# A METHOD FOR EXTRACTING MAJOR WORKFLOW COMPOSED OF INGREDIENTS, TOOLS, AND ACTIONS FROM COOKING PROCEDURAL TEXT

†*Yoko Yamakata* *Shinji Imahori* *Hirokuni Maeta* *Shinsuke Mori*

†The University of Tokyo Chuo University Cybozu, Inc. Kyoto University
†7-3-1, Hongo, Bunkyo-ku, Tokyo, Japan., yamakata@hal.t.u-tokyo.ac.jp

## ABSTRACT

In this paper, we propose a method for extracting a major workflow of cooking procedure from a Japanese recipe on the Web. It is utilized for various applications including recipe search, summarization, and visualization. This major workflow is represented as a tree in which each leaf corresponds to a food or a cooking tool, each intermediate node corresponds to a cooking action, and the root node corresponds to the final cooking action. We had already developed a method for converting a procedural text to a workflow as a directed acyclic graph, and the proposed method is based on the previous one. The results are evaluated by comparing with manually extracted major workflows as a ground truth.

***Index Terms***— Cooking recipe processing, procedural workflow extraction, procedural text processing

## 1. INTRODUCTION

Cooking is one of the most significant but complicated activities for human life. Because it is manufacturing process including parallel processing and using various types of foods and tools, structural representation is useful for explaining its procedures.

Even though a recipe text explains its procedure with a sequence of steps, a chef not always cooks them sequentially in practical scene. Suppose a recipe text says "boil a pasta" as "Step 1" and "make a sauce" as "Step 2." A chef should not do Step 1 and then Step 2 sequentially but should do both steps simultaneously to complete the cooking in a short time. It is extremely important skill for a chef to understand which process should be performed simultaneously but it is difficult especially for a beginner.

Cooking actions that should be processed simultaneously can be easily found using workflow representation. In cooking, multiple foods come together into one product with having several cooking processes, and cooking processes toward a food should be completed before the food is mixed with others. Since all cooking actions which are performed to the same ingredient are always aligned on the path from a food to a root, combinations of processes that can be performed simultaneously are easily found as subgraphs that hang from one node.

Part of this research was supported by JSPS KAKENHI Grant Numbers 26280039 and 26280084.

According to our study, workflow of most of cooking recipes can be represented as a tree if we focus on the major stream composed of foods, tools, and actions. Trees can adopt various kind of effective algorithm such as computing the editing distance. Therefore, we aim to construct a system that extracts the major workflow as a tree whose leaves correspond to foods or tools and whose intermediate nodes correspond to cooking actions from the procedural text. Hereafter, we call the tree formatted workflow as a "recipe tree."

In general, a procedural text of a recipe is representable as a directed acyclic graph (DAG) so that the information described in the text is included as much as possible [1]. Maeta et al. created a corpus of 208 Japanese recipes manually (r-FG corpus), and constructed a dependency parser specialized for cooking recipes using machine learning techniques [2]. However, in this representation, not only foods, tools, and actions but other types of elements such as quantities, status of food or tool, conditions are also included and a food or a tool, which should be appeared as a leaf, can be appeared as an intermediate node when it has co-reference relation with the other element. This rich representation often makes a similarity calculation more complicated. Therefore, in this paper, we target to extract the major workflow tree composed of only foods and tools as leaves and cooking actions as intermediate nodes.

## 2. RELATED WORK

Workflow representation of cooking procedure is utilized for various applications for cooking support including recipe search, summarization, task scheduling, and visualization [3, 4, 5]. It is useful to see at a glance how the cooking process is going on. A cooking recipe service named "Nestle Valance Recipe" provides a flow-graph as a visual instruction of a cooking procedure[1]. They insist that flow-graph representation has an advantage for instructing the procedure with such a small display as a smart phone.

[1]http://m.nestle.jp/wellness/appli/04.php

Workflow is also available for schedule optimization. In general, people cook multiple dishes at the same time and it is desirable all dishes to be completed at the same time in a short time. Hamada et al. constructed a scheduling system "Cooking Navi" that schedules a cooking order based on workflow represented recipes when a chef specifies multiple dishes and inputs available cookwares and appliances in a kitchen [5].

