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

Iterative Delexicalization for Improved Spoken Language Understanding

Avik Ray[1], *Yilin Shen*[2], *Hongxia Jin*[3]

Samsung Research America, Mountain View, CA, USA

[1]avik.r@samsung.com, [2]yilin.shen@samsung.com, [3]hongxia.jin@samsung.com

Abstract

Recurrent neural network (RNN) based joint intent classification and slot tagging models have achieved tremendous success in recent years for building spoken language understanding and dialog systems. However, these models suffer from poor performance for slots which often encounter large semantic variability in slot values after deployment (e.g. message texts, partial movie/artist names). While greedy delexicalization of slots in the input utterance via substring matching can partly improve performance, it often produces incorrect input. Moreover, such techniques cannot delexicalize slots with out-of-vocabulary slot values not seen at training. In this paper, we propose a novel iterative delexicalization algorithm, which can accurately delexicalize the input, even with out-of-vocabulary slot values. Based on model confidence of the current delexicalized input, our algorithm improves delexicalization in every iteration to converge to the best input having the highest confidence. We show on benchmark and in-house datasets that our algorithm can greatly improve parsing performance for RNN based models, especially for out-of-distribution slot values.

Index Terms: spoken language understanding, delexicalization, slot tagging

1. Introduction

Spoken language understanding (SLU) models play a key role in modern voice controlled personal agents and AI chatbots. Given a user's utterance, a SLU model identifies the appropriate intent and slots from the utterance, which are subsequently used to fulfill the user command or continue dialog with the user. An intent classifier assigns the most likely intent label to the user utterance, while a slot tagger assigns a slot label to every word in the utterance. The slot values or informative words/phrases in the utterance are then extracted for executing the desired action. Correct identification of intent and slot values by SLU models play a vital role in the success of downstream tasks [1].

Recently, joint intent classification and slot labeling SLU models based on recurrent neural networks has been shown to achieve state-of-the-art performance on benchmark datasets [2, 3, 4, 5, 6]. However, these models often suffer from poor slot labeling accuracy when an utterance contain slots with large semantic variability, dissimilar to those encountered during training e.g. message text, partial show/artist names. In this paper, we refer such slots as *out-of-distribution (OOD) slots*. The main characteristics of such OOD slots are that they do not have fixed vocabulary, and moreover can take values with length and word distribution very dissimilar to their training vocabulary. Usually they also contain many out-of-vocab words. Standard SLU models are trained on expensive labeled training datasets where OOD slots with large semantic variability are never well represented. Presence of out-of-vocabulary words and phrases have been shown to further degrade performance of RNN based SLU models [7]. As an example, in a Facebook dataset, a state-of-the-art bidirectional RNN based SLU model [3] is particularly poor in identifying slots containing message text compared to other slot types as seen in Table 1.

A standard approach to tackle the high semantic variability of the input utterance is to preprocess the input, thereby replacing partly or wholly slot words/phrases with special tokens from training vocabulary, a process called delexicalization. For example, in Figure 1 the word *"Alice"* and phrase *"happy birthday"* in the original utterance is replaced by special tokens $\langle contact \rangle$ and $\langle message \rangle$ respectively. The delexicalized utterance is then used by the model to infer the intent and slot labels of the original utterance. Delexicalization, mainly based on greedy longest string matching, has been explored before in SLU [8, 9, 10]. However, they do not always work well since they neglect any utterance context, and often resulting in erorneous input. Moreover, OOD slots with many out-of-vocab words (e.g. message text) are almost imposible to match and delexicalize. Using delexicalization to improve performance of other NLP systems has also been studied in the context of natural language generation [11], dependency parsing [12, 13], semantic parsing [14], and representation learning [10].

