Syntactic Graph Convolutional Network for Spoken Language Understanding

Keqing He[1]*, Shuyu Lei[2], Yushu Yang[2], Huixing Jiang[2], Zhongyuan Wang[2]
[1]Beijing University of Posts and Telecommunications, Beijing, China
[2]Meituan Group
{kqin,leishuyu}@bupt.edu.cn
{yangyushu,jianghuixing,wangzhongyuan02}@meituan.com

Abstract

Slot filling and intent detection are two major tasks for spoken language understanding. In most existing work, these two tasks are built as joint models with multi-task learning with no consideration of prior linguistic knowledge. In this paper, we propose a novel joint model that applies a graph convolutional network over dependency trees to integrate the syntactic structure for learning slot filling and intent detection jointly. Experimental results show that our proposed model achieves state-of-the-art performance on two public benchmark datasets and outperforms existing work. At last, we apply the BERT model to further improve the performance on both slot filling and intent detection.

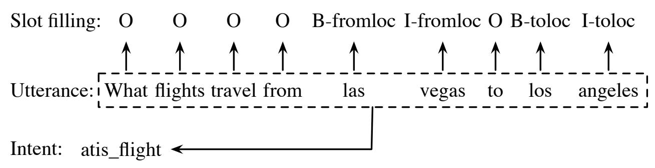
1 Introduction

Spoken Language Understanding (SLU) plays a vital role in a task-oriented dialogue system. Slot filling and intent detection (Tur and De Mori, 2011) are two major tasks for SLU as shown in Figure 1(a). Slot filling aims to obtain the semantic structure for the utterance. Meanwhile, intent detection annotates the categorical intent of the utterance.

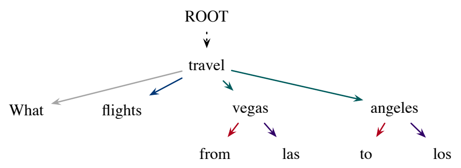
In typical pipeline methods, slot filling and intent detection are built separately. Slot filling is implemented as a standard sequence labeling task (Yao et al., 2014) and intent detection is built as a classification task (Lai et al., 2015), respectively. Essentially, slot filling and intent detection impact mutually. Therefore, more prior work (Hakkani-Tür et al., 2016; Liu and Lane, 2016; Goo et al., 2018; Li et al., 2018; Wang et al., 2018; Zhang et al., 2018a; E et al., 2019; Qin et al., 2019) implement two aforementioned tasks jointly as multi-task learning and achieve more promising results than those pipeline methods. However, most prior work utilize sequential model, such as recurrent neural network, to accumulate the contextual representation for each word to implement SLU with no consideration of prior linguistic knowledge. Intuitively, slot filling and intent detection rely on indicative contextual words for disambiguation and suffer from the degradation on wide contexts. Syntactic dependency parse tree as shown in Figure 1(b), which provides linguistic dependency relation among words, has been shown generally beneficial in various NLP tasks such as machine reading comprehension (Zhang et al., 2019), neural machine translation (Chen et al., 2018) and relation extraction (Zhang et al., 2018b). The major reasons are that the dependency parse tree can capture long-range relations between words and contain implicit clues for disambiguation.

To access a better SLU, we emphasize that SLU model should utilize the dependency representation as prior linguistic knowledge. In this paper, we propose a joint SLU model that applies a Graph Convolutional Network (GCN) over dependency trees to integrate the syntactic structure for joint learning slot filling and intent detection, where the GCN can pool information over arbitrary dependency structures efficiently, which has been proven in (Zhang et al., 2018b). Concretely, our proposed model encode the utterance and output a contextual word representation via a bi-directional LSTM (Hochreiter and Schmidhuber, 1997), then a GCN over dependency tree take contextual word representation as input to

*The work was done when the first author was an intern at Meituan Group. The first two authors contribute equally.


Slot filling: O O O O B-fromloc I-fromloc O B-toloc I-toloc
Utterance: What flights travel from las vegas to los angeles
Intent: atis_flight

(a) An example of slot filling (BIO format) and intent detection for utterance "What flights travel from las vegas to los angeles".



(b) An example of dependency tree over the utterance "What flights travel from las vegas to los angeles".