On the other hand, workflow representation is unusual and needs training to write. To utilize existing millions recipes, an automatic method for extracting such a workflow from a procedural text must be required. In previous researches, Hamada et al. proposed an automatic workflow extraction [6] and then Karikome et al. [7] improved their methods. In their methods, they adopt general dependency parsing technique to extract a structure from each sentence of a procedural text, and then combined the structures into one using rule-based approach. For English recipes, Walter et al. also proposed a method for workflow extraction from a recipe by Java Annotation Patterns Engine (JAPE) [8] . JAPE provides finite state transduction over annotations based on regular expressions and it is useful for semantic extraction with pattern-matching operation. If it is assumed that a recipe dataset is organized well by professional authors and editors, writing formats are unified and such rule-based approaches work well. However, most recipes on the Web are generated by an ordinal author. Since writing style and wording differs for each recipe and even for each sentence of one recipe, and also they contain a lot of errors, we consider that rule-based approaches does not work constantly.

We had proposed a method to calculate dependency likelihood using corpus based machine leaning technique [2]. The advantages of their method is that it can easily improve the accuracy by increasing corpus and can consistently support new writing style and wording by adding related corpus. Their method composed of three parts; a named entity recognition, dependency likelihood calculation, and DAG extraction. The proposed method in this paper uses the dependency likelihoods calculated by this method. See Sec. 4 for details.

While we focus on Japanese recipes and adopted structure analysis method for Japanese ones, machine leaning based structure analysis for English recipes were also proposed by Kiddon et al. [9] and Jermsurawong and Habash [10].

Once a workflow is extracted from a recipe text, various kinds of effective algorithms for graph analysis are adaptable. We had proposed a method for feature extraction and summarization based on such recipe tree [3, 4]. In this research, the recipe tree in this research was extracted manually. Wang et al. [11] proposed a graphical representation of the cooking procedure, called a "cooking graph," in which ingredients and actions are modeled as nodes, which are connected by directed edges that indicate the ingredient and cooking flows. Then Xie et al. [12] proposed a recipe search method based on the cooking graph representation. In their work, the cooking graph was built manually from recipe texts.

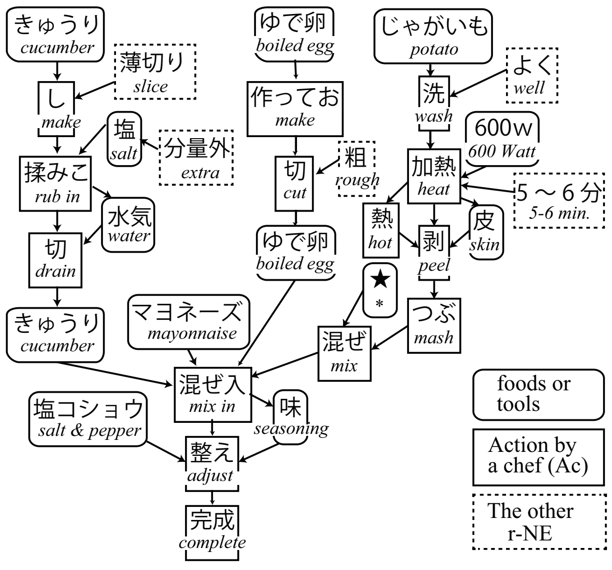

**Fig. 1**. Manually generated procedural workflow as a DAG.

## 3. MAJOR WORKFLOW EXTRACTION FROM COOKING PROCEDURAL TEXT

Figure 1 shows an example of procedural workflow of the recipe flowgraph corpus (r-FG corpus) [1]. In this figure, a label of each node shows its occurrence number and description. A round square node corresponds to a F̲ood or a T̲ool, an square node corresponds to an A̲ction c̲hef, and a square node with dotted line corresponds to other types of nodes. Arcs between two nodes show the types of relationship between these nodes. In most cases, all flows meet together and come to one node in the last when the procedures are completed.

The target recipe tree that is extracted from Figure 1 is shown in Figure 2. The differences between the workflow and this recipe tree are as follows;

- Nodes are composed of foods (F), tools (T), and chef's action (Ac) only.
  We considered that these three types of nodes are important to recognize the major cooking workflow.

- Foods (F) and tools (T) are appeared only as leaves.
  In the original DAG workflow, foods and tools appears also as intermediate nodes, but they are used as co-references on the description and omissible.

- Arcs are composed of "Targ", "Dest", "T-comp", and "F-comp" only (see the manings of the labels on Table 2). "Agent" and "V-tm" are mostly used for referring condition of actions and is not required to understand the major workflow. "F-eq", "F-part-of", "F-set", "T-eq", "T-part-of", and "A-eq' are mostly used for co-reference relationship so these are also omissible.

