A Stack-Propagation Framework with Token-Level Intent Detection for Spoken Language Understanding

Libo Qin, Wanxiang Che[*], Yangming Li, Haoyang Wen, Ting Liu
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{lbqin,car,yangmingli,hywen,tliu}@ir.hit.edu.cn

Abstract

Intent detection and slot filling are two main tasks for building a spoken language understanding (SLU) system. The two tasks are closely tied and the slots often highly depend on the intent. In this paper, we propose a novel framework for SLU to better incorporate the intent information, which further guides the slot filling. In our framework, we adopt a joint model with Stack-Propagation which can directly use the intent information as input for slot filling, thus to capture the intent semantic knowledge. In addition, to further alleviate the error propagation, we perform the token-level intent detection for the Stack-Propagation framework. Experiments on two publicly datasets show that our model achieves the state-of-the-art performance and outperforms other previous methods by a large margin. Finally, we use the Bidirectional Encoder Representation from Transformer (BERT) model in our framework, which further boost our performance in SLU task.

1 Introduction

Spoken language understanding (SLU) is a critical component in task-oriented dialogue systems. It usually consists of intent detection to identify users' intents and slot filling task to extract semantic constituents from the natural language utterances (Tur and De Mori, 2011). As shown in Table 1, given a movie-related utterance "*watch action movie*", there are different slot labels for each token and an intent for the whole utterance.

Usually, intent detection and slot filling are implemented separately. But intuitively, these two tasks are not independent and the slots often highly depend on the intent (Goo et al., 2018). For example, if the intent of a utterance is

Sentence	*watch*	*action*	*movie*
Gold Slots	O	B-movie_name	I-movie_name
Gold Intent	WatchMovie		

Table 1: An example with intent and slot annotation (BIO format), which indicates the slot of movie name from an utterance with an intent WatchMovie.

WatchMovie, it is more likely to contain the slot movie_name rather than the slot music_name. Hence, it is promising to incorporate the intent information to guide the slot filling.

Considering this strong correlation between the two tasks, some joint models are proposed based on the multi-task learning framework (Zhang and Wang, 2016; Hakkani-Tür et al., 2016; Liu and Lane, 2016) and all these models outperform the pipeline models via mutual enhancement between two tasks. However, their work just modeled the relationship between intent and slots by sharing parameters. Recently, some work begins to model the intent information for slot filling explicitly in joint model. Goo et al. (2018) and Li et al. (2018) proposed the gate mechanism to explore incorporating the intent information for slot filling. Though achieving the promising performance, their models still suffer from two issues including: (1) They all adopt the gate vector to incorporate the intent information. In the paper, we argue that it is risky to simply rely on the gate function to summarize or memorize the intent information. Besides, the interpretability of how the intent information guides slot filling procedure is still weak due to the interaction with hidden vector between the two tasks. (2) The utterance-level intent information they use for slot filling may mislead the prediction for all slots in an utterance if the predicted utterance-level intent is incorrect.

In this paper, we propose a novel framework to address both two issues above. For the first issue, inspired by the Stack-Propagation which was

[*] Email corresponding.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 2078–2087,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

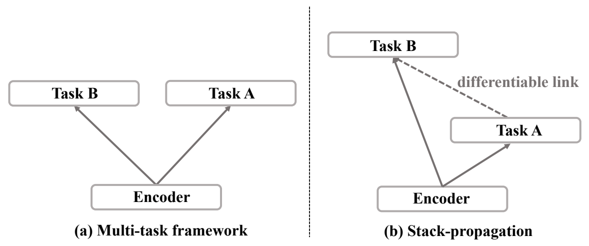

Figure 1: Multi-task framework vs. Stack-Propagation.

proposed by Zhang and Weiss (2016) to leverage the POS tagging features for parsing and achieved good performance, we propose a joint model with Stack-Propagation for SLU tasks. Our framework directly use the output of the intent detection as the input for slot filling to better guide the slot prediction process. In addition, the framework make it easy to design oracle intent experiment to intuitively show how intent information enhances slot filling task. For the second issue, we perform a token-level intent prediction in our framework, which can provide the token-level intent information for slot filling. If some token-level intents in the utterance are predicted incorrectly, other correct token-level intents will still be useful for the corresponding slot prediction. In practice, we use a self-attentive encoder for intent detection to capture the contextual information at each token and hence predict an intent label at each token. The intent of an utterance is computed by voting from predictions at each token of the utterance. This token-level prediction, like ensemble neural networks (Lee et al., 2016), reduces the predicted variance to improve the performance of intent detection. And it fits better in our Stack-Propagation framework, where intent detection can provide token-level intent features and retain more useful intent information for slot filling.

