# Named Entity Recognition by Character-based Word Classification using a Domain Specific Dictionary

Makoto Hiramatsu[1], Kei Wakabayashi[1], and Jun Harashima[2]

[1] University of Tsukuba, Tsukuba, Ibaraki, 305–8550, Japan, JAPAN
[2] Cookpad Inc., Ebisu, Shibuya-ku, Tokyo, 150–6012, JAPAN

**Abstract.** Named entity recognition is a fundamental task in natural language processing and has been widely studied. The construction of a recognizer requires training data that contains annotated named entities. However, it is expensive to construct such training data for low-resource domains. In this paper, we propose a recognizer that uses not only training data but also a domain specific dictionary that is available and easy to use. Our recognizer first uses character-based distributed representations to classify words into categories in the dictionary. The recognizer then uses the output of the classification as an additional feature. We conducted experiments to recognize named entities in recipe text and report the results to demonstrate the performance of our method.

**Keywords:** Named entity recognition · recipe text · neural network

## 1 Introduction

Named entity recognition (NER) is one of the fundamental tasks in natural language processing (NLP) [20]. The task is typically formulated as a sequence labeling problem, for example, estimating the most likely tag sequence $Y = (y_1, y_2, \ldots, y_n)$ for a given word sequence $X = (x_1, x_2, \ldots, x_n)$. We can train a recognizer using annotated data that consists of $(X, Y)$.

However, the construction of such annotated data is labor-intensive and time-consuming. Although the beginning, inside, and outside (BIO) format is often used in NER, it is challenging to annotate sentences with tags, particularly for people who are not familiar with NLP. Furthermore, there are low-resource domains that do not have a sufficient amount of data. We focus on the recipe domain as an example of such a domain.

Even in such domains, we can find a variety of dictionaries available. For example, Nanba et al. [13], constructed a cooking ontology, Harashima et al. [3] constructed a dictionary for ingredients, and Yamagami et al. [21]

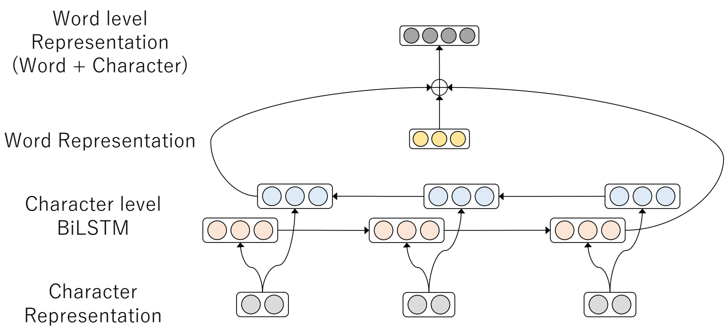


**Fig. 1.** LSTM-CRF based neural network.

built a knowledge base for basic cuisine. These resources can be utilized for NER.

In this paper, we propose a method to integrate a domain-specific dictionary into a neural NER using a character-based word classifier. We demonstrate the effectiveness of the proposed method using experimental results on the Cooking Ontology dataset [13] as a dictionary. We report our experimental results on the recipe domain NE corpus [12].

## 2 Related Work

In recent years, NER methods that use long short-term memory (LTSM) [4] and conditional random fields (CRF) [7] have been extensively studied [8, 9, 19, 15, 10]. This type of neural network is based on Huang et al. [5]. Note that they used Bidirectional LSTM (Bi-LSTM), which concatenate two types of LSTM; one is forward LSTM, and another is backward LSTM. In these studies, the researchers assumed that training data with sequence label annotation was provided in advance.

In our experiments, we use recipe text as a low-resource domain to evaluate our proposed method. Although Mori et al. [12] constructed an r-NE corpus, it consists of only 266 Japanese recipes. To overcome this problem, Sasada et al. [18] proposed an NE recognizer that is trainable from partially annotated data. However, as seen in Section 5.4, the method does not perform better than recent neural network-based methods.