Table 1: *Slot labeling F1 score comparison for different slot categories in Facebook dataset using baseline Attention BiRNN parser [3].*

Slot category	Total fraction	Baseline F1 %	Our model F1 %
Message text	60.4%	85.16	**91.25**
Other non-message	39.6%	96.8	95.4
Overall	100%	89.82	**92.91**

In this paper, we develop a novel algorithm to iteratively delexicalize the input utterance guided by model's confidence on the current input as well as utterance context. Our approach allows effective delexicalization of OOD slot values, even when they have large semantic variability. Our delexicalization based hybrid parsing model is also modular and can be applied with any RNN based parser to improve its parsing performance. On both benchmark and in-house datasets, and using different RNN based parsers, our algorithm is demonstrated to significantly improve performance in both slot labeling and intent classification, achieving the state-of-the-art.

2. Problem and background

In this section we formally define our spoken language understanding problem and the delexicalization approach to tackle it. Let a user provide an input utterance $\mathbf{x} = (x_1, \ldots, x_n)$, with words $x_i \in \mathcal{V}_T$, the vocabulary. The task of an intent classifier is to determine the intent $I(\mathbf{x})$ of utterance $\mathbf{x}$, while a slot taggers labels each word x_i with a slot label y_i, thereby generating a sequence of slot labels $\mathbf{y} = (y_1, \ldots, y_n)$. A joint intent classifier and slot tagger model or parser $\mathcal{P}$ is jointly trained on the two tasks using a labeled training dataset T. In delexicalization, we replace words/phrases in the input utterance $\mathbf{x}$ with some special tokens to generate a delexicalized in-

http://dx.doi.org/10.21437/Interspeech.2019-2955

put $\mathbf{x}' = (x'_1, \ldots, x'_m)$, $m \leq n$. A delexicalization algorithm performs this mapping $f : \mathbf{x} \rightarrow \mathbf{x}'$, possibly taking into account the training vocabulary $\mathcal{V}_T$ and current parse results $\{I(\mathbf{x}), \mathbf{y}\}$. Now, the parser $\mathcal{P}$ is used to infer the slot labels $\mathbf{y}'$ and intent $I(\mathbf{x}')$ of the modified input $\mathbf{x}'$. These results are subsequently used to infer (or modify) the parse of the original utterance $\mathbf{x}$. Our iterative delexicalization algorithm performs this iteratively to converge at the best parsing result for the user utterance $\mathbf{x}$.

Recurrent neural network parsers: RNN based parsers have been successfully applied for intent classification and slot labeling tasks in recent years [15, 2, 3, 4, 5]. These models use a recurrent network (e.g. BiLSTM/GRU) to encode the input utterance $\mathbf{x}$ to a sequence of hidden state representations $\{\mathbf{h}_t\}_{t=1}^n$. These hidden states, along with possible self-attention and/or intent attention are combined to generate the output slot label distribution $\{\mathbf{z}_t\}_{t=1}^n$. Finally, the output slot labels are inferred as $y_t = \arg\max_{l_j} P(y_t = l_j) = \arg\max_j \mathbf{z}_t(j)$, where l_j denotes the slot label corresponding to the $j-$th index.

3. Our model

In this section, we present our main iterative delexicalization algorithm. We refer as *slot words* the words which are informative of command parameters and has a ground-truth slot labels other than "O". *Context words* refer the words with slot label "O" which are either non-informative or conveys the utterance intent. Our hybrid SLU model can be viewed as a two step parser where the first step delexicalizes the input utterance $\mathbf{x}$ to $\mathbf{x}'$, and the second step uses a RNN based parser to infer the slot and intent labels $\{I(\mathbf{x}), \mathbf{y}\}$ using this delexicalized input $\mathbf{x}'$. Our iterative inference algorithm executes these steps repeatedly until it converges to the best parse result.

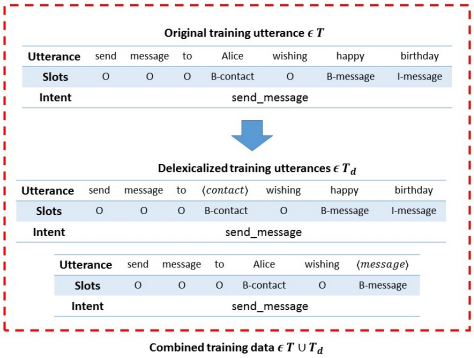

Figure 1: *Illustration of our combined dataset generation for training the iterative delexicalization parser $\mathcal{P}$.*