We conduct experiments on two benchmarks SNIPS (Coucke et al., 2018) and ATIS (Goo et al., 2018) datasets. The results of both experiments show the effectiveness of our framework by outperforming the current state-of-the-art methods by a large margin. Finally, Bidirectional Encoder Representation from Transformer (Devlin et al., 2018, BERT), as the pre-trained model, is used to further boost the performance of our model.

To summarize, the contributions of this work are as follows:

- We propose a Stack-Propagation framework in SLU task, which can better incorporate the intent semantic knowledge to guide the slot filling and make our joint model more interpretable.
- We perform the token-level intent detection for Stack-Propagation framework, which improves the intent detection performance and further alleviate the error propagation.
- We present extensive experiments demonstrating the benefit of our proposed framework. Our experiments on two publicly available datasets show substantial improvement and our framework achieve the state-of-the-art performance.
- We explore and analyze the effect of incorporating BERT in SLU tasks.

For reproducibility, our code for this paper is publicly available at `https://github.com/LeePleased/StackPropagation-SLU`.

2 Background

In this section, we will describe the formulation definition for intent detection and slot filling, and then we give a brief description of the multi-task framework and the joint model with Stack-Propagation framework.

2.1 Intent Detection and Slot Filling

Intent detection can be seen as a classification problem to decide the intent label o^I of an utterance. Slot filling is a sequence labeling task that maps an input word sequence $\mathbf{x} = (x_1, \ldots, x_T)$ to slots sequence $\mathbf{o}^S = (o_1^S, \ldots, o_T^S)$.

2.2 Multi-task Framework vs. Stack-Propagation

For two correlative tasks *Task A* and *Task B*, multi-task framework which is shown in Figure 1 (a) can learn the correlations between these two tasks by the shared encoder. However, the basic multi-task framework cannot provide features from up-stream task to down-stream task explicitly. The joint model with Stack-Propagation framework which is shown in Figure 1 (b) can mitigate the shortcoming. In this figure, *Task B* can leverage features of *Task A* without breaking the differentiability in Stack-Propagation framework and promote each other by joint learning at the same time.

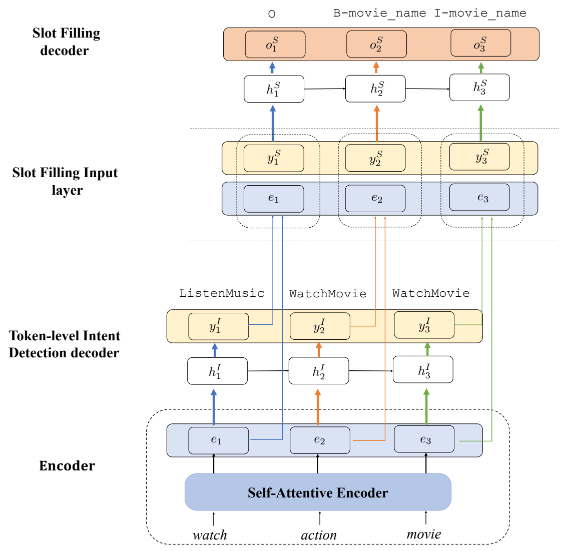

Figure 2: Illustration of our Stack-Propagation framework for joint intent detection and slot filling. It consists of one shared self-attentive encoder and two decoders. The output distribution of intent detection network and the representations from encoder are concatenated as the input for slot filling.

3 Approach

In this section, we will describe our Stack-Propagation framework for SLU task. The architecture of our framework is demonstrated in Figure 2, which consists of an encoder and two decoders. First, the encoder module uses one shared self-attentive encoder to represent an utterance, which can grasp the shared knowledge between two tasks. Then, the intent-detection decoder performs a token-level intent detection. Finally, our Stack-Propagation framework leverages the explicit token-level intent information for slot filling by concatenating the output of intent detection decoder and the representations from encoder as the input for slot filling decoder. Both intent detection and slot filling are optimized simultaneously via a joint learning scheme.

