INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Slot Filling with Weighted Multi-Encoders for Out-of-Domain Values

Yuka Kobayashi[1], *Takami Yoshida*[1], *Kenji Iwata*[1] *and Hiroshi Fujimura*[1]

[1]Corporate Research & Development Center, Toshiba Corporation, Japan

{yuka3.kobayashi, takami.yoshida, kenji4.iwata, hiroshi4.fujimura}@toshiba.co.jp

Abstract

This paper proposes a new method for slot filling of out-of-domain (OOD) slot values, which are not included in the training data, in spoken dialogue systems. Word embeddings have been proposed to estimate the OOD slot values included in the word embedding model from keyword information. At the same time, context information is an important clue for estimation because the values in a given slot tend to appear in similar contexts. The proper use of either or both keyword and context information depends on the sentence. Conventional methods input a whole sentence into an encoder and extract important clues by the attention mechanism. However, it is difficult to properly distinguish context and keyword information from the encoder outputs because these two features are already mixed. Our proposed method uses two encoders, which distinctly encode contexts and keywords, respectively. The model calculates weights for the two encoders based on a user utterance and estimates a slot with weighted outputs from the two encoders. Experimental results show that the proposed method achieves a 50% relative improvement in F1 score compared with a baseline model, which detects slot values from user utterances and estimates slots at once with a single encoder.

Index Terms: spoken dialogue system, spoken language understanding, slot filling

1. Introduction

Semantic frame parsing is essential for task-oriented spoken dialogue systems. It consists of intent determination and slot filling. Slot filling is treated as a sequence labeling problem in which labels of slot values are estimated from provided input tokens by using machine learning methods. Slot filling treats named entities such as product names and restaurant names. In real world, new named entities appear continuously and it is difficult to add all of them to a training data. Thereby, it is inevitable that a user inputs utterance with OOD slot values, which are not included in the training data, even if a large corpus is prepared. Therefore, estimating the OOD slot values correctly is important.

Estimation models include Conditional Random Field (CRF) models [1] and Recurrent Neural Network (RNN) encoder-decoder models [2, 3], with or without attention mechanisms [4–10]. CRF models achieve high performance but require manually selected features [11]. RNN encoder-decoder models decode labels on the basis of previously encoded word tokens. The models use a whole sentence to improve label estimation. However, a large corpus is needed to train such models. In addition, the models may fail to understand utterances with OOD slot values. Thus, there are many studies to handle OOD slot values with RNN encoder-decoder models.

User utterances consist of keywords of slot values, and contexts. Conventional methods using keyword or context information have been proposed. First, we demonstrate some studies using keyword information. Recently, word embeddings [12] have been introduced to allow encoder-decoder models to make use of inter-word similarity in order to understand keywords corresponding to slot values. Nevertheless, word embeddings are weak for out-of-vocabulary (OOV) words, which are not included in the word embedding model. Character embeddings [13–16] and sub-word unit techniques [17–19] have been proposed in sequence labeling for OOV words. Word and character embeddings enable models to estimate OOD slot values which are close in an embedding space to keywords in the training data. Whereas, they are not applicable to OOD slot values which are not similar to keywords in the training data and named entities such as product names or restaurant names.

Next, we present some work using context information. Here, we define context information as several words in a user utterance in which keywords of a slot value are used, not several past utterances. Context information is an important clue for slot filling because the values in a given slot tend to appear in similar contexts. Delexicalization has been applied [20–23] to handle OOV or rare words in the training data. Slot values are delexicalized to symbols in the training data using a table, and the system uses slot embeddings instead of word tokens and contexts. This results in the ability to handle unknown words not included in the training data. On the other hand, it cannot handle unknown words not included in the table.

Methods extracting context information without a table have been proposed. In some studies, keywords that correspond to slot values in the ontology have been replaced with non-value words in the training data [24] or with randomly chosen words [25]. These methods convert keywords into random features to make models ignore keyword information. These methods do not convert them while evaluating and therefore do not need a table. Target feature dropout [26] and word dropout [27] have also been proposed to improve regularization for unknown words. However, values in different slots may sometimes be used in similar contexts, which makes it difficult to distinguish between slots using context information.