Idea: The main idea behind our delexicalization based hybrid approach to parsing is based on the following observation. When an utterance is encountered having out-of-distribution slot words, the parser exhibits a low slot tagging confidence over these unknown words, often resulting in incorrect slot labels. However, if such slot words/phrases with low parsing confidence are replaced by suitable delexicalized tokens, which were also present in the training data, then typically there are two outcomes; **(a)** parsing confidence on the new delexicalized utterance improves if delexicalization was performed correctly; or **(b)** it degrades if delexicalization we performed incorrectly. Our iterative delexicalization algorithm exploits this phenomenon to generate a new set of candidate delexicalized utterances from the current input guided by parser's confidence in each iteration. Such iterations are continued as long as there exists at least one new candidate delexicalized input which improves parsing confidence over the previous set of candidate inputs. Finally, when there are no further confidence improvement the one having the best parsing confidence is used to generate the final parsing result.

3.1. Iterative delexicalization algorithm

We now describe our iterative delexicalization algorithm in details. The overall algorithm can be divided into two steps; first a model training step, and an iterative inference step. First we describe the model training step.

Model training: To enable an RNN based parser to correctly parse a delexicalized input utterance it should correctly interpret the special delexicalized tokens from the utterance context. To enable this, the training set T is augmented with a delexicalized training set T_d where the ground truth slot words/phrases have been substituted by some special tokens, usually an unique token per slot type[1]. However, to prevent the model to overfit to correlations between such special tokens, we substitute tokens randomly with probability $p_s < 1$ (e.g. $p_s = 0.75$). Then, the RNN parser $\mathcal{P}$ is trained on the combined training set $T' = T \cup T_d$. In Figure 1 we illustrate this training utterance generation in Facebook domain.

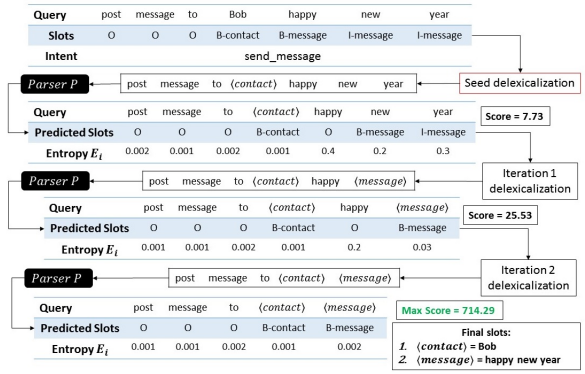

Figure 2: *Illustration of our iterative delexicalization inference Algorithm 1, using a threshold $\tau = 0.05$. Note that iteration 1 delexicalization is based on* **proper slot sequence***, and iteration 2 delexicalization is based on* **special token expansion**.

Inference via iterative delexicalization: When a user provides an input utterance $\mathbf{x}$, the iterative delexicalization Algorithm 1 is used to infer the intent $I(\mathbf{x})$ and slot labels $\mathbf{y}$. At first the function $SeedDelexicalization(.)$ is used to generate an initial set S of delexicalized utterances, each of which is parsed by $\mathcal{P}$ and given a parsing confidence using score function $Score(.)$. Next, in every iteration, each delexicalized input $\mathbf{x} \in S$ is used to generate more candidate delexicalized utterances using the function $Delexicalization(.)$. These new candidates are then parsed and their parsing scores are computed. We repeat this process till the maximum parsing confidence score no longer improves, or if no new candidates can be generated. Our algorithm also use a confidence threshold τ and

[1] An exception is slots which have a semantic hierarchy e.g. city name which can be both a source and destination slot. In such cases both slots can be replaced by the same token.

a maximum size parameter K to limit the amount of computation per iteration. We illustrate our iterative delexicalization inference algorithm using an example from Facebook domain in Figure 2. Note that, our iterative algorithm is guaranteed to converge since each delexicalization step can only reduce the size of the input. Next we further elaborate the key subroutines and our score function.