3.1 Self-Attentive Encoder

In our Stack-Propagation framework, intent detection task and slot filling task share one encoder, In the self-attentive encoder, we use BiLSTM with self-attention mechanism to leverage both advantages of temporal features and contextual information, which are useful for sequence labeling tasks (Zhong et al., 2018; Yin et al., 2018).

The BiLSTM (Hochreiter and Schmidhuber, 1997) reads input utterance $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_T)$ (T is the number of tokens in the input utterance) forwardly and backwardly to produce context-sensitive hidden states $\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_T)$ by repeatedly applying the recurrence $\mathbf{h}_i = \text{BiLSTM}\left(\phi^{\text{emb}}(x_i), \mathbf{h}_{i-1}\right)$.

Self-attention is a very effective method of leveraging context-aware features over variable-length sequences for natural language processing tasks (Tan et al., 2018; Zhong et al., 2018). In our case, we use self-attention mechanism to capture the contextual information for each token. In this paper, we adopt Vaswani et al. (2017), where we first map the matrix of input vectors $\mathbf{X} \in \mathbb{R}^{T \times d}$ (d represents the mapped dimension) to queries ($\mathbf{Q}$), keys ($\mathbf{K}$) and values ($\mathbf{V}$) matrices by using different linear projections and the self-attention output $\mathbf{C} \in \mathbb{R}^{T \times d}$ is a weighted sum of values:

$$\mathbf{C} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}. \tag{1}$$

After obtaining the output of self-attention and BiLSTM. We concatenate these two representations as the final encoding representation:

$$\mathbf{E} = \mathbf{H} \oplus \mathbf{C}, \tag{2}$$

where $\mathbf{E} \in \mathbb{R}^{T \times 2d}$ and $\oplus$ is concatenation operation.

3.2 Token-Level Intent Detection Decoder

In our framework, we perform a token-level intent detection, which can provide token-level intent features for slot filling, different from regarding the intent detection task as the sentence-level classification problem (Liu and Lane, 2016). The token-level intent detection method can be formalized as a sequence labeling problem that maps a input word sequence $\mathbf{x} = (x_1, ..., x_T)$ to sequence of intent label $\mathbf{o}^I = (o_1^I, ..., o_T^I)$. In training time, we set the sentence's intent label as every token's gold intent label. The final intent of an utterance is computed by voting from predictions at each token of the utterance.

The self-attentive encoder generates a sequence of contextual representations $\mathbf{E} = (\mathbf{e}_1, ..., \mathbf{e}_T)$ and each token can grasp the whole contextual information by self-attention mechanism. We use a unidirectional LSTM as the intent detection network. At each decoding step i, the decoder state $\mathbf{h}_i^I$ is calculated by previous decoder state $\mathbf{h}_{i-1}^I$, the previous emitted intent label distribution $\mathbf{y}_{i-1}^I$ and the aligned encoder hidden state $\mathbf{e}_i$:

$$\mathbf{h}_i^I = f\left(\mathbf{h}_{i-1}^I, \mathbf{y}_{i-1}^I, \mathbf{e}_i\right). \tag{3}$$

Then the decoder state $\mathbf{h}_i^I$ is utilized for intent detection:

$$\mathbf{y}_i^I = \text{softmax}\left(\mathbf{W}_h^I \mathbf{h}_i^I\right), \quad (4)$$
$$o_i^I = \text{argmax}(\mathbf{y}_i^I), \quad (5)$$

where $\mathbf{y}_i^I$ is the intent output distribution of the ith token in the utterance; o_i^I represents the intent lable of ith token and $\mathbf{W}_h^I$ are trainable parameters of the model.

The final utterance result o^I is generated by voting from all token intent results:

$$o^I = \text{argmax} \sum_{i=1}^{m} \sum_{j=1}^{n_I} \alpha_j \mathbb{1}[o_i^I = j], \quad (6)$$

where m is the length of utterance and n_I is the number of intent labels; α_j denotes a 0-1 vector $\alpha \in \mathbb{R}^{n_I}$ of which the jth unit is one and the others are zero; argmax indicates the operation returning the indices of the maximum values in α.