Figure 1: An example of utterance in ATIS dataset.

obtain the syntactic structure representation for utterance. In addition, it is worth noting that existing dependency parsers are impossible to parse all the sentences exactly. Thus, a multi-head attention is utilized to fuse the syntactic representation with original contextual word representation against the errors caused by incomplete dependency parser, where the multi-head attention can supplement the syntactic representation with contextual word representation as a self-adaption manner. At last, the fused representation is applied to implement slot filling and intent detection jointly.

The experiments are conducted on two benchmarks SLU datasets: ATIS (Hemphill et al., 1990) and Snips (Coucke et al., 2018). The experimental results demonstrate that our proposed model outperforms the existing state-of-the-art approaches. At last, BERT model (Devlin et al., 2019), as a pre-trained model, is used to our proposed framework. The experimental results also show that our proposed model incorporated with BERT model can further improve the performance on both slot filling and intent detection.

The main contributions of this work are therefore include as follows: 1)We introduce a model that utilizes a GCN to integrate the syntactic structure for joint learning slot filling and intent detection, which to the best of our knowledge is the first work that syntactic structure and GCN are used to implement above two tasks jointly. 2) We utilize a multi-head attention to fuse the syntactic representation with contextual word representation against the errors caused by incomplete dependency parser. 3) We conduct our experiments on two public datasets, and our proposed model achieves new the state-of-the-art performance in overall accuracy metric.

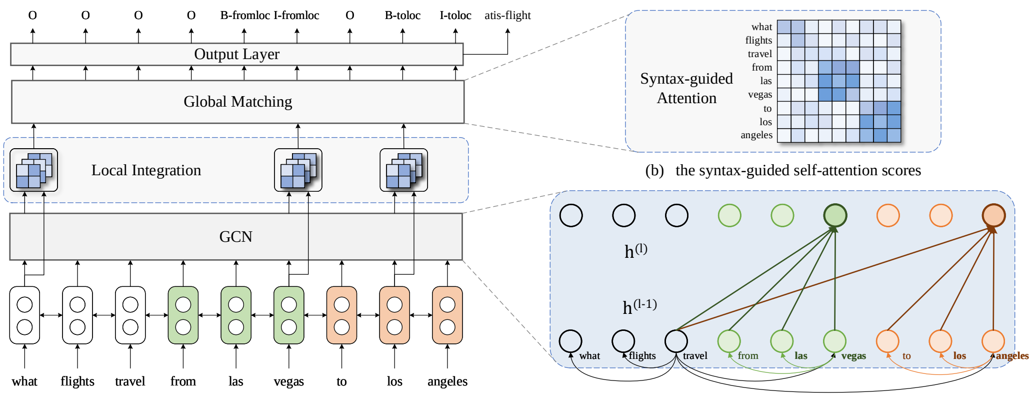
2 Methodology

In this section, we will describe our syntactic graph convolutional network for SLU tasks. The overall architecture of our model is demonstrated in Fig 2. We first use a BiLSTM encoder to obtain the contextual representation of an utterance. Then we perform multi-hop GCN propagation over the dependency tree initialized by the hidden states of the BiLSTM encoder to capture syntactic representation. Subsequently, we integrate the syntactic representation and the contextual hidden states via a feature aggregation layer. Finally, we pass the fused representation to the output layer for final predictions. Both slot filling and intent detection are optimized simultaneously via a joint learning scheme.

2.1 Notations

We now formally define the task of slot filling and intent detection. Let $\mathbf{X} = [x_1, \ldots, x_n]$ denotes a sentence, where n denotes the sequence length. We first use a syntactic parser to generate a dependency tree where each word represents a node. After obtaining a tree with n nodes, we can represent the graph structure with an $n * n$ adjacency matrix $\mathbf{A}$ where $A_{ij} = 1$ if there is an edge going from word x_i to word x_j. [1] Given the input sequence and the corresponding dependency tree, our goal is to predict the slot labels $\mathbf{o}^S = \left(o_1^S, \ldots, o_n^S\right)$ and the intent label o^I.

[1]We treat the dependency tree as an undirected graph, i.e.$\forall i, j, A_{ij} = A_{ji}$. We hypothesize that modeling edge directions and types does not offer additional discriminative power to the network because the GCN can capture adequately informative syntactic patterns for SLU. Besides, models with high complexity are prone to overfitting.