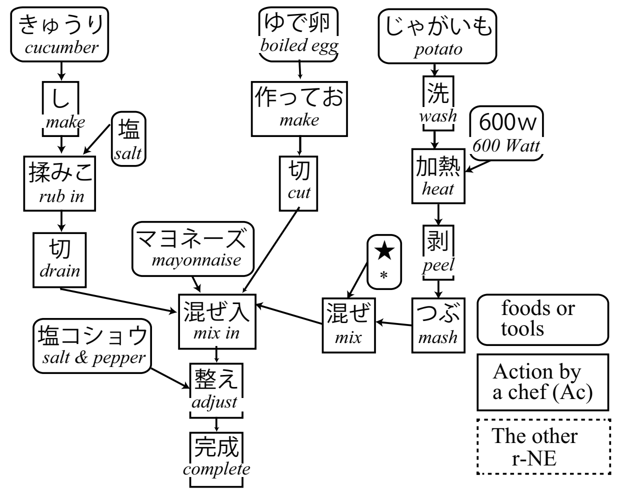

**Fig. 2**. Recipe tree of workflow of Fig. 1.

Major workflows of most of recipes are representable as trees except such cases when one ingredient is divided into two or more, different actions are performed to each one, and mixed again (for example, an egg is divided into the white and yolk, whip the white, and mix them together), and when more than two dishes are cooked from a single ingredient (for example, meringue and pudding are cooked from an egg.) Because such cases appear less than five percent in our previous survey, we address tree representable workflow only in this paper.

The processes of generating the major workflow from a procedural text are as follows;

**Step 1. A complete graph with edge costs is estimated from a procedural text (Sec. 4)**

Extract important recipe terms as nodes from a procedural text and calculate the weight between all combinations and both directions of two of the extracted terms. The weight is small when the dependency of two terms is strong. In this step, one complete graph is estimated from one recipe.

**Step 2. A recipe tree extraction from the complete graph (Sec. 5)**

As a recipe tree, the optimal tree is extracted from the complete graph as a spanning arborescence of minimum cost in which the root corresponds to the final cooking action. In this process, several heuristic operations are introduced so that the extracted tree satisfies the conditions mentioned above.

## 4. COMPLETE GRAPH ESTIMATION FROM A PROCEDURAL TEXT

This process consists of the following three processes combined in the cascaded manner.

**Table 1**. Node labels of recipe flow graphs.

| Used | Type Tag | Meaning |
|---|---|---|
| Y | F | Food |
| Y | T | Tool |
| N | D | Duration |
| N | Q | Quantity |
| Y | Ac | Action by the chef |
| N | Af | Action by foods |
| N | Sf | State of foods |
| N | St | State of tools |

**Table 2**. Edge labels of recipe flow graphs.

| Used | Arc label | Meaning |
|---|---|---|
| N | Agent | Action agent |
| Y | Targ | Action target |
| Y | Dest | Action destination |
| Y | F-comp | Food complement |
| Y | T-comp | Tool complement |
| N | F-eq | Food equality |
| N | F-part-of | Food part-of |
| N | F-set | Food set |
| N | T-eq | Tool equality |
| N | T-part-of | Tool part-of |
| N | A-eq | Action equality |
| N | V-tm | Head of a clause for timing |
| N | other-mod | Other relationships |

1. Word segmentation
2. Recipe term identification
3. Edge weight estimation

In this section we explain these processes one by one.

### 4.1. Word Segmentation

Japanese, the target language of this paper, has no obvious word boundary like whitespace in English. Thus first we identify words in the texts. This process is called word segmentation (WS).

For WS we use a text analysis tool `KyTea` [13] because its domain adaptation is easy.[2] We added the annotated sentences in the flow graph corpus to the training data and retrained the model. The accuracy on the small portion of the corpus is more than 0.97, which is comparable to that of the default model on the general domain.

### 4.2. Recipe Term Identification

The second process is the recipe term identification. A recipe term is a span of words without overlap annotated with its

[2]`KyTea` is available with the default model for the general domain from `http://www.phontron.com/kytea/` (accessed on 13 May. 2016).

type (see Table 1), which will be a node in the recipe tree. The term identification problem is the same as the named entity (NE) recognition except for the tag definition. To solve the problem, Sasada et. al. adopt one of the state-of-the-art NE recognizers, `PWNER`[3], and trained it from the sentences annotated with recipe named entities (r-NEs) in the flow graph corpus. To increase the accuracy, Sasada et. al. annotated other 208 sentences with r-NEs and achieved the accuracy of 0.94 (F-measure).