By performing the token-level intent detection, there are mainly two advantages:

1. Performing the token-level intent detection can provide features at each token to slot filling in our Stack-Propagation framework, which can ease the error propagation and retain more useful information for slot filling. Compared with sentence-level intent detection (Li et al., 2018), if the intent of the whole sentence is predicted wrongly, the wrong intent would possibly apply a negative impact on all slots. However, in token-level intent detection, if some tokens in the utterance predicted wrongly, other correct token-level intent information will still be useful for the corresponding slot filling.

2. Since each token can grasp the whole utterance contextual information by using the self-attentive encoder, we can consider predictions at each token in an utterance as individual prediction to the intent of this utterance. And therefore, like ensemble neural networks, this approach will reduce the predicted variance and improve the performance of intent detection. The experiments section empirically demonstrate the effectiveness of the token-level intent detection.

3.3 Stack-propagation for Slot Filling

In this paper, one of the advantages of our Stack-Propagation framework is directly leveraging the explicit intent information to constrain the slots into a specific intent and alleviate the burden of slot filling decoder. In our framework, we compose the input units for slot filling decoder by concatenating the intent output distribution $\mathbf{y}_i^I$ and the aligned encoder hidden state $\mathbf{e}_i$.

For the slot-filling decoder, we similarly use another unidirectional LSTM as the slot-filling decoder. At the decoding step i, the decoder state $\mathbf{h}_i^S$ can be formalized as:

$$\mathbf{h}_i^S = f\left(\mathbf{h}_{i-1}^S, \mathbf{y}_{i-1}^S, \mathbf{y}_i^I \oplus \mathbf{e}_i\right), \quad (7)$$

where $\mathbf{h}_{i-1}^S$ is the previous decoder state; $\mathbf{y}_{i-1}^S$ is the previous emitted slot label distribution.

Similarily, the decoder state $\mathbf{h}_i^I$ is utilized for slot filling:

$$\mathbf{y}_i^S = \text{softmax}\left(\mathbf{W}_h^S \mathbf{h}_i^S\right), \quad (8)$$
$$o_i^S = \text{argmax}(\mathbf{y}_i^S), \quad (9)$$

where o_i^S is the slot label of the ith word in the utterance.

3.4 Joint Training

Another major difference between existing joint work (Zhang and Wang, 2016; Goo et al., 2018) and our framework is the training method for intent detection, where we convert the sentence-level classification task into token-level prediction to directly leverage token-level intent information for slot filling. And the intent detection objection is formulated as:

$$\mathcal{L}_1 \triangleq -\sum_{j=1}^{m} \sum_{i=1}^{n_I} \hat{\mathbf{y}}_j^{i,I} \log\left(\mathbf{y}_j^{i,I}\right). \quad (10)$$

Similarly, the slot filling task objection is defined as:

$$\mathcal{L}_2 \triangleq -\sum_{j=1}^{m} \sum_{i=1}^{n_S} \hat{\mathbf{y}}_j^{i,S} \log\left(\mathbf{y}_j^{i,S}\right), \quad (11)$$

where $\hat{\mathbf{y}}_j^{i,I}$ and $\hat{\mathbf{y}}_j^{i,S}$ are the gold intent label and gold slot label separately; n_S is the number of slot labels.

To obtain both slot filling and intent detection jointly, the final joint objective is formulated as

$$\mathcal{L}_\theta = \mathcal{L}_1 + \mathcal{L}_2. \quad (12)$$

Through the joint loss function, the shared representations learned by the shared self-attentive encoder can consider two tasks jointly and further ease the error propagation compared with pipeline models (Zhang and Wang, 2016).

4 Experiments

4.1 Experimental Settings

To evaluate the efficiency of our proposed model, we conduct experiments on two benchmark datasets. One is the publicly ATIS dataset (Hemphill et al., 1990) containing audio recordings of flight reservations, and the other is the custom-intent-engines collected by Snips (SNIPS dataset) (Coucke et al., 2018). [1] Both datasets used in our paper follows the same format and partition as in Goo et al. (2018). The dimensionalities of the word embedding is 256 for ATIS dataset and 512 for SNIPS dataset. The self-attentive encoder hidden units are set as 256. L2 regularization is used on our model is 1×10^{-6} and dropout ratio is adopted is 0.4 for reducing overfit. We use Adam (Kingma and Ba, 2014) to optimize the parameters in our model and adopted the suggested hyper-parameters for optimization. For all the experiments, we select the model which works the best on the dev set, and then evaluate it on the test set.