Preparing training data for NER is time-consuming and difficult. In addition to the strategy that uses partial annotation, there have been attempts to make use of available resources. Peters et al. [15, 16] acquired informative features using language modeling. However, these approaches require a large amount of unlabeled text for training, which makes it difficult in a low-resource scenario. To avoid this difficulty, making use of a task that does not require a large amount of data could be useful.

Whereas it is time-consuming to prepare training data for NER, it is relatively easy to construct a domain-specific dictionary [1, 13, 21]. Some researchers have used a dictionary as an additional feature [19, 10]. Pham et al. [10] incorporated dictionary matching information as additional dimensions of a feature vector of a token. In their method, the representations are zero vectors for words that are not in the dictionary. Our proposed method overcomes this limitation by extracting character-based features from a classifier trained on a dictionary.

## 3 Baseline Method

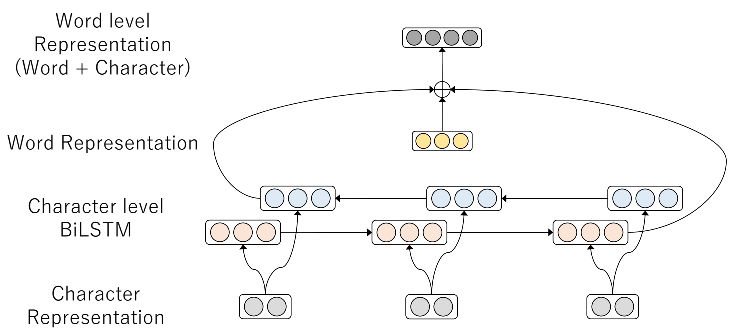


**Fig. 2.** Word-level feature extractor proposed by Lample et al. [8]

As described in Section 2, the popular methods use Bi-LSTM (bidirectional-LSTM) and CRF, which is so called LSTM-CRF. Lample et al. [8] takes account of not only word-level but also character-level information to extract features. We show an illustration of the word-level feature extractor proposed by Lample et al. in Fig. 2. Let $X = (x_1, x_2, \ldots, x_N)$ be an input word sequence and $C_t = (c_{t,1}, c_{t,2}, \ldots, c_{t,M})$ be the character sequence of the t'th word. A word distributed representation corresponding to $x_t$ is defined by $\mathbf{v}_{x_t}$, and a character distributed representation corresponding to $C_{t,k}$ is defined by $\mathbf{v}_{C_{t,k}}$. Let $V_{C_t} = (\mathbf{v}_{C_{t,1}}, \mathbf{v}_{C_{t,2}}, \ldots, \mathbf{v}_{C_{t,M}})$. Then their model can be represented as

$$\mathbf{w}_t{}^{(char)} = \text{Bi-LSTM}^{(char)}(V_{C_t}), \tag{1}$$

$$\mathbf{x}_t = [\mathbf{w}_t; \mathbf{w}_t{}^{(char)}]. \tag{2}$$

Then, let $V_X = (\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N})$,

$$\mathbf{h}_t = \text{Bi-LSTM}(V_X)_t, \tag{3}$$

where $\mathbf{w_t}$ indicates the word representation corresponding to $x_t$.

After extracting the feature vector $\mathbf{h}_t$ of the sequence, they applied CRF to predict the tag sequence considering their tag transitions. To use CRF, $\mathbf{h}_t$ is transformed by $\mathbf{z}_t = \mathbf{W}\mathbf{h}_t + \mathbf{b}$, where $\mathbf{W}$ is the $L \times H$ weight matrix, $L$ is the number of tags, and $H$ is the size of the hidden state of word-level Bi-LSTM. Let $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ be a tag sequence. Using $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n)$, we can calculate the probability of the tag sequence using

$$P(\mathbf{y} \mid \mathbf{Z}; \mathbf{W}, \mathbf{b}) = \frac{\prod_{i=1}^{n} \psi_i(y_{i-1}, y_i, \mathbf{Z})}{\sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{Z})} \prod_{i=1}^{n} \psi_i(y'_{i-1}, y'_i, \mathbf{Z})}, \tag{4}$$