(a) the overall architecture of our proposed method

(b) the syntax-guided self-attention scores

(c) syntactic graph convolutional network

Figure 2: Spoken language understanding with a syntactic graph convolutional network. Fig (a) shows the overall architecture of our proposed method. Fig (b) displays the syntax-guided self-attention scores. Fig (c) shows one-layer detailed graph convolution computation for the word *"vegas"* and *"angeles"* for clarity. For local integration, we only show the computation of three timesteps. As we describe in the introduction, the syntactic structure lets a word focus more on its dependency words, such as $from \leftarrow vegas$ and $to \leftarrow angeles$. These syntactic constraints could enhance contextual representations to distinguish the departure city from the arrival city.

2.2 Syntactic Graph Convolutional Networks over Dependency Trees

The graph convolutional network(GCN) (Kipf and Welling, 2017) has been proved useful for encoding structural information in graphs. GCNs provide flexibility to represent diverse syntactic and semantic relationships between words. Essentially, GCN operates on a graph structure and compute representations for the nodes of the graph by looking at the neighborhood of the node. We can stack L layers of GCNs to account for neighbors that are L-hops away from the current node. Formally, in an L-layer GCN as Fig 2(c) shows, we denote the input vector as $h_i^{(l-1)}$ and output vector as $h_i^{(l)}$ where i represents the i-th node and l represents the l-th layer. Hence, the one-hop graph convolution operation can be written as $h_i^{(l)} = \sigma\left(\sum_{j=1}^{n} A_{ij} W^{(l)} h_j^{(l-1)} + b^{(l)}\right)$, where $W^{(l)}$ is a linear transformation, $b^{(l)}$ a bias term, and σ a nonlinear function (e.g., ReLU).

To initialize the first layer input vector $h^{(0)}$, we first feed the input word vectors into a BiLSTM network to generate contextualized representations. Note that our method is not limited to cooperate with BiLSTM, but any contextual encoder like ELMo(Peters et al., 2018), BERT(Devlin et al., 2019). We also conduct BERT based experiments for comparison. We will show empirically in Section 4.7 that both encoders substantially improve the performance over the original baselines.

Then, we perform the aforementioned graph convolution operation on dependency trees by converting each tree into its corresponding adjacency matrix $\mathbf{A}$, where $A_{ij} = 1$ if there is a dependency edge between words i and j. Adopted from (Zhang et al., 2018b), we also normalize the activations in the graph convolution before feeding it through the nonlinearity and adding self-loops to each node in the graph as $h_i^{(l)} = \sigma\left(\sum_{j=1}^{n} \tilde{A}_{ij} W^{(l)} h_j^{(l-1)}/d_i + b^{(l)}\right)$, where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ with $\mathbf{I}$ being the $n * n$ identity matrix and $d_i = \sum_{j=1}^{n} \tilde{A}_{ij}$ is the degree of token i in the resulting graph.

2.3 Feature Aggregation Mechanism: From Local To Global

After applying L-layer GCNs over dependency trees, we obtain the syntactic knowledge vector $h_i^{(L)}$ of each token x_i. To fuse syntactic representation and contextual representation, we propose the feature aggregation mechanism comprising of the local integration layer and global matching layer. The former builds strong interactions between syntactic vector and contextual vector of one word while the latter models connections at the overall utterance-level. The aggregation mechanism aims to integrate syntactic graph information and contextual representation and enable our model more robust to potential noise from the dependency parser.

Local Integration Given the syntactic representation $h_i^{(L)}$ and contextual representation $h_i^{(0)}$ of word

x_i, local integration intends to control how much information in $h_i^{(L)}$ and $h_i^{(0)}$ should be passed down. We employ a multi-head attention (Vaswani et al., 2017) to capture the relation between syntax and semantics. For the i-th word, we project $\mathbf{H}_i^{local} = \{h_i^{(L)}, h_i^{(0)}\}$ into the distinct key, value, and query representations, denoted K_j, Q_j and V_j for each head j. Then we perform the scaled dot product attention as Attention $(Q, K, V) = \operatorname{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$.

