ingredient/action, so that the mapping costs between similar types of ingredients or actions will be lower than between dissimilar types.

Integration Weight. The system integrated two trees to one according to their weight. The weight of a tree means the number of trees that are integrated into the tree. However, not all parts of the tree are overlapped in all previous integrations; some parts can be joined at the last integration. As future work, we will give a weight not to the whole of a tree but to each node. The weight of a node should be counted according to how many times the node is overlapped in the previous integrations.

Applications. In this paper, we described a method for generating a general recipe and for extracting characteristic features of a recipe. In a previous study, we used a recipe tree as a scenario of a chef's behavior for recognizing the chef's cooking in a cooking video [11]. Recipe-tree mapping can be used also for recipe rewriting. For example, a simple recipe can be transformed into a detailed recipe, because the system can obtain pairs of sentences in simple and detailed descriptions by tree mapping.

Though we generated a general recipe from the top ten results in this paper, it is possible to address more than ten results, if there are sufficient calculation time, memory, and processors.

7 Conclusions

In this paper, we proposed a method for obtaining a general way to cook from a set of multiple recipes and extracting characteristic features of each recipe. All recipes were converted in advance to recipe trees. This process will be automatized as we stated in section 3.1. The system calculated the edit distance of all combinations of pairs of recipes and integrated the recipes of the closest pair into one tree. In these processes, the system took into account the differences in importance between the action types. As the result, our method succeeded in extracting 54% of manually extracted features while the previous research addressed 37% of them.

Acknowledgments. This work was supported by JSPS KAKENHI Grant Numbers 23700144, 23500177 and 24240030.

References

1. Demaine, E.D., Mozes, S., Rossman, B., Weimann, O.: An optimal decomposition algorithm for tree edit distance. ACM Trans. on Algorithms 6, Article No. 2 (2009)
2. Do, H.-H., Rahm, E.: COMA – a system for flexible combination of schema matching approaches. In: Proc. of the 28th International Conference on Very Large Data Bases, pp. 610–621 (2002)

3. Hamada, R., Ide, I., Sakai, S., Tanaka, H.: Structural analysis of cooking preparation steps. IEICE Trans. J85-D-II, 79–89 (2002) (in Japanese)
4. Karikome, S., Fujii, A.: Improving structural analysis of cooking recipe text. IEICE Technical Report 112(75), DE2012-8, 43–48 (2012) (in Japanese)
5. Mori, S., Sasada, T., Yamakata, Y., Yoshino, K.: A machine learning approach to recipe text processing. In: Cooking with Computers Workshop, pp. 1–6 (2012)
6. Phillips, C., Warnow, T.J.: The asymmetric median tree – a new model for building consensus trees. Discrete Applied Mathematics 71, 311–335 (1996)
7. Sang, E.F.T.K., Meulder, F.D.: Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In: Proc. of CoNLL 2003, pp. 142–147 (2003)
8. Shasha, D., Wang, J.T.-L., Zhang, K., Shih, F.Y.: Exact and approximate algorithms for unordered tree matching. IEEE Trans. on Systems, Man, and Cybernetics 24, 668–678 (1994)
9. Tai, K.-C.: The tree-to-tree correction problem. Journal of the ACM 26, 422–433 (1979)
10. Tsukuda, K., Nakamura, S., Yamamoto, T., Tanaka, K.: Recommendation of addition and deletion ingredients based on the recipe structure and its stability for exploration of recipes. IEICE Trans. J94-A, 476–487 (2011) (in Japanese)
11. Yamakata, Y., Kakusho, K., Minoh, M.: A method of recipe to cooking video mapping for automated cooking content construction. IEICE Trans. Inf. & Syst. J90-D, 2817–2829 (2007) (in Japanese)
12. Yoshino, K., Mori, S., Kawahara, T.: Predicate argument structure analysis using partially annotated corpora. In: Proc. of the Sixth International Joint Conference on Natural Language Processing (to appear)
13. Zhang, K., Jiang, T.: Some MAX SNP-hard results concerning unordered labeled trees. Information Processing Letters 49, 249–254 (1994)
14. COOKPAD (May 29, 2013), `http://cookpad.com/` (in Japanese)