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. 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.

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

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:

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

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.

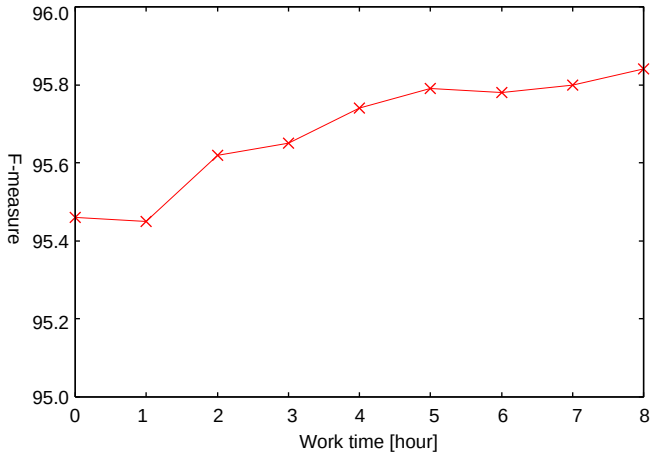

Figure 4. Learning curve of word segmentation.

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

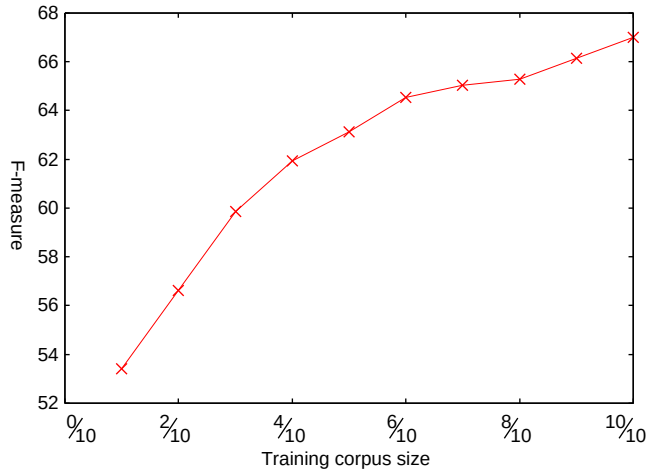

Figure 5. Learning curve of named entity recognition.

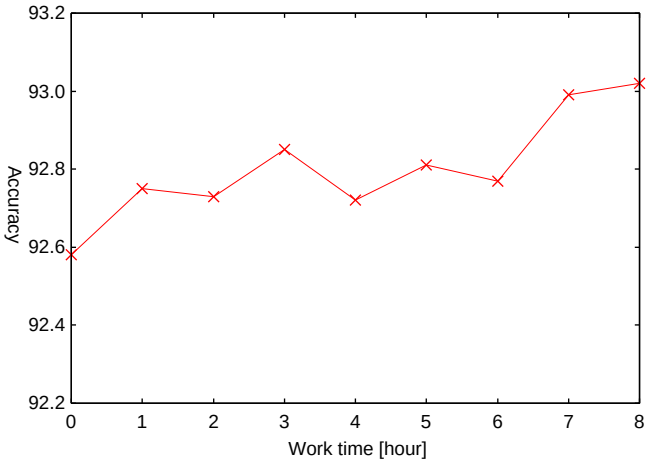

Figure 6. Learning curve of syntactic analysis.

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).