## 4.3. Edge Weight Estimation

To a recipe text with r-NEs identified we apply the following method to form a complete graph with edge weights.

The weight of an edge of a certain label $l$ between two r-NEs, $u$ and $v$, is defined as follows:

$$s(u, v, l) = \frac{\exp\{\mathbf{\Theta} \cdot \boldsymbol{f}(u, v, l)\}}{\sum_{(x,r)\in(V\setminus\{v\})\times L} \exp\{\mathbf{\Theta} \cdot \boldsymbol{f}(u, x, r)\}},$$

where $L$ is the set of edge labels (see Table 2), $\mathbf{\Theta}$ is the weight vector, and $\boldsymbol{f}(u, v, l)$ is a function returning the feature vector for the edge. The features are the word sequences of $u$ and $v$, their r-NE types, their surrounding words, etc.

The values of the weight vector $\mathbf{\Theta}$ are estimated by a log-linear model [14] referring to the training data. Given $T$ training instances $\{(V_t, u_t, v_t, l_t)\}_{t=1}^{T}$, where $V_t$ is the set of the nodes and each $(u_t, v_t, l_t)$ is the pair of the edge and the label annotated manually, we estimate the values of the weight vector $\mathbf{\Theta}$ maximizing the following value:

$$\sum_{t=1}^{T} \left\{ \log \frac{\exp(\mathbf{\Theta} \cdot \boldsymbol{f}(u_t, v_t, l_t))}{\sum\limits_{(x,r)\in(V_t\setminus\{u_t\})\times L} \exp(\mathbf{\Theta} \cdot \boldsymbol{f}(u_t, x, r))} \right\} - \frac{1}{2}\|\mathbf{\Theta}\|^2.$$

## 5. RECIPE TREE EXTRACTION

From the calculated complete digraph in the previous section, the optimal workflow, whose root corresponds to the final cooking action and whose cost is the lowest in the candidates, can be found efficiently as a spanning arborescence of minimum weight [15, 16] However, this tree does not satisfy the condition define in the Section 3. Therefore, we introduce the following three operations N, E and P. In Section 6, we consider eight cases to apply or not to apply each operation. For example, N1E0P1 is the case that operations N and P are applied, but operation E is not applied.

**Operation N: Remove needless nodes from a given complete graph in advance**

As defined in Sec. 3, a target tree is composed of nodes whose labels are "F", "T" and "Ac" only. (More precisely, leaf nodes have labels "F" or "T", and intermediate nodes have label "Ac".) Therefore, we eliminate needless nodes whose labels are "D", "Q", "Af", "Sf" and "St" from the given complete graph and then compute a spanning arborescence of minimum weight from it. In Sec. 6, "N1" means this operation is applied; "N0" otherwise.

[3] `PWNER` is available from `http://plata.ar.media.kyoto-u.ac.jp/tool/PWNER/` (accessed on 13 May. 2016).

**Operation E: Remove needless edges from a given complete graph in advance**

As defined in Sec. 3, a target tree is composed of edges whose labels are "Targ", "Dest", "F-comp" and "T-comp" only. Therefore, we eliminate needless edge whose labels are "Agent", "F-eq", "F-part-of", "F-set", "T-eq", "T-part-of", "A-eq", "V-tm", and "other mod" and then compute a spanning arborescence of minimum weight from it.

**Operation P: Remove needless nodes and needless edges from the calculated spanning arborescence**

After calculating a spanning arborescence of minimum weight, needless nodes and needless edges are eliminated from the arborescence according the following procedures;

1. Remove nodes whose labels are neither "F", "T" nor "Ac".
   When we remove a leaf node, we also remove its adjacent edge. When we remove an internal node, we remove all the adjacent edges, and add edges from its child nodes to its parent node. As the result, a tree whose node labels are "F", "T" and "Ac" only is induced. (n.b., if we have applied operation N, this step is skipped.)
2. Remove internal nodes whose labels are "F" or "T".
   When we remove an internal node, we remove all the adjacent edges, and add edges from its child nodes to its parent node. After this step, we have a tree whose internal nodes have label "Ac" only

Although the labels of the leaf nodes of the target tree must be "F" or "T" only according to the definition in the Sec. 3, the resulting tree can have "Ac" as a leaf node. This situation may occur when it fails to find a connection from the "Ac" node to its target "F" or "T" node. Also the resulting tree can have edges whose labels are neither "Targ", "Dest", "F-comp" nor "T-comp" if we do not apply operation E. However, we do not remove such edges to keep the connectivity of a graph.