where $\psi_i(y', y, \mathbf{Z}) = \exp(W_{y',y}^T \mathbf{z}_i + \mathbf{b}_{y',y})$. $\mathbf{W}$ is the $L \times L$ weight matrix, which controls a transition of NE tags. And $\mathbf{b}$ is the $L$ dimensional bias vector corresponding to $\mathbf{W}$. What we want is the optimal tag sequence $\hat{\mathbf{y}}$, which is defined by

$$\hat{\mathbf{y}} = \underset{\mathbf{y}' \in \mathcal{Y}(\mathbf{Z})}{\text{argmax}}\, P(\mathbf{y}' \mid \mathbf{Z}; \mathbf{W}, \mathbf{b}). \tag{5}$$

We can obtain the optimal tag sequence $\hat{\mathbf{y}}$ by maximizing $P$ using the Viterbi algorithm.

## 4 Proposed Method

In this paper, we propose a recognizer that uses not only training data but also a domain-specific dictionary. As described in Section 1, it is

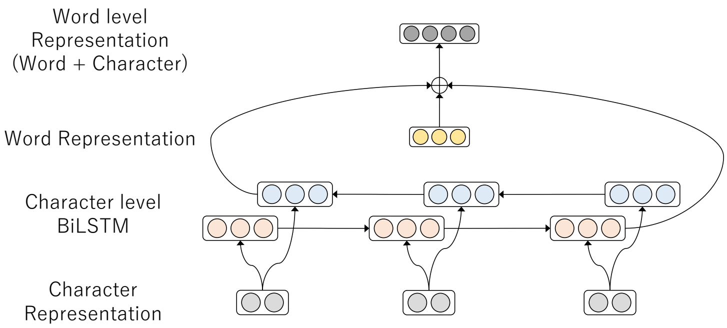


**Fig. 3.** Overview of the character-based word classifier. We use a 3 stacked Bi-LSTM.

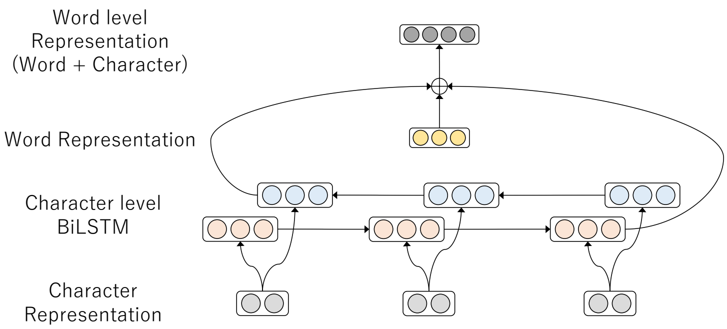


**Fig. 4.** Overview of the proposed method. We concatenate the classifier output to a feature vector from the Bi-LSTM.

expensive to construct training data for a recognizer. We thus make use of a domain-specific dictionary that contains pairs that consist of a word and category.

Fig. 4 shows the architecture of our proposed recognizer. Our recognizer can be considered as an extension of Lample et al. [8]. We incorporate the character-based word classifier which calculates $\mathbf{a}_t$ as follows:

$$\mathbf{h}^{(classifier)}{}_t = \text{Stacked Bi-LSTM}(C_t) \tag{6}$$

$$\mathbf{a'}_t = \mathbf{W}\mathbf{h}^{(classifier)}{}_t + \mathbf{b} \tag{7}$$

$$\mathbf{a}_t = \text{Softmax}(\mathbf{a'}_t), \tag{8}$$

**Table 1.** The statistics of corpora using our experiments. Note that the **r-NE corpus is annotated for NEs with the BIO format**. We show character-level information only for r-NE because it is used to train a recognizer.

| Attribute | Cookpad | Wikipedia | r-NE |
|---|---|---|---|
| Doc | 1,715,589 | 1,114,896 | 436 |
| Sent | 12,659,170 | 18,375,840 | 3,317 |
| Token | 216,248,517 | 600,890,895 | 60,542 |
| Type | 221,161 | 2,306,396 | 3,390 |
| Char token | – | – | 91,560 |
| Char type | – | – | 1,130 |