Then the outputs of all heads are concatenated and passed through a feed-forward layer followed by GeLU activations (Hendrycks and Gimpel, 2017) and a layer normalization. And we perform average pooling on the final outputs of local integration for each word, denoted as $\mathbf{H}' = \{h'_1, \ldots, h'_n\}$, where the syntactic representation and original contextual word representation are fused. Note that all the parameters of the local integration layer are shared for all the words. In Fig 2(a), we only show three network blocks for clarity.

Global Matching After local integration captures the relationship between syntax and semantics of each word, we propose a global matching layer to model connections between fused representations at the overall utterance-level. Similar to the local integration, we use another multi-head attention layer where we project $\mathbf{H}^{global} = \mathbf{H}' = \{h'_1, \ldots, h'_n\}$ into the the distinct key, value, and query representations. The syntactic information from GCN will guide a word to other words of syntactic importance in a sentence. To reduce the overall parameters and computational cost, we do not employ a stack of multiple identical blocks but only one multi-head attention layer for both local integration and global matching. The outputs of global matching will be forwarded into the final output layer.

2.4 Joint Optimization

Finally, we use the output hidden vectors of global matching to predict the slot types and intent. For slot types, we directly perform softmax operation over the hidden vector at each timestep. For intent prediction, we perform max pooling over all the words to obtain a fixed-length vector. Then the final vector is fed to the multi-layer perceptron (MLP) classifier, with one hidden layer, tanh activation, and softmax output layer. The entire model is trained by minimizing the sum of two cross-entropy losses in an end-to-end manner. The overall objective is formulated as $p\left(y^S, y^I|\mathbf{X}\right) = p\left(y^I|\mathbf{X}\right) \prod_{t=1}^{n} p\left(y_t^S|\mathbf{X}\right)$, where y^S, y^I are the softmax output probability of slots and intent respectively.

3 Experiments

3.1 Settings

To evaluate the proposed model, we conduct experiments on two public benchmark datasets, ATIS (Hemphill et al., 1990) and Snips (Coucke et al., 2018). ATIS contains audio recordings of flight reservations, and Snips is collected from the Snips personal voice assistant. For the syntactic parser, we adopt the Stanford parser from (Chen and Manning, 2014). The parser is not updated with our SLU model. For the SLU model, we use the glove embedding with the dimension of 300 and set the dropout rate as 0.1. L2 regularization is used with a rate of 1×10^{-3}. We use the Adam optimizer (Kingma and Ba, 2014) with the learning rate of 0.001. For the GCN model, we set the propagation layer num as 2 and treat the dependency tree as an undirected graph. All the results are reported on the test set after early stopping on the dev set.

3.2 Baselines

We compare our model with the existing baselines including: Joint Seq(Hakkani-Tür et al., 2016) proposes an RNN-based multi-task modeling approach for jointly modeling domain detection, intent detection, and slot filling. Attention BiRNN(Liu and Lane, 2016) leverages the attention mechanism to learn the relationship between slot and intent. Slot-Gated Atten(Goo et al., 2018) proposes the slot-gate to model the correlation of slot filling and intent detection. Self-Attentive Model(Li et al., 2018) proposes 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) proposes the Bi-model to consider the intent and slot filling cross-impact to each other. CAPSULE-NLU(Zhang et al., 2018a) proposes a capsule-based

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., 2018a)	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
Stack-Propagation (Qin et al., 2019)	94.2	98.0	86.9	**95.9**	96.9	86.5
Our model	**94.8***	**98.2***	**87.6***	95.7	**97.2***	**86.9***

Table 1: 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.

model with a dynamic routing-by-agreement schema to accomplish slot filling and intent detection. SF-ID Network(E et al., 2019) introduces an SF-ID network to establish multiple direct connections for the slot filling and intent detection to help them promote each other mutually. Stack-Propagation(Qin et al., 2019) proposes a Stack-Propagation framework that can directly use the intent information as input for slot filling, thus to capture the intent semantic knowledge. We report the experiment results of these models adopted from (Qin et al., 2019).