Algorithm 1 Iterative delexicalization (inference)

Input: Input utterance $\mathbf{x}$, parser $\mathcal{P}$, confidence threshold τ, size threshold K
Output: Intent $I(\mathbf{x})$, slot labels $\mathbf{y}$

$S \leftarrow SeedDelexicalization(\mathbf{x})$
maxConfidence $\leftarrow 0$
Use $\mathcal{P}$ to parse each $\mathbf{x}' \in S$. Compute confidence score $Score(\mathbf{x}')$ of each such parsing result
currentConfidence $\leftarrow \max_{\mathbf{x}' \in S} Score(\mathbf{x}')$
while currentConfidence $>$ maxConfidence and $|S| > 0$ **do**
    $S' \leftarrow S$, and $S \leftarrow \emptyset$
    If $|S'| > K$, pick top K delexicalized utterance $\mathbf{x}'$ with highest confidence score $Score(\mathbf{x}')$ in previous iteration
    for $\mathbf{x}' \in S'$ **do**
        $S \leftarrow S \cup Delexicalization(\mathbf{x}', \mathcal{P}, \tau)$
    end for
    Use $\mathcal{P}$ to parse each $\mathbf{x}' \in S$. Compute confidence score $Score(\mathbf{x}')$ of each such parsing result
    currentConfidence $\leftarrow \max_{\mathbf{x}' \in S} Score(\mathbf{x}')$
end while
$\mathbf{x}_{best} \leftarrow \arg\max_{\mathbf{x}' \in S'} Score(\mathbf{x}')$
Infer intent $I(\mathbf{x})$, slot labels $\mathbf{y}$ using the parsing results $I(\mathbf{x}_{best}), \mathbf{y}_{best}$ of $\mathbf{x}_{best}$
Output intent $I(\mathbf{x})$, slot labels $\mathbf{y}$

Seed delexicalization: To generate the starting set of delexicalized input utterances we use a $SeedDelexicalization(.)$ subroutine. This function uses longest first string matching to identify possible word/phrase that can be delexicalized. However, unlike replacing all matched substrings greedily similar to [8, 9], our algorithm generates two candidates per match, one with and one without delexicalizing the matched substring. This ensures that if a substring is matched incorrectly (i.e. it also includes context words) at least one of the candidate is still correct (the one not delexicalized). We also do not match words/phrases which appeared as context words/phrases in the training data, and slot words/phrases which may be part of two different slot tags (e.g. an artist who is both actor and director slot). However, this step is not expected to delexicalize OOD slots such as message text.

Model based delexicalization: In the main iteration loop we delexicalize using a subroutine $Delexicalization(.)$. Unlike the previous $SeedDelexicalization$ function which uses string matching, the $Delexicalization$ function performs delexicalization based on parsing output of $\mathbf{x}'$ and some domain independent rules as described next. Recall that the RNN based parser $\mathcal{P}$ generates the slot label distribution $\mathbf{z}_t$ for every word/token x_t in the input. We use the entropy function $E_t = Entropy(\mathbf{z}_t)$ to first capture the parser's slot level confidence for the t-th word. The main motivation behind the rules in $Delexicalization$ function is to refine an incorrect delexicalized candidate input towards a better candidate.

1. *Proper slot sequence:* Suppose slot words/phrases have been partly or wholly identified by the model using a sequence of tags (B-Y, I-Y, ..., I-Y) for a slot tag Y, and which already do not contain any special tokens. Then these are replaced by the special token corresponding to slot tag Y, to generate the delexicalized candidate for next iteration. An example application of this rule is demonstrated in Figure 2.
2. *Improper slot sequence:* Often for OOD slots the model may improperly identify the slot words and assign a sequence of labels (I-Y, ..., I-Y) without any beginning token B-Y. Then, such sequences are also replaced by the special token corresponding to slot tag Y, to generate the next delexicalized candidate.
3. *Special token expansion:* For an out-of-distribution slot value of large size (e.g. a long message text), the parser may correctly identify only subset of the slot words. Applying the previous rules only delexicalizes part of this slot value. Hence such partial delexicalization needs to be expanded to include the remaining slot words. This is performed as follows. Starting with a special token $\langle Y \rangle$ at position t, any contiguous sequence of words $(x_s, \ldots, x_{t-1})$ and $(x_{t+1}, \ldots, x_r)$ before/after token $\langle Y \rangle$, which satisfies the condition $E_i > \tau$, for $i \in [s, t-1] \cup [t+1, r]$ and $E_{s-1} \leq \tau$, $E_{r+1} \leq \tau$, are also delexicalized by the same token $\langle Y \rangle$ to generate a new candidate.