This classifier is a neural network that consists of a embedding layer, a stacked Bi-LSTM layer, and a fully connected layer. Stacked Bi-LSTM is one kind of neural network which applies Bi-LSTM $k$ times where $k > 1$. Classifier takes the character sequence of words as input and predicts categories of it defined in a dictionary. After passing the word classifier, our method concatenates the hidden state calculated in Section 3 and the output of Classifier defined by $\mathbf{h}'_t = \mathbf{h}_t \oplus \mathbf{a}_t$. Finally, as in Section 3, our method transforms $\mathbf{h}'$ by $\mathbf{z}_t = \mathbf{W}\mathbf{h}'_t + \mathbf{b}'$, and the CRF predicts the most likely tag sequence.

Although our method is simple, it has two advantages: First, our method is based on character-level distributed representations, which avoid the mismatching problem between words in the training data and words in the dictionary. Second, the method can use a dictionary with arbitrary categories that are not necessarily equal to the NE categories in the sequence labels. Consequently, our method can be applied in all scenarios in which there is a small amount of training data that contains NEs and there is a domain dictionary constructed arbitrarily.

## 5 Experiments

### 5.1 Datasets

We used the following four datasets:

**r-NE [12]** : used to train and test methods. We used 2,558 sentences for training, 372 for validation, and 387 for testing.

**Cooking Ontology [13]** : used to train the word classifier. We use 3,825 words for training, 1,000 for validation, and 1,000 for testing.

**Table 2.** r-NEs and their frequencies.

| NE | Description | # of Examples |
|---|---|---|
| F | Food | 6,282 |
| T | Tool | 1,956 |
| D | Duration | 409 |
| Q | Quantity | 404 |
| Ac | Action by the chef | 6,963 |
| Af | Action by foods | 1,251 |
| Sf | State of foods | 1,758 |
| St | State of tools | 216 |

**Table 3.** Word categories, frequencies, and results on classification.

| Category | # of Examples | Prec. | Recall | Fscore |
|---|---|---|---|---|
| Ingredient-seafood (example: salmon) | 452 | 0.60 | 0.62 | 0.61 |
| Ingredient-meat (example: pork) | 350 | 0.88 | 0.83 | 0.85 |
| Ingredient-vegetable (example: lettuce) | 935 | 0.75 | 0.79 | 0.77 |
| Ingredient-other (example: bread) | 725 | 0.75 | 0.71 | 0.73 |
| Condiment (example: salt) | 907 | 0.81 | 0.84 | 0.83 |
| Kitchen tool (example: knife) | 633 | 0.79 | 0.74 | 0.76 |
| Movement (example: cut) | 928 | 0.94 | 0.99 | 0.96 |
| Other | 896 | 0.70 | 0.66 | 0.68 |

**Cookpad [2]** : used to train word embeddings. Cookpad corpus contains 1.7M recipe texts.

**Wikipedia** : used to train word embeddings. There were various types of topics in this corpus. We downloaded the raw data of this corpus from the Wikipedia dump[3]. Wikipedia corpus contains 1.1M articles.

As in Table. 2 and Table. 3, the categories in the cooking ontology were different from the tags in the r-NE corpus. However, as described in Section 4, our method flexibly incorporated such information into its network.