4.2 Baselines

We compare our model with the existing baselines including:

- **Joint Seq**. Hakkani-Tür et al. (2016) proposed a multi-task modeling approach for jointly modeling domain detection, intent detection, and slot filling in a single recurrent neural network (RNN) architecture.
- **Attention BiRNN**. Liu and Lane (2016) leveraged the attention mechanism to allow the network to learn the relationship between slot and intent.
- **Slot-Gated Atten**. Goo et al. (2018) proposed the slot-gated joint model to explore the correlation of slot filling and intent detection better.
- **Self-Attentive Model**. Li et al. (2018) proposed a novel self-attentive model with the intent augmented gate mechanism to utilize the semantic correlation between slot and intent.
- **Bi-Model**. Wang et al. (2018) proposed the Bi-model to consider the intent and slot filling cross-impact to each other.
- **CAPSULE-NLU.** Zhang et al. (2019) proposed a capsule-based neural network model with a dynamic routing-by-agreement schema to accomplish slot filling and intent detection.
- **SF-ID Network.** (E et al., 2019) introduced an SF-ID network to establish direct connections for the slot filling and intent detection to help them promote each other mutually.

For the *Joint Seq*, *Attention BiRNN*, *Slot-gated Atten*, *CAPSULE-NLU* and *SF-ID Network*, we adopt the reported results from Goo et al. (2018); Zhang et al. (2019); E et al. (2019). For the *Self-Attentive Model*, *Bi-Model*, we re-implemented the models and obtained the results on the same datasets.[2]

4.3 Overall Results

Following Goo et al. (2018), we evaluate the SLU performance of slot filling using F1 score and the performance of intent prediction using accuracy, and sentence-level semantic frame parsing using overall accuracy. Table 2 shows the experiment results of the proposed models on SNIPS and ATIS datasets.

From the table, we can see that our model significantly outperforms all the baselines by a large margin and achieves the state-of-the-art performance. In the SNIPS dataset, compared with the best prior joint work *Bi-Model*, we achieve 0.7% improvement on Slot (F1) score, 0.8% improvement on Intent (Acc) and 3.1% improvement on Overall (Acc). In the ATIS dataset, we achieve 0.4% improvement on Slot (F1) score, 0.5% improvement on Intent (Acc) and 0.8% improvement on Overall (Acc). This indicates the effectiveness of our Stack-Propagation framework. Especially, our framework gains the largest improvements on sentence-level semantic frame accuracy,

[1] https://github.com/snipsco/nlu-benchmark/tree/master/2017-06-custom-intent-engines

[2] All experiments are conducted on the publicly datasets provided by Goo et al. (2018), *Self-Attentive Model* and *Bi-Model* don't have the reported result on the same datasets or they did different preprocessing. For directly comparison, we re-implemented the models and obtained the results on the ATIS and SNIPS datasets preprocessed by Goo et al. (2018). Because all baselines and our model don't apply CRF layer, we just report the best performance of *SF-ID Network* without CRF. It's noticing that our model does outperform *SF-ID Network* with CRF layer.

Model	**SNIPS**			**ATIS**		
	Slot (F1)	Intent (Acc)	Overall (Acc)	Slot (F1)	Intent (Acc)	Overall (Acc)
Joint Seq (Hakkani-Tür et al., 2016)	87.3	96.9	73.2	94.3	92.6	80.7
Attention BiRNN (Liu and Lane, 2016)	87.8	96.7	74.1	94.2	91.1	78.9
Slot-Gated Full Atten (Goo et al., 2018)	88.8	97.0	75.5	94.8	93.6	82.2
Slot-Gated Intent Atten (Goo et al., 2018)	88.3	96.8	74.6	95.2	94.1	82.6
Self-Attentive Model (Li et al., 2018)	90.0	97.5	81.0	95.1	96.8	82.2
Bi-Model (Wang et al., 2018)	93.5	97.2	83.8	95.5	96.4	85.7
CAPSULE-NLU (Zhang et al., 2019)	91.8	97.3	80.9	95.2	95.0	83.4
SF-ID Network (E et al., 2019)	90.5	97.0	78.4	95.6	96.6	86.0
Our model	**94.2***	**98.0***	**86.9***	**95.9***	**96.9***	**86.5***
Oracle (Intent)	96.1	-	-	96.0	-	-