For each candidate in current iteration, we apply all three rules to generate candidates for next iteration. We also ignore duplicate candidates.

Confidence score: To evaluate the confidence of $\mathcal{P}$ on the current parsing output for an input $\mathbf{x}'$, we use a score function $Score(\mathbf{x}')$ in Algorithm 1. We implement this score function as the inverse average entropy of slot label distribution in the parser output. More precisely we define:

$$Score(\mathbf{x}') = n / \sum_{t=1}^{n} E_t = n / \sum_{t=1}^{n} Entropy(\mathbf{z}_t) \quad (1)$$

We note that, the iterative delexicalization step is performed mainly to improve the delexicalization of OOD slots and may not significantly improve slot labeling performance of other slots. Hence in our experiments we perform the seed delexicalization step for every slot, but restrict the iterative delexicalization step only to specific OOD slots. One may be concerned that the iterative inference of our algorithm can be slow. However, in each iteration the delexicalized candidates can be parsed independently in parallel using batch inference. We also observe our algorithm to converge fast within 2 or 3 iterations in benchmark and in-house datasets, even with long message text slots. The average inference time observed was around 0.08 seconds, fast enough for any practical systems.

Discussion: An alternative approach to improve the performance of RNN parsers is to add an output CRF layer to optimize over label sequence via dynamic programming [16]. Our algorithm can also be viewed as performing optimization over the space of delexicalized inputs using dynamic programming. In experiments (Section 4) we show that our model even outperform baseline RNN parsers with output CRF layer. Beam search can also generate multiple slot label sequences. However, using our score metric in equation 1, all such beam search candidates would receive the same score. Since our delexicalization approach generates candidates with different input lengths and tokens, they can be sufficiently differentiated using our score function. Our proposed system also has the added advantage that during training time, for a slot with very large vocabulary (e.g. movie/show titles), it is sufficient to just train with

few delexicalized utterances instead of using examples from the entire vocabulary, thereby reducing training time considerably. During inference the seed delexicalization step ensures that such instances will be delexicalized and correctly labeled by the model. Many slot values (e.g. artist, song titles) also experience a distribution shift over time, and may require sophisticated models and retraining to adapt to such changes [17]. Our model can easily support such domain adaptation/continuous learning without any model retraining by just updating training vocabulary with contemporary slot values.

4. Experiments

In this section we present our main experimental results.

We evaluate our algorithm on two different datasets. First, we use the benchmark SNIPS dataset [18], and second an in-house Facebook dataset. The SNIPS dataset has total 13,784 training and 700 test utterances. Our Facebook dataset contains a total of 1,191 utterances. SNIPS dataset contain OOD slots such as $object\_name$, $movie\_name$; where as Facebook dataset contain OOD slots such as $message$. Due to the smaller size, in Facebook domain we perform a 5 fold validation using five different splits and averaging the result across split. Our datasets are summarized in Table 2. Facebook dataset has a larger out-of-vocab percentage due to presence of OOD message text slot. We do not consider the popular ATIS dataset since it lacks OOD slots, hence not suited for our evaluation. As baseline, we use two state-of-the-art RNN based parsers which perform joint intent classification and slot labeling. First, the attention based BiRNN parser by Liu and Lane [3], second is the recent Slot Gated parser by Goo et al. [5]. We combine our iterative delexicalization algorithm using both the baseline parsers as $\mathcal{P}$, and compare their performance on both the datasets. We also implement few advanced baselines. First we augment an output CRF layer to attention BiRNN parser to test the effect of output sequence level optimization for OOD slots. We also show the performance of our models when delexicalization is only performed using greedy longest substring matching [8, 9]. Our evaluation metric is both intent classification accuracy, and slot labeling F1 score.