### 5.2 Methods

We compared the following methods in our experiments:

**Sasada et al. [18]** is a pointwise tagger. They use Logistic Regression as the tagger.

[3] `https://dumps.wikimedia.org/jawiki/`

**Table 4.** Results on NER (averaged over five times except for Sasada et al. [18] because KyTea [14], the text analysis toolkit used in their experiments, does not have the option to specify a random seed).

| Method | Decoder | Embedding | Prec. | Recall | Fscore |
|---|---|---|---|---|---|
| Sasada et al. | – | – | 82.34 | 80.18 | 81.20 |
| Sasada et al. | DP | – | 82.94 | 82.82 | 82.80 |
| Lample et al. | CRF | Uniform | 82.59 (± 0.94) | 88.19 (± 0.25) | 85.24 (± 0.46) |
| Lample et al. | CRF | Cookpad | 84.54 (± 1.22) | 88.47 (± 0.69) | 86.40 (± 0.89) |
| Lample et al. | CRF | Wikipedia | 85.31 (± 0.67) | 88.22 (± 0.65) | 86.68 (± 0.47) |
| Proposed | CRF | Uniform | 82.81 (± 0.88) | 88.40 (± 0.41) | 85.46 (± 0.58) |
| Proposed | CRF | Cookpad | 85.08 (± 1.30) | 88.46 (± 0.18) | 86.68 (± 0.71) |
| **Proposed** | CRF | **Wikipedia** | **85.63 (± 0.52)** | **88.87 (± 0.37)** | **87.18 (± 0.34)** |

**Table 5.** Results on NER (averaged over five times except for Sasada et al. [18] because KyTea [14], the text analysis toolkit used in their experiments, does not have the option to specify a random seed).

| Method | Decoder | Embedding | Prec. | Recall | Fscore |
|---|---|---|---|---|---|
| Sasada et al. [18] | – | – | 82.34 | 80.18 | 81.20 |
| Sasada et al. [18] | DP | – | 82.94 | 82.82 | 82.80 |
| Lample et al. [8] | CRF | Uniform | 82.59 (± 0.94) | 88.19 (± 0.25) | 85.24 (± 0.46) |
| Lample et al. [8] | CRF | Cookpad | 84.54 (± 1.22) | 88.47 (± 0.69) | 86.40 (± 0.89) |
| Lample et al. [8] | CRF | Wikipedia | 85.31 (± 0.67) | 88.22 (± 0.65) | 86.68 (± 0.47) |
| Dictionary | CRF | Uniform | 82.36 (± 1.25) | 88.28 (± 0.25) | 85.18 (± 0.71) |
| Dictionary | CRF | Cookpad | 83.91 (± 1.21) | 88.60 (± 0.41) | 86.16 (± 0.72) |
| Dictionary | CRF | Wikipedia | 85.44 (± 1.04) | 87.67 (± 0.25) | 86.50 (± 0.56) |
| Proposed | CRF | Uniform | 82.81 (± 0.88) | 88.40 (± 0.41) | 85.46 (± 0.58) |
| Proposed | CRF | Cookpad | 85.08 (± 1.30) | 88.46 (± 0.18) | 86.68 (± 0.71) |
| **Proposed** | CRF | **Wikipedia** | **85.63 (± 0.52)** | **88.87 (± 0.37)** | **87.18 (± 0.34)** |

**Sasada et al. [18]+DP** is an extension of LR, which optimizes LR's prediction using dynamic programming. This method achieved state-of-the-art performance for the r-NE task.

**Lample et al. [8]** is an LSTM-CRF tagger described in Section 2.

**Dictionary** is an LSTM-CRF based naive baseline that uses a dictionary. A dictionary feature is added to Lample's feature in the form of a one-hot vector.

**Proposed** is the proposed method that uses the character-level word classifier described in Section 4.

### 5.3 Pre-trained Word Embeddings

In NLP, a popular approach is to make use of pre-trained word embeddings to initialize parameters in neural networks. In this paper, three strategies are used to initialize word vectors:

**Uniform** initializes word vectors by sampling from the uniform distribution over $[\frac{-3}{\mathrm{dim}}, \frac{3}{\mathrm{dim}}]$.

**Wikipedia** initializes word vectors using those trained on the Wikipedia corpus. Word vectors not in pre-trained word vectors are initialized by **Uniform**.

**Cookpad** initializes word vectors using those trained on the Cookpad corpus. Word vectors not in pre-trained word vectors are initialized by **Uniform**.