**Discussion**

It is sometimes very clear which terms of a recipe text should be leaf nodes in the resulting tree. For example, when an ingredient first appears in a procedural text, it should be a leaf node of the resulting tree in most cases. From this perspective, it might have better performance if we calculate a minimum weight arborescence under the specified leaf nodes beforehand. However, this problem is NP-hard (it is easy to show the NP-hardness of the problem with Hamiltonian path

**Table 3**. The specitications of recipe flow graph corpus.

| #Recipes | #Sent. | #Terms | #Words |
|---|---|---|---|
| 208 | 1,374 | 8,316 | 25,446 |

problem, a standard NP-hard problem) and will not be solved efficiently.

## 6. EXPERIMENTAL EVALUATION

### 6.1. Flow Graph Corpus

In the experimental evaluation, we used recipe flow graph corpus (r-FG corpus) [1]. r-FG corpus contains 208 recipes randomly extracted from an Internet site storing recipes posted by users. The recipes are annotated with the following information.

1. Word boundary: Each sentence is segmented into words. The definition of words is super short word unit (SSUW), which is a variation of short word unit [17] with a modification of inclectional ending separation.
2. Recipe term: Important word sequences are marked with their types (see Table 1).
3. Flow graph: Each recipe is annotated with a rooted DAG, whose nodes are recipe terms and the labeled edges (see Table 2) are relationships between them. The root corresponds to the final cooking action.

Table 3 shows specifications of the r-FG corpus.

### 6.2. Recipe Tree Estimation

As we mentioned in Section 4, the accuracies of word segmentation and term recognition are very high. Therefore, we used the manual annotation results as the input of the complete graph estimation process instead of running WS and r-NE identification in the experiments. We divided the r-FG corpus into 10 parts and used nine of ten parts as a training data and used one of ten part as a test data.

### 6.3. Test Set and Evaluation Method

We picked up ten recipes from the test data and generated ground truth by converting them to recipe trees manually. The average number of nodes on the ground truths is 22.7. An accuracy of an extracted recipe tree was evaluated by calculating the editing distance between the extracted tree and its ground truth. Note that the sibling order is ignored in a recipe tree and calculating editing distance of such an unordered tree is NP-hard problem in general. Therefore, we adopt a fast calculation method specialized for recipe tree that we had proposed in [3]. All operation costs of substitution, insertion, and deletion were set as one in the following experiment.

### 6.4. Experiment and Discussions

We evaluate our method for all combinations of operations N, E and P. The editing distances of ten recipe "A" to "J" are shown in the Figure 3.

As shown in the table, the average distances of the results with the operation P were always shorter than the results without the operation P when the other conditions were the same. It means that removing intermediate nodes except "Ac" is always effective.

The shortest average distance was obtained by the method "N1E0P1." It means that the most accurate tree is extracted when it removed nodes except "F", "T", and "Ac" and then extracted a spanning arborescence of minimum weight using all types of edges from a complete graph. However, the method "N0E0P1" also obtained almost the same average distance with "N1E0P1." "N0E0P1" obtained shorter distances in three of ten recipes while "N1E0P1" obtained shorter distances in three of ten recipes, and both obtained the same distances in the rest four recipes.

According to more detailed survey, in such cases that two "Ac" nodes were connected via node "Sf" and "Af", the method "N0E0P1" was more accurate because it failed to extract the correct relationship between the two "Ac" without in-between node "Sf" and "Af"." For example, a recipe says "fry an onion. when the onion is heated well, add a cup of water," it failed to find the relationship between "fry" and "add" because the sentence "the onion is heated" between these two "Ac" was lost. On the other hand, in such cases that an original DAG workflow contained a divergent flow, the method "N1E0P0" tended to be more accurate. The most divergent flows are composed of the major flow and sub flows. Because a sub flow tends to be composed of nodes except for "F", "T", and "Ac," the major flow was extracted easier when the nodes except for "F", "T", and "Ac" were removed. In the test data, these two cases are occurred at the same frequency and the accuracies of these two methods got equal. As a future work, We will evaluate the benefit of each algorithm with larger dataset and find out a way of automatically selecting better method for each recipe.