As mentioned above, keyword or context information alone is insufficient for estimating OOD slot values. In some cases, context information is more effective than keyword information, and vice versa. Therefore, by exploiting context and keyword information properly, the system can estimate slots in any given case.

The attention mechanism is one of the most widely used methods to solve this problem. It extracts important clues from the input sentence by calculating a weight for each word token. Existing methods input a whole sentence into an encoder but cannot distinguish context and keyword information from the encoder outputs because these two features are already mixed.

This paper proposes a neural network model with multiple encoders for slot filling of OOD slot values in spoken dialogue systems. Our proposed method uses two encoders, which distinctly encode contexts and keywords, respectively. The model

http://dx.doi.org/10.21437/Interspeech.2019-1226

I	want	to	drink	afternoon	tea
O	O	O	O	B-FOOD	I-FOOD

Figure 1: *IOB labeling*

calculates weights for the two encoders based on a user utterance and estimates a slot with weighted outputs from the two encoders. The model divides user utterances into context and keyword information by slot value detection. The context encoder, which does not depend on slot values in the training data, makes the model robust. Furthermore, existing methods choose important words from the entire of a user utterance, which makes models sparse. In contrast, our proposed methods choose from only two encoders, and are therefore, less sparse than existing methods.

In Section 2, we explain a pipelined model consisting of a value detector and a slot estimator, as well as a multi-encoder model for slot estimation. Section 3 describes the dataset used in experiments and the experimental procedures. We report the experimental results and discuss them in Section 4. Finally, Section 5 concludes this paper.

2. Proposed method

2.1. Pipelined model

The slot filling task consists of two parts: detecting slot values from utterances and estimating slots for detected slot values.

To use context and keyword information for slot estimation independently, we use a model for each task, which is a so-called pipelined model. Our proposed method detects keywords corresponding slot values in a user utterance by value detection, divides the utterance into context and keyword information, and estimates slots using them.

Zhao et al. [9] also provided a pipelined model. They input a whole sentence into a single encoder in slot estimation. Our proposed method uses context and keyword information respectively. It can be more effective than a single encoder with a whole sentence.

We provide and optimize a model for each task. We use an existing model [25] for the slot value detection task. Our proposed method with multi encoders is used as a slot estimator.

2.2. Value detector

We treat the problem as a sequence labeling task. The task is to estimate the sequence of labels $y = (y_1, y_2, y_3, ..., y_N)$ given the sequence of word inputs $x = (x_1, x_2, x_3, ..., x_N)$, as follows:

$$\hat{y} = \arg\max_{y} P(y|x). \tag{1}$$

Here, the labels are given in in-out-begin (IOB) format (Fig. 1). Each token in the utterances of the training data is labeled as B-SLOT if the token is the beginning of the keywords of a value, I-SLOT if it is inside, and as O (i.e., outside) otherwise. For value detection, we ignore the slot labels of IOB labeling. IOB labels consist of only I, O, and B labels.

The value detector uses a Long Short-Term Memory (LSTM) RNN (hereinafter, LSTM) encoder-decoder model with aligned inputs [5]. Figure 2 shows a block diagram of the LSTM encoder-decoder model. The encoder encodes a sequence of inputs x into a representation vector of the whole input sequence. The decoder decodes the output sequence from the representation vector. The encoder is a bidirectional LSTM

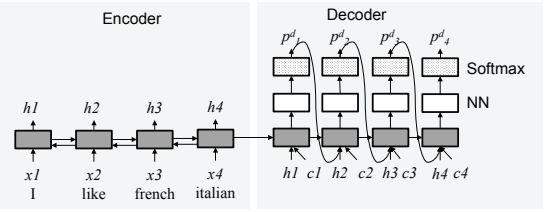

Figure 2: *Value detector*

[28] and the decoder is a unidirectional LSTM. We employ a bidirectional LSTM in the encoder because we want to use context information in both the forward and backward directions. The encoder reads the input word sequence and generates a hidden state at each time step $h = (h_1, h_2, h_3, ..., h_N)$, where h_i is a concatenation of the forward state $\overrightarrow{h_i}$ and backward state $\overleftarrow{h_i}$.