We use train word embeddings with skip-gram with negative sampling (SGNS) [11]. As the hyperparameter of SGNS, we set 100 as the dimension of the word vector, 5 for the size of the context window, and 5 for the size of negative examples, and use default parameters defined in Gensim [17] for other parameters.

In our proposed network, we set 50 dimensions for character-level distributed representations and $2 \times 50$ for character-level Bi-LSTM as a word classifier. The word feature extracted by the word classifier is concatenated with the word-level representation and fed into the word-level Bi-LSTM to obtain the entire word features. To train neural networks, we use the Adam optimizer [6] with mini-batch size 10 and clip gradient with threshold 5.0.

### 5.4 Experimental Results and Discussion

Table. 3 shows the performance of our word classifier. Our classifier successfully classified words with a certain degree of accuracy. We show the results of comparing each recognizer in Table. 5 In our experiments, **(i) pre-trained word vectors played an essential role in improving the performance of NER** and **(ii) our classifier enhanced the performance of the Lample's method**. Interestingly, we obtained the best result when pre-trained word vectors were trained on the Wikipedia corpus, which is not a domain-specific corpus. This suggests that our method to have successfully combined universal knowledge from pre-trained word

**Table 6.** Results on named entity recognition (for each NE, averaged over five times).

| NE | Precision | Recall | Fscore |
|---|---|---|---|
| Ac | 91.77 (± 1.02) | 95.23 (± 0.42) | 93.46 (± 0.33) |
| Af | 78.87 (± 3.68) | 78.12 (± 1.19) | 78.46 (± 2.22) |
| D | 96.63 (± 1.71) | 93.88 (± 2.88) | 95.23 (± 2.16) |
| F | 85.84 (± 0.94) | 89.01 (± 0.65) | 87.39 (± 0.59) |
| Q | 58.70 (± 3.81) | 70.00 (± 3.19) | 63.69 (± 1.82) |
| Sf | 75.12 (± 4.40) | 78.17 (± 1.95) | 76.52 (± 2.04) |
| St | 66.03 (± 5.64) | 52.63 (± 4.70) | 58.46 (± 4.52) |
| T | 82.53 (± 2.30) | 89.09 (± 1.26) | 85.66 (± 1.21) |

vectors and domain-specific knowledge from the classifier trained on a domain-specific dictionary.

We show the label-wise results of prediction in Table. 6. In this result, we can see that the proposed model successfully predicted tags of Ac, D, F, and T. However, prediction performances for Af, Q, Sf, and St were limited because there is no entry for these categories in our dictionary.

**Fig. 5.** Prediction results for an example

| | | | | | |
|---|---|---|---|---|---|
| Example | Yoji | de | to | me | te |
| Translation | Cocktail stick | DAT | | clip | |
| Ground Truth | B-T | O | B-Ac | O | O |
| Baseline | B-Sf | O | B-Ac | O | O |
| **Proposed method** | **B-T** | O | B-Ac | O | O |

**Fig. 6.** Prediction results for another example

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Example | Denshi | renji | ( | 500 | W | ) | de |
| Translation | Microwave | | ( | 500 | W | ) | DAT |
| Ground Truth | B-T | I-T | O | B-St | I-St | O | O |
| Baseline | B-T | I-T | O | B-T | I-T | O | O |
| **Peoposed method** | B-T | I-T | O | **B-St** | **I-St** | O | O |

Fig. 5 and Fig. 6 show prediction results for the baseline and our methods. Note that the abbreviation DAT means dative. In the first example, the word classifier taught the model that **cocktail stick** was a kitchen tool, which made the proposed method successfully recognize it as a tool. In the second example, the word classifier taught the model that **500W** is not a kitchen tool. Then, the proposed method avoided the baseline's failure and estimate the correct NE tag sequence.

## 6 Conclusion

We proposed a recognizer that is trainable from not only annotated NEs but also a list of examples for some categories related to NE tags. The proposed method uses the output of a character-based word classifier. Thanks to this character-based modeling, the proposed method considers sub-word information to extract dictionary features for words not in the dictionary.