For baseline models we use code made available by the authors of [3, 5]. We train the baseline models using their default parameters. To integrate the baseline models as base parser $\mathcal{P}$ with our iterative delexicalization algorithm, we also train them on the combined training set T' using the same default parameters. We use a slot confidence threshold $\tau = 1 \times 10^{-5}$.

Table 2: *Statistics of datasets used in our experiments.*

Dataset	Total size	#Splits	#Intents	#Slots	#Vocab	#OOV
SNIPS	14,484	1	7	52	11,604	2.9 %
Facebook	1,191	5	9	4	1,158	4.7 %

Results: First, we compare the performance of all models on the benchmark SNIPS dataset. Table 3 presents the numerical results. We observe that for both the RNN based parsers our model using iterative delexicalization achieves both highest slot tagging F1 score and intent accuracy. While adding a CRF layer to baseline RNN parser improves slot labeling performance, it still perform worse than our algorithm. Moreover, our algorithm also performs better than greedy longest substring matching based delexicalization. This is because string matching based delexicalization can often result in erroneous match, and cannot match out-of-vocab words/phrases. Using attention BiRNN parser and iterative delexicalization, our model also achieves state-of-the-art slot labeling F1 score of $\mathbf{93.24}\%$ and intent accuracy $\mathbf{98.57}\%$. Our baseline Slot Gated parser perform slightly worse on SNIPS than [5] since we use default parameter settings without parameter tuning. In Table 4, we compare the performance of all models on Facebook dataset. Once again we observe that our iterative delexicalization algorithm achieves significant improvement of slot tagging F1 score as well as intent accuracy. The gain is particularly high due to the presence of OOD message slots in Facebook, which show a large semantic variability in the test set.

Table 3: *Intent classification and slot labeling performance comparison of all models on benchmark SNIPS dataset.*

RNN Parser	Slot tagging F1 %	Intent accuracy %
Slot Gated BiRNN [5]	88.8	97.0
Capsule NLU [19]	91.8	97.3
Attention BiRNN (our baseline)	90.64	98.0
Attention BiRNN + CRF	91.91	98.0
Attention BiRNN + greedy delex.	92.56	98.29
Attention BiRNN + iterative delex.	**93.24**	**98.57**
Slot Gated BiRNN (our baseline)	85.25	93.14
Slot Gated BiRNN + greedy delex.	86.83	94.86
Slot Gated BiRNN + iterative delex.	**88.14**	**95.14**

We also perform some slot level error analysis to observe the advantage of our iterative delexicalization algorithm for out-of-distribution slots. For Facebook dataset, we divide the slots into two categories; first those of the form of *message text*, and the *other non-message* slots, since message texts can often have large semantic variability. Using the Attention BiRNN parser [3] as baseline, we plot the average F1 score for a particular train-test split for these two slot categories in Table 1. We observe that our iterative delexicalization algorithm significantly improves the F1 score of message text slots while maintaining similar F1 score for other slots.

Table 4: *Performance comparison of all models on Facebook dataset. We compute* 5 *fold average performance in this dataset due to smaller size.*

RNN Parser	Slot tagging F1 %	Intent accuracy %
Attention BiRNN (our baseline)	84.46	93.80
Attention BiRNN + CRF	86.79	93.80
Attention BiRNN + greedy delex.	85.65	94.31
Attention BiRNN + iterative delex.	**89.22**	**94.82**
Slot Gated BiRNN (our baseline)	84.86	93.46
Slot Gated BiRNN + greedy delex.	86.49	92.72
Slot Gated BiRNN + iterative delex.	**88.89**	**93.48**

5. Conclusion

State-of-the-art RNN based joint SLU models perform poorly on certain out-of-distribution slots which may encounter slot values with large semantic variability after deployment. Previous string matching based delexicalization techniques are inadequate to handle such slots since these slot values are mostly out-of-vocab. In this paper we propose a novel iterative delexicalization algorithm which exploits model uncertainty to improve delexicalization for such out-of-distribution slots. In addition, our model also enables faster model training for slots with large training vocabulary (e.g. movie/show titles), and supports some continuous learning without requiring model updates by simply maintaining an updated contemporary vocabulary. In experiments on benchmark and in-house datasets, we demonstrate significant improvement in SLU performance using our algorithm, thereby achieving state-of-the-art results.