Table 2: Slot filling and intent detection results on two datasets. The numbers with * indicate that the improvement of our model over all baselines is statistically significant with $p < 0.05$ under t-test.

we attribute this to the fact that our framework directly take the explicit intent information into consideration can better help grasp the relationship between the intent and slots and improve the SLU performance.

To see the role of intent information for SLU tasks intuitively, we also present the result when using the gold intent information.[3] The result is shown in the *oracle* row of Table 2. From the result, we can see that further leveraging better intent information will lead to better slot filling performance. The result also verifies our assumptions that intent information can be used for guiding the slots prediction.

4.4 Analysis

In Section 4.3, significant improvements among all three metrics have been witnessed on both two publicly datasets. However, we would like to know the reason for the improvement. In this section, we first explore the effect of Stack-Propagation framework. Next, we study the effect of our proposed token-level intent detection mechanism. Finally, we study the effect of self-attention mechanism in our framework.

4.4.1 Effect of Stack-Propagation Framework

To verify the effectiveness of the Stack-Propagation framework. we conduct experiments with the following ablations:

1) We conduct experiments to incorporate intent information by using gate-mechanism which is similar to Goo et al. (2018), providing the intent information by interacting with the slot filling decoder by gate function.[4] We refer it as *gate-mechanism.*

2) We conduct experiments on the pipelined model where the intent detection and slot filling has their own self-attentive encoder separately. The other model components keep the same as our framework. We name it as *pipelined model.*

Table 3 gives the result of the comparison experiment. From the result of *gate-mechanism* row, we can observe that without the Stack-Propagation learning and simply using the gate-mechanism to incorporate the intent information, the slot filling (F1) performance drops significantly, which demonstrates that directly leverage the intent information with Stack-Propagation can improve the slot filling performance effectively than using the gate mechanism. Besides, we can see that the intent detection (Acc) and overall accuracy (Acc) decrease a lot. We attribute it to the fact that the bad slot filling performance harms the intent detection and the whole sentence semantic performance due to the joint learning scheme.

Besides, from the *pipeline model* row of Table 3, we can see that without shared encoder, the performance on all metrics declines significantly. This shows that Stack-Propagation model can learn the correlation knowledge which may promote each other and ease the error propagation effectively.

[3]During the inference time, we concatenate the gold intent label distribution (one-hot vector) and the aligned encoder hidden state e_i as the composed input for slot filling decoder. To keep the train and test procedure as the same, we replace our intent distribution as one-hot intent information for slot filling when train our model in our oracle experiments setting.

[4]For directly comparison, we still perform the token-level intent detection.

Model	SNIPS			ATIS		
	Slot (F1)	Intent (Acc)	Overall (Acc)	Slot (F1)	Intent (Acc)	Overall (Acc)
gate-mechanism	92.2	97.6	82.4	95.3	96.2	83.4
pipelined model	90.8	97.6	81.8	95.1	96.1	82.3
sentence intent augmented	93.7	97.5	86.1	95.5	96.7	85.8
lstm+last-hidden	-	97.1	-	-	95.2	-
lstm+token-level	-	97.5	-	-	96.0	-
without self-attention	94.1	97.8	86.6	95.6	96.6	86.2
Our model	**94.2**	**98.0**	**86.9**	**95.9**	**96.9**	**86.5**

Table 3: The SLU performance on baseline models compared with our Stack-Propagation model on two datasets.

4.4.2 Effect of Token-Level Intent Detection Mechanism

In this section, we study the effect of the proposed token-level intent detection with the following ablations:

1) We conduct the sentence-level intent detection in intent detection separately, which utilizes the last hidden vector of BiLSTM encoder for intent detection. We refer it to *lstm + last-hidden*. For comparison, our token-level intent detection without joint learning with slot filling is named as *lstm+token-level* in Table 3.