Our experiment demonstrates that our method achieves state-of-the-art performance on the r-NE task. This implies that the proposed method successfully extracts an informative feature to improve the performance of NER.

# Bibliography


[1] Chung, Y.j.: Finding food entity relationships using user-generated data in recipe service. In: Proceedings of International Conference on Information and Knowledge Management. pp. 2611–2614 (2012)

[2] Harashima, J., Michiaki, A., Kenta, M., Masayuki, I.: A Large-scale Recipe and Meal Data Collection as Infrastructure for Food Research. In: Proceedings of International Conference on Language Resources and Evaluation. pp. 2455–2459 (2016)

[3] Harashima, J., Yamada, Y.: Two-step validation in character-based ingredient normalization. In: Proceedings of Joint Workshop on Multimedia for Cooking and Eating Activities and Multimedia Assisted Dietary Management. pp. 29–32 (2018)

[4] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1–32 (1997)

[5] Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF Models for Sequence Tagging (2015), `http://arxiv.org/abs/1508.01991`

[6] Kingma, D.P., Ba, J.L.: Adam: a Method for Stochastic Optimization. In: Proceedings of International Conference on Learning Representations (2015)

[7] Lafferty, J., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of International Conference on Machine Learning. pp. 282–289 (2001)

[8] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural Architectures for Named Entity Recognition. In: Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 260–270 (2016)

[9] Ma, X., Hovy, E.: End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In: Proceedings of Annual Meeting of the Association for Computational Linguistics (2016)

[10] Mai, K., Pham, T.H., Nguyen, M.T., Duc, N.T., Bollegala, D., Sasano, R., Sekine, S.: An Empirical Study on Fine-Grained Named Entity Recognition. In: Proceedings of International Conference on Computational Linguistics. pp. 711–722 (2018)

[11] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space. In: Proceedings of International Conference on Learning Representations (2013)

[12] Mori, S., Maeta, H., Yamakata, Y., Sasada, T.: Flow Graph Corpus from Recipe Texts. In: Proceedings of International Conference on Language Resources and Evaluation. pp. 2370–2377 (2014)

[13] Nanba, H., Takezawa, T., Doi, Y., Sumiya, K., Tsujita, M.: Construction of a cooking ontology from cooking recipes and patents. In: Proceedings of ACM International Joint Conference on Pervasive and Ubiquitous Computing Adjunct Publication. pp. 507–516 (2014)

[14] Neubig, G., Nakata, Y., Mori, S.: Pointwise Prediction for Robust , Adaptable Japanese Morphological Analysis. In: Proceedings of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. pp. 529–533 (2011)

[15] Peters, M.E., Ammar, W., Bhagavatula, C., Power, R.: Semi-supervised sequence tagging with bidirectional language models. In: Proceedings of Annual Meeting of the Association for Computational Linguistics. pp. 1756–1765 (2017)

[16] Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 2227–2237 (2018)

[17] Rehurek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of LREC Workshop on New Challenges for NLP Frameworks. pp. 45–50 (2010)

[18] Sasada, T., Mori, S., Kawahara, T., Yamakata, Y.: Named entity recognizer trainable from partially annotated data. In: Proceedings of International Conference of the Pacific Association for Computational Linguistics. vol. 593, pp. 148–160 (2015)

[19] Sato, M., Shindo, H., Yamada, I., Matsumoto, Y.: Segment-Level Neural Conditional Random Fields for Named Entity Recognition. In: Proceedings of International Joint Conference on Natural Language Proceedings ofssing. pp. 97–102. No. 1 (2017)

[20] Yadav, V., Bethard, S.: A Survey on Recent Advances in Named Entity Recognition from Deep Learning models. In: Proceedings of International Conference on Computational Linguistics. pp. 2145–2158 (2018)

[21] Yamagami, K., Kiyomaru, H., Kurohashi, S.: Knowledge-based Dialog Approach for Exploring User's Intention. In: Procceedings of FAIM/ISCA Workshop on Artificial Intelligence for Multimodal Human Robot Interaction. pp. 53–56 (2018)