6. References

[1] X. Li, Y. Chen, L. Li, J. Gao, and A. Çelikyilmaz, "End-to-end task-completion neural dialogue systems," in *Proc. of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Volume 1: Long Papers*, 2017, pp. 733–743.

[2] D. Hakkani-Tür, G. Tür, A. Celikyilmaz, Y.-N. Chen, J. Gao, L. Deng, and Y.-Y. Wang, "Multi-domain joint semantic frame parsing using bi-directional rnn-lstm." in *INTERSPEECH*, 2016, pp. 715–719.

[3] B. Liu and I. Lane, "Attention-based recurrent neural network models for joint intent detection and slot filling," in *Interspeech 2016*, 2016, pp. 685–689.

[4] Y. Kim, S. Lee, and K. Stratos, "ONENET: joint domain, intent, slot prediction for spoken language understanding," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop*, 2017, pp. 547–553.

[5] C.-W. Goo, G. Gao, Y.-K. Hsu, C.-L. Huo, T.-C. Chen, K.-W. Hsu, and Y.-N. Chen, "Slot-gated modeling for joint slot filling and intent prediction," in *Proc. of the 16th NAACL-HLT*, 2018.

[6] Y. Wang, Y. Shen, and H. Jin, "A bi-model based RNN semantic frame parsing model for intent detection and slot filling," in *Proc. of the 2018 NAACL-HLT, Volume 2 (Short Papers)*, 2018, pp. 309–314.

[7] A. Ray, Y. Shen, and H. Jin, "Robust spoken language understanding via paraphrasing," in *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018.*, 2018, pp. 3454–3458.

[8] L. Heck and D. Hakkani-Tür, "Exploiting the semantic web for unsupervised spoken language understanding," in *Spoken Language Technology Workshop (SLT), 2012 IEEE*. IEEE, 2012, pp. 228–233.

[9] L. P. Heck, D. Hakkani-Tür, and G. Tür, "Leveraging knowledge graphs for web-scale unsupervised semantic parsing," in *INTERSPEECH 2013*, 2013, pp. 1594–1598.

[10] J. Lee, D. Kim, R. Sarikaya, and Y.-B. Kim, "Coupled representation learning for domains, intents and slots in spoken language understanding," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 714–719.

[11] J. Juraska, P. Karagiannis, K. Bowden, and M. A. Walker, "A deep ensemble model with slot alignment for sequence-to-sequence natural language generation," in *Proc. of the 2018 NAACL-HLT, Volume 1 (Long Papers)*, 2018, pp. 152–162.

[12] D. Zeman and P. Resnik, "Cross-language parser adaptation between related languages," in *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*, 2008.

[13] P. Denis and M. Dehouck, "Delexicalized word embeddings for cross-lingual dependency parsing," in *Proc. of the EACL 2017, Volume 1: Long Papers*, 2017, pp. 241–250.

[14] L. Dong and M. Lapata, "Language to logical form with neural attention," in *Proc. of the 54th ACL 2016*, 2016.

[15] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu *et al.*, "Using recurrent neural networks for slot filling in spoken language understanding," *IEEE/ACM Trans. on Audio, Speech and Language Processing (TASLP)*, vol. 23, no. 3, pp. 530–539, 2015.

[16] K. Yao, B. Peng, G. Zweig, D. Yu, X. Li, and F. Gao, "Recurrent conditional random field for language understanding," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*, 2014, pp. 4077–4081.

[17] Y. Kim, K. Stratos, and D. Kim, "Adversarial adaptation of synthetic or stale data," in *Proc. of ACL 2017*, 2017, pp. 1297–1307.

[18] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril, M. Primet, and J. Dureau, "Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces," *CoRR*, vol. abs/1805.10190, 2018.

[19] C. Zhang, Y. Li, N. Du, W. Fan, and P. S. Yu, "Joint slot filling and intent detection via capsule neural networks," *CoRR*, vol. abs/1812.09471, 2018.