The model in Fig. 2 uses the last state of the encoder h_N to compute the initial decoder hidden state. At each decoding step i, the decoder state s_i is calculated as a function of the previous decoder state s_{i-1} and a concatenation of the previous emitted label index y_{i-1}, a context vector c_i, and the encoder hidden state h_i. A context vector c_i is computed with a self-attention mechanism.

The decoder is followed by a full-connected layer and a softmax layer. Finally, the value detector outputs $p_i^d = [p_{i,B}, p_{i,I}, p_{i,O}]$.

2.3. Weighted multi-encoder model for slot estimator

2.3.1. Encoder

Slot estimation is a task that estimates slots for detected values by the value detector. We treat slot estimation as a classification task. An input utterance is divided into a context and a keyword by the value detector. We use two types of encoder, which encodes contexts and keywords, respectively (Fig. 3).

The slot estimator uses the word embeddings of the words in utterances x_i and the value detector outputs p_i^d.

If there are multiple slot values in an utterance, the model estimates a slot for each value.

- Context encoder
  The system replaces word embeddings x_i which are part of the slot values detected by the value detector with zero vectors in the same dimension as x_i. This enables the model to estimate slots using only context information.

$$h_i^C = \text{BDLSTM}(p_i^d \oplus \hat{x_i}, h_{i-1}^C), \tag{2}$$

$$\hat{x_i} = \begin{cases} x_i & (\arg\max(p_i^d) = \text{O}), \\ 0_d & (\text{otherwise}), \end{cases} \tag{3}$$

  $\text{BDLSTM}(\cdot, \cdot)$ is a bidirectional LSTM and $\oplus$ is the vector concatenation.

- Keyword encoder
  The system inputs only word embeddings of a portion of the slot values detected by the value detector.

$$h_i^K = \text{BDLSTM}(p_i^d \oplus x_i, h_{i-1}^K), \tag{4}$$

$$i \in \arg\max(p_i^d) = \text{B or I}. \tag{5}$$

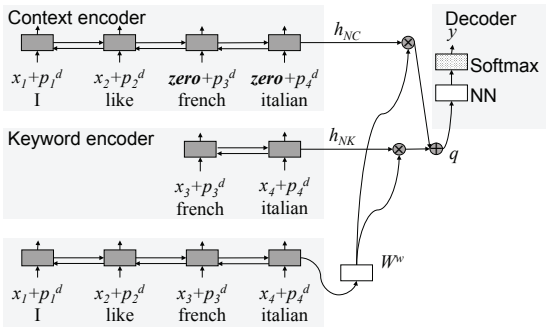

Figure 3: *Multi-encoder model*

2.3.2. Decoder

A model consists of two encoders. We define this model as a multi-encoder model. The model outputs probabilities for each slot. The final state of each encoder is weighted. The weights are calculated using an utterance as follows:

$$h_i^u = \text{BDLSTM}(p_i^d \oplus x_i, h_{i-1}^u), \tag{6}$$

$$\mathbf{W^w} = \mathbf{W^f} h_N^u + b^f, \tag{7}$$

where $\mathbf{W^f}$ and $\mathbf{W^w}$ are the weight parameters, and b^f is the bias. The model is able to calculate appropriate weights for each sentence. The weighted outputs of the two encoders are input to the decoder.

$$h_N = \left[h_N^C, h_N^K\right], \tag{8}$$

$$q = \mathbf{W^w} h_N, \tag{9}$$

$$p^e = \text{softmax}(\mathbf{W^g} q + b^g), \tag{10}$$

$$y^e = \arg\max_j p_j^e, \tag{11}$$

where $\mathbf{W^g}$ is the weight matrix, b^g is the bias, q is an N_{slot}-dimensional vector (N_{slot} is the number of slot labels), and p_j^e is the probability of j-th slot.

3. Experimental setup

3.1. Dataset

We used the Dialogue State Tracking Challenge 2 (DSTC2) dataset [29] and the DSTC3 dataset [30] (Table 1) in experiments.

The DSTC2 traindev and DSTC2 test datasets were used for training and in-domain (ID) evaluation, respectively. We also used the DSTC3 test dataset for OOD evaluation.