2) We conduct a joint learning framework that slot filling uses the utterance-level intent information rather than token-level intent information for each token, similar to intent-gated mechanism (Li et al., 2018), which is named as *sentence intent augmented.*

We show these two comparison experiments results in the first block of table 3. From the result, we can see that the token-level intent detection obtains better performance than the utterance-level intent detection. We believe the reason is that intent prediction on each token has similar advantage to ensemble neural networks, which can reduce the predicted variance to improve the intent performance. As a result, our framework can provide more useful intent information for slot filling by introducing token-level intent detection.

In addition, we can observe that if we only provide the sentence-level intent information for slot filling decoder, we obtain the worse results, which demonstrates the significance and effectiveness of incorporating token-level intent information. The main reason for this can be that incorporating the token-level intent information can retain useful features for each token and ease the error propagation.

4.4.3 Effect of Self-attention Mechanism

We further investigate the benefits of self-attention mechanism in our framework. We conduct the comparison experiments with the same framework except the self-attention encoder is replaced with BiLSTM.

Results are shown in the *without self-attention* row of Table 3. We can observe the self-attention mechanism can further improve the SLU performance. We attribute this to the fact that self-attention mechanism can capture the contextual information for each token. Without the self-attention mechanism, it will harm the intent detection and have bad influence on slot filling task by joint learning.

It is noticeable that even without the self-attention mechanism, our framework still performs the state-of-the-art Bi-model model (Li et al., 2018), which again demonstrates the effectiveness and robustness of our other framework components.

4.5 Effect of BERT

Finally, we also conduct experiments to use pre-trained model, BERT (Devlin et al., 2018), to boost SLU performance. In this section, we replace the self-attentive encoder by *BERT base* model with the fine-tuning approach and keep other components as same with our framework.

Table 4 gives the results of BERT model on ATIS and SNIPS datasets. From the table, the BERT model performs remarkably well on both two datasets and achieves a new state-of-the-art performance, which indicates the effectiveness of a strong pre-trained model in SLU tasks. We attribute this to the fact that pre-trained models can provide rich semantic features, which can help to improve the performance on SLU tasks. In addition, our *model + BERT* outperforms the *BERT SLU* (Chen et al., 2019) which apply BERT for

Model	SNIPS			ATIS		
	Slot (F1)	Intent (Acc)	Overall (Acc)	Slot (F1)	Intent (Acc)	Overall (Acc)
Our model	94.2	98.0	86.9	95.9	96.9	86.5
Intent detection (BERT)	-	97.8	-	-	96.5	-
Slot filling (BERT)	95.8	-	-	95.6	-	-
BERT SLU (Chen et al., 2019)	97.0	98.6	92.8	96.1	97.5	88.2
Our model + BERT	97.0	99.0	92.9	96.1	97.5	88.6

Table 4: The SLU performance on BERT-based model on two datasets.

joint the two tasks and there is no explicit interaction between intent detection and slot filling in two datasets in overall acc metric. It demonstrates that our framework is effective with BERT.

Especially, we also conduct experiments of intent detection task and slot filling separately based on BERT model. For intent detection, we put the special `[CLS]` word embedding into a classification layer to classify the intent. For slot filling, we feed the final hidden representation $\mathbf{h}_{\text{BERT}_i} \in \mathbb{R}^d$ for each token i [5] into a classification layer over the slot tag set. The results are also shown in the Table 4. From the result, we can see that the slot filling (F1) and intent detection accuracy (Acc) is lower than our joint model based on BERT, which again demonstrates the effectiveness of exploiting the relationship between these two tasks.

5 Related Work

Slot filling can be treated as a sequence labeling task, and the popular approaches are conditional random fields (CRF) (Raymond and Riccardi, 2007) and recurrent neural networks (RNN) (Xu and Sarikaya, 2013; Yao et al., 2014). The intent detection is formulated as an utterance classification problem, and different classification methods, such as support vector machine (SVM) and RNN (Haffner et al., 2003; Sarikaya et al., 2011), has proposed to solve it.

Recently, there are some joint models to overcome the error propagation caused by the pipelined approaches. Zhang and Wang (2016) first proposed the joint work using RNNs for learning the correlation between intent and slots. Hakkani-Tür et al. (2016) proposed a single recurrent neural network for modeling slot filling and intent detection jointly. Liu and Lane (2016) proposed an attention-based neural network for modeling the two tasks jointly. All these models outperform the pipeline models via mutual enhancement between two tasks. However, these joint models did not model the intent information for slots explicitly and just considered their correlation between the two tasks by sharing parameters.