## 7. CONCLUSION

In this paper, we proposed a method to extract the major workflow that is a tree composed of foods, tools, and cooking actions from a procedural text. The average editing distance between manually extracted recipe trees and automatically extracted recipe trees by the proposed method was 13.9 when the average number of node of the test set was 22.7 and the substitution, deletion, and insertion costs were set as one. Because the most suitable algorithm varied according to a recipe, it is a future work to develop technique to select algorithm automatically.

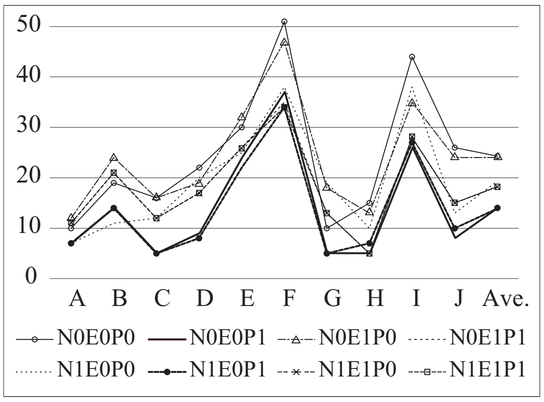

**Fig. 3**. Editing distances between ground truths and automatically extracted recipe tree.

## 8. REFERENCES

[1] Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, and Tetsuro Sasada, "Flow graph corpus from recipe texts," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, 2014, pp. 2370–2377.

[2] Hirokuni Maeta, Tetsuro Sasada, and Shinsuke Mori, "A framework for procedural text understanding," in *Proceedings of the 14th International Conference on Parsing Technologies*, 2015.

[3] Yoko Yamakata, Shinji Imahori, Yuichi Sugiyama, Shinsuke Mori, and Katsumi Tanaka, "Feature extraction and summarization of recipes using flow graph," in *SocInfo*, 2013, vol. 8238 of *LNCS*, pp. 241–254.

[4] Yoko Yamakata, Koh Kakusho, and Michihiko Minoh, "Object recognition based on object's identity for cooking recognition task," in *CEA*, 2010, pp. 278–283.

[5] 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 13th annual ACM international conference on Multimedia*, 2005, MULTIMEDIA '05, pp. 371–374.

[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*, 2000, IRAL '00, pp. 157–164.

[7] Shihono Karikome and Atsushi Fujii, "Improving structural analysis of cooking recipe text," *IEICE technical report. Data engineering*, vol. 112, no. 75, pp. 43–48, 2012.

[8] Kirstin Walter, Mirjam Minor, and Ralph Bergmann, "Workflow Extraction from Cooking Recipes," in *Proceedings of the Computer Cooking Contest*, 2011, pp. 207–216.

[9] Chloe Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi, "Mise en place: Unsupervised interpretation of instructional recipes," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 982–992.

[10] Jermsak Jermsurawong and Nizar Habash, "Predicting the structure of cooking recipes," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 781–786.

[11] 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*, 2008, WWW '08, pp. 979–988.

[12] Haoran Xie, Lijuan Yu, and Qing Li, "A hybrid semantic item model for recipe search by example," in *2010 IEEE International Symposium on Multimedia (ISM)*, 2010, pp. 254–259.

[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, pp. 529–533.

[14] Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra, "A maximum entropy approach to natural language processing," *Computational Linguistics*, vol. 22, no. 1, pp. 39–71, 1996.

[15] Jack Edmonds, "Optimum Branchings," *Journal of Research of the National Bureau of Standards*, vol. 71B, pp. 233–240, 1967.

[16] Harold N. Gabow, Zvi Galil, Thomas Spencer, and Robert E. Tarjan, "Efficient algorithms for finding minimum spanning trees in undirected and directed graphs," *Combinatorica*, vol. 6, no. 2, pp. 109–122, 1986.

[17] Kikuo Maekawa, Makoto Yamazaki, Takehiko Maruyama, Masaya Yamaguchi, Hideki Ogura, Wakako Kashino, Toshinobu Ogiso, Hanae Koiso, and Yasuharu Den, "Design, compilation, and preliminary analyses of balanced corpus of contemporary written Japanese," in *Proceedings of the Seventh International Conference on Language Resources and Evaluation*, 2010.