Our purpose is to detect OOD values of ID slots. The ontology is different between the DSTC2 and DSTC3 datasets. Notably, the DSTC3 ontology includes slots and values not included in DSTC2. We excluded utterances with OOD slots (*type*, *near*, *hastv* and *chidrenallowed*) and with ID values in the DSTC3 dataset from evaluation.

The DSTC2 and DSTC3 datasets include both transcriptions and results of speech recognition for each user utterance. We used the transcriptions to remove effects of speech recognition errors. We annotated the utterances in the datasets with IOB labels by hand and removed some utterances which were difficult to annotate.

Table 1: *Dataset size*

Dataset	Number of utterances	Number of values
DSTC2-traindev	2,661	212
DSTC2-test	5,063	212
DSTC3-test	991	210

3.2. Evaluation method and parameters

We evaluated performance of value detection and slot filling. For the value detector, we ignored slot labels of IOB labeling in calculating the F1 score. For slot filling, the slot estimator output slot labels for detected values, and we converted these outputs to IOB labels to calculate the F1 score. The performance of slot filling originates from value detection and slot estimation.

We used the GloVe model [31] for word embeddings.

Table 2 shows our parameter values. Dropout was applied to non-recurrent connections during training of the model. We used word dropout and target feature dropout and tested rates of 0.3, 0.5, and 0.8 for each. We also used a method with random vectors [25] to prevent overfitting. This method replaces the word embedding vectors of the keywords corresponding to slot values with randomly chosen other word vectors from the word embedding space when training the model. The vector randomization enables the model not to utilize the keywords of slot values but context information. The model was trained on both the original and augmented data. We applied this method to the training and evaluation data while training the model and chose a good performance model for OOD slot values.

Table 2: *Experimental parameters*

Parameter	**Value**
word vector size	300
batch size	32
Bi LSTM size (encoder)	32
LSTM size (decoder)	33
hidden layer (encoder, decoder)	1
gradient clip	1.0
dropout	0.3, 0.5
learning rate	0.001
weight decay	0
optimization method	Adam
word dropout	0.3, 0.5, 0.8
target feature dropout	0.3, 0.5, 0.8

4. Results

4.1. Evaluation of the pipelined model

First, we evaluated a pipelined model consisting of a value detector and a slot estimator.

A baseline model solves value detection and slot estimation simultaneously. It estimates IOB labels with slot labels using the model described in Section 2.2.

To evaluate an effectiveness of the pipelined model, we defined a simple slot estimator. We defined a sentence encoder as follows:

$$h_i^S = \text{BDLSTM}(p_i^d \oplus x_i, h_{i-1}^S). \tag{12}$$

An input is a whole sentence. The slot estimator of the pipelined model used only the sentence encoder and fixed the weight for the encoder to 1.

The value detector was trained on the parameters described in Section 3.2 and the model with the highest performance was chosen. The slot estimator was trained using the best value detector.

Table 3: *Comparison of baseline and pipelined models with F1 score. (VD : value detection, SF : slot filling, Pip : a pipelined model, WD : word dropout, TFD : target feature dropout, and RV : random vector.)*

Model	OOD		ID	
	VD	SF	VD	SF
Baseline	65.9	50.8	98.87	98.75
+ attention [5]	65.49	54.05	98.99	98.87
+ WD (0.3) [27]	77.91	66.12	98.99	98.45
+ TFD (0.8) [26]	84.95	76.35	98.53	97.95
+ RV [25]	88.93	74.29	99.01	98.74
Pipelined model	90.38	74.19	99.17	85.6
Pip + WD (0.3) [27]	90.38	78.41	99.17	99.11
Pip + TFD (0.8) [26]	90.38	78.25	99.17	82.66
Pip + RV [25]	90.38	**81.24**	99.17	98.86

Table 3 shows the best score for each method. The figures in parentheses are the rates that achieved the best performance for word dropout and target feature dropout.

We can see that the pipelined model performed better than the baseline model regardless of the method used. The value detector was optimized focusing on value detection and outperformed the baseline model. It greatly improved the performance of slot filling.

4.2. Evaluation of slot estimators

Next, we compared the networks of slot estimators. We used the same value detector as in Section 4.1 for all tests, which resulted that the performance of value detection was the same and the difference of the performance of slot filling originated only from slot estimation.