Recently, some joint models have explored incorporating the intent information for slot filling. Goo et al. (2018) utilize a slot-gated mechanism as a special gate function to model the relationship between the intent detection and slot filling. Li et al. (2018) proposed the intent augmented gate mechanism to utilize the semantic correlation between slots and intent. Our framework is significantly different from their models including: (1) both of their approaches utilize the gate mechanism to model the relationship between intent and slots. While in our model, to directly leverage the intent information in the joint model, we feed the predicted intent information directly into slot filling with Stack-Propagation framework. (2) They apply the sentence-level intent information for each word while we adopt the token-level intent information for slot filling and further ease the error propagation. Wang et al. (2018) propose the Bi-model to consider the cross-impact between the intent and slots and achieve the state-of-the-art result. Zhang et al. (2019) propose a hierarchical capsule neural network to model the the hierarchical relationship among word, slot, and intent in an utterance. E et al. (2019) introduce an SF-ID network to establish the interrelated mechanism for slot filling and intent detection tasks. Compared with their works, our model can directly incorporate the intent information for slot filling explicitly with Stack-Propagation which makes the interaction procedure more interpretable, while their model just interacts with hidden state implicitly between two tasks.

6 Conclusion

In this paper, we propose a joint model for spoken language understanding with Stack-Propagation to better incorporate the intent information for slot

[5] We only consider the first subword label if a word is broken into multiple subwords

filling. In addition, we perform the token-level intent detection to improve the intent detection performance and further ease the error propagation. Experiments on two datasets show the effectiveness of the proposed models and achieve the state-of-the-art performance. Besides, we explore and analyze the effect of incorporating strong pre-trained BERT model in SLU tasks. With BERT, the result reaches a new state-of-the-art level.

Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by the National Natural Science Foundation of China (NSFC) via grant 61976072, 61632011 and 61772153.

References

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909.*

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190.*

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805.*

Haihong E, Peiqing Niu, Zhongfu Chen, and Meina Song. 2019. A novel bi-directional interrelated model for joint intent detection and slot filling. In *Proc. of ACL.*

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proc. of NAACL.*

Patrick Haffner, Gokhan Tur, and Jerry H Wright. 2003. Optimizing svms for complex call classification. In *In Proc. of ICASSP.*

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech.*

Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990.*

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8).

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Stefan Lee, Senthil Purushwalkam Shiva Prakash, Michael Cogswell, Viresh Ranjan, David Crandall, and Dhruv Batra. 2016. Stochastic multiple choice learning for training diverse deep ensembles. In *NIPS.*

Changliang Li, Liang Li, and Ji Qi. 2018. A self-attentive model with gate mechanism for spoken language understanding. In *Proc. of EMNLP.*

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454.*

Christian Raymond and Giuseppe Riccardi. 2007. Generative and discriminative algorithms for spoken language understanding. In *Eighth Annual Conference of the International Speech Communication Association.*

Ruhi Sarikaya, Geoffrey E Hinton, and Bhuvana Ramabhadran. 2011. Deep belief nets for natural language call-routing. In *ICASSP.*

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proc. of AAAI.*

Gokhan Tur and Renato De Mori. 2011. *Spoken language understanding: Systems for extracting semantic information from speech.* John Wiley & Sons.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, $\mathcal{L}$ *ukasz* Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS.*

Yu Wang, Yilin Shen, and Hongxia Jin. 2018. A bi-model based rnn semantic frame parsing model for intent detection and slot filling. In *Proc. of ACL.*

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding.*

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. Spoken language understanding using long short-term memory neural networks. In *SLT.*

Qingyu Yin, Yu Zhang, Wei-Nan Zhang, Ting Liu, and William Yang Wang. 2018. Deep reinforcement learning for chinese zero pronoun resolution. In *Proc. of ACL.*

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip Yu. 2019. Joint slot filling and intent detection via capsule neural networks. In *Proc. of ACL.*

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *Proc. of IJCAI*.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proc. of ACL*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *Proc. of ACL*.