Combinations of three encoders, which are the sentence, context and keyword encoder, were tested with the parameters described in Section 3.2. When the model used only one encoder, the weight was fixed to 1.0. We call this model a single-encoder model.

The results in Table 3 reported that the method with random vectors was the most effective. As such, we performed tests with random vectors. We did not apply the method with random vectors for the single-encoder model using the context encoder or the keyword encoder, because the context encoder does not use keywords of slot values, whereas the keyword encoder uses only keywords of slot values. We applied word dropout and target feature dropout for these two models and chose the best results.

The F1 score of slot filling in each combination of encoders is shown in Table 4. The figures in parentheses are the rates that achieved the best performance for word dropout and target feature dropout.

The single-encoder model using the context or the keyword encoder performed worse than the sentence encoder. The results show that only context or keyword information is insufficient.

Table 4: *Slot filling performance comparison among slot estimators. 'sent' means sentence encoder.*

Model	OOD	ID
sent	81.24	98.86
context (WD 0.5)	54.27	80.76
keyword (WD 0.0, TFD 0.0)	57.46	98.31
sent + context + keyword	80.11	98.47
sent + context	77.98	98.16
context + keyword	**88.28**	98.43
sent + keyword	77.65	98.47

It can also be seen that the context encoder performed poorly with ID data, on the other hand, the other models achieved high performance. The reason of this may be that the context encoder does not use keyword information which is effective to estimate known slot values.

The best of the multi-encoder models achieved the higher performance than the best of the single-encoder models. By using context and keyword information appropriately, the multi-encoder model outperformed the single-encoder model. The sentence encoder uses context and keyword information, whereas these two features are mixed and it cannot use them properly.

Among the multi-encoder models, the model with both context and keyword encoders outperformed the others. This model uses these two features most effectively because it does not use the sentence encoder, which mixes context and keyword information.

5. Conclusion

This paper has proposed a neural network model with multiple encoders for slot filling of OOD slot values from user utterances in spoken dialogue systems.

Word embeddings were proposed to estimate the OOD slot values included in the word embedding model from keyword information. At the same time, context information is an important clue for estimation because the values in a given slot tend to appear in similar contexts. The proper use of either or both keyword and context information depends on the sentence.

Our proposed method uses two encoders, which distinctly encode contexts and keywords, respectively. The model calculates weights for the two encoders based on a user utterance and estimates a slot with weighted outputs from the two encoders. The model divides user utterances into context and keyword information by slot value detection. The context encoder, which does not depend on slot values in the training data, makes the model robust.

The results of our experiments show that the proposed method achieves a 50% relative improvement in F1 score compared with a baseline model.

We tested our models using transcriptions of datasets. However, the system needs to work correctly on the outputs of speech recognition. We consider the proposed method to be robust against speech recognition errors because it uses whole-word sequences.

6. Acknowledgements

We are grateful to Dr. Masami Akamine at Tohoku University for fruitful suggestions and discussions.

7. References

[1] J. Lafferty, M. Andrew, and F. C. N. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01, 2001, pp. 282–289.

[2] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *Advances in neural information processing systems 27*, 2014, pp. 3103–3112.

[3] K. Cho, B. Merriënboer, G. Çaglar, D. Bahdanau, F. Bougares, H. Holger, and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. ACL, Oct. 2014, pp. 1724–1734.

[4] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016, pp. 1054–1063.

[5] B. Liu and I. Lane, "Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling," in *INTERSPEECH*, Sep 2016, pp. 685–689.

[6] E. Simonnet, P. Deleglise, N. Camelin, and Y. Esteve, "Exploring the use of attention-based recurrent neural networks for spoken language understanding," in *Machine Learning for SLU and Interaction NIPS 2015 Workshop (SLUNIPS 2015)*, Dec 2015.

[7] Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning, "Position-aware attention and supervised data improve slot filling," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2017, pp. 35–45.

[8] V. Zhong, C. Xiong, and R. Socher, "Global-locally self-attentive encoder for dialogue state tracking," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2018, pp. 1458–1467.

[9] L. Zhao and Z. Feng, "Improving slot filling in spoken language understanding with joint pointer and attention," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, July 2018, pp. 426–431.

[10] Y.-N. Chen, D. Z. Hakkani-Tür, G. Tür, J. Gao, and L. Deng, "End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding," in *INTERSPEECH*, 2016.

[11] X. Ma and E. H. Hovy, "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016, pp. 1064–1074.

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," in *Proceedings of the International Conference on Machine Learning (ICLR)*, 2013, pp. 1310–1318.

[13] A. Jaech, L. Heck, and M. Ostendorf, "Domain Adaptation of Recurrent Neural Networks for Natural Language Understanding," in *INTERSPEECH*, 2016, pp. 690–694.

[14] J. P. C. Chiu and E. Nichols, "Named Entity Recognition with Bidirectional LSTM-CNNs," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 357–370, 2016.

[15] A. Bharadwaj, D. R. Mortensen, C. Dyer, and J. G. Carbonell, "Phonologically Aware Neural Model for Named Entity Recognition in Low Resource Transfer Settings," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[16] A. Makazhanov and Z. Yessenbayev, "Character-based feature extraction with LSTM networks for POS-tagging task," in *2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT)*, Oct 2016, pp. 1–5.

[17] E. Ribeiro, R. Ribeiro, and D. M. de Matos, "A study on dialog act recognition using character-level tokenization," in *Artificial Intelligence: Methodology, Systems, and Applications*, G. Agre, J. van Genabith, and T. Declerck, Eds. Cham: Springer International Publishing, 2018, pp. 93–103.

[18] S. Park, J. Byun, S. Baek, Y. Cho, and A. Oh, "Subword-level word vector representations for Korean," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 2429–2438. [Online]. Available: https://www.aclweb.org/anthology/P18-1226

[19] E. Egorova and L. Burget, "Out-of-vocabulary word recovery using fst-based subword unit clustering in a hybrid asr system," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 5919–5923.

[20] G. Wang, C. Li, W. Wang, Y. Zhang, D. Shen, X. Zhang, R. Henao, and L. Carin, "Joint embedding of words and labels for text classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2018, pp. 2321–2331.

[21] J. Rajendran, J. Ganhotra, X. Guo, M. Yu, and S. Singh, "Named entities troubling your neural methods? build ne-table: A neural approach for handling named entities," *CoRR*, vol. abs/1804.09540, 2018.

[22] T. Luong, I. Sutskever, Q. Le, O. Vinyals, and W. Zaremba, "Addressing the rare word problem in neural machine translation," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, 2015, pp. 11–19.

[23] Y. Shin, K. M. Yoo, and S. goo Lee, "Slot filling with delexicalized sentence generation," in *Interspeech*, 2018.

[24] M. Hou, X. Wang, C. Yuan, G. Yang, S. Hu, and Y. Shi, "Attention Based Joint Model with Negative Sampling for New Slot Values Recognition," in *Proceedings of the Ninth International Workshop on Spoken Dialogue Systems Technology (IWSDS 2018)*, 2018.

[25] Y. Kobayashi, T. Yoshida, K. Iwata, H. Fujimura, and M. Akamine, "Out-of-domain slot value detection for spoken dialogue systems with context information," in *2018 IEEE Spoken Language Technology Workshop (SLT)*, Dec 2018, pp. 854–861.

[26] Q. H. Puyang Xu, " An End-to-end Approach for Handling Unknown Slot Values in Dialogue State Tracking," in *ACL*, 2018.

[27] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III, "Deep unordered composition rivals syntactic methods for text classification," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, 2015, pp. 1681–1691.

[28] M. Schuster and K. K. Paliwal, "Bidirectional Recurrent Neural Networks," *Trans. Sig. Proc.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[29] M. Henderson, B. Thomson, and J. Williams, "The Second Dialog State Tracking Challenge," in *Proceedings of SIGDIAL*. ACL, June 2014.

[30] ——, "The Third Dialog State Tracking Challenge," in *Proceedings IEEE Spoken Language Technology Workshop (SLT)*. IEEE, December 2014.

[31] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. ACL, October 2014, pp. 1532–1543.