3.3 Overall Results

We evaluate the SLU performance about slot filling using F1 score, intent prediction using accuracy, and sentence-level semantic frame parsing using overall frame accuracy. We take the overall accuracy as the main evaluation metric since the metric considers the joint performance of both slot filling task and intent detection task. Table 1 displays the performance of our proposed model and baseline models on ATIS and Snips dataset. As shown in Table 1, we observe that our model outperforms all the baselines obviously on Overall (Acc). Specially, compared with the best prior joint work Stack-Propagation, we achieve 0.7% improvement on Overall (Acc) in the Snips dataset. Meanwhile, we achieve 0.4% improvement on Overall (Acc) in the ATIS dataset. In Snips dataset, our model outperforms the baseline models on all the evaluation metric. Although our model achieve 0.2% lower performance than Stack-Propagation on Slot (F1) in ATIS dataset, our model still outperforms Stack-Propagation on main evaluation metric, i.e. Overall (Acc), obviously. Above results indicate that our proposed model can improve the SLU performance significantly by integrating the syntactic structure with contextual information.

4 Qualitative Analysis

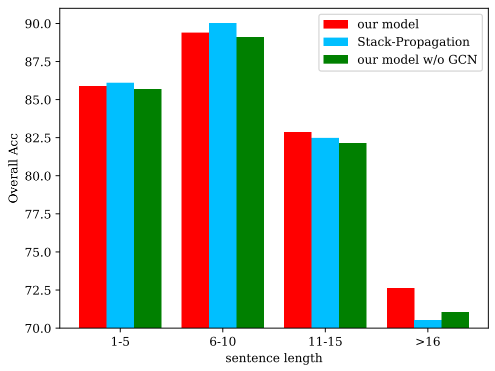
In this section, we present a detailed qualitative analysis on each component of our proposed model and provide certain typical cases to show the effectiveness of incorporating syntactic information. We first perform an ablation study to validate the effect of different modules of our model. Then we explore more methods of feature aggregation and demonstrate the superiority of our proposed local-to-global multi-head attention mechanism. Next, we give some typical cases of our model and baseline to show the effect of syntactic structure. Finally, we conduct experiment with BERT to verify that our model is more effective with pre-trained model.

4.1 Effect of Syntactic GCN

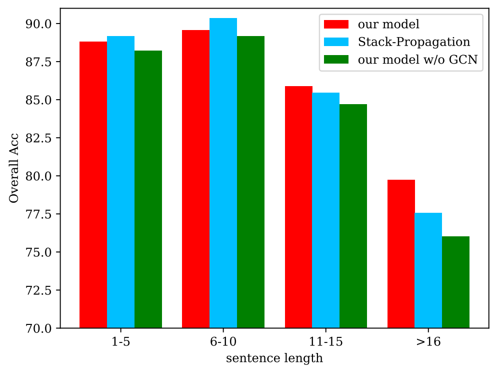
To verify the effectiveness of syntactic information delivered by the dependency tree, we conduct comparison experiments with the same architecture except for the way of constructing the adjacency matrix $\mathbf{A}$. As Table 2.1 shows, we experiment with two distinct adjacency matrices of all 0s and all 1s. The *Adj=0* model which fills the adjacency matrix with all 0s aims to disentangle GCN from our proposed model as a baseline. And the *Adj=1* model which fills the adjacency matrix with all 1s demonstrates the effect of the dependency tree.

Model	SNIPS			ATIS		
	Slot (F1)	Intent (Acc)	Overall (Acc)	Slot (F1)	Intent (Acc)	Overall (Acc)
Adj=0*	93.3	97.7	84.6	94.9	96.5	84.9
Adj=1**	92.9	97.7	83.5	90.8	95.5	79.8
Syntax GCN	**94.8**	**98.2**	**87.6**	**95.7**	**97.2**	**86.9**

Table 2: Effect of Syntactic GCN. * indicates that the *Adj=0* model fills the adjacency matrix with all 0s. By contrast, ** indicates that the *Adj=1* model fills the adjacency matrix with all 1s. *Syntax GCN* represents our proposed syntactic GCN model where the adjacency matrix is filled with the dependency tree as described in Section 2.1.



(a) Split by sentence length in the ATIS test dataset



(b) Split by sentence length in the Snips test dataset

Figure 3: Test set performance with regard to sentence length for our proposed model, Stack-Propagation (Qin et al., 2019) and our model w/o GCN which fills the adjacency matrix with all 0s.

In the Snips dataset, compared to the *Adj=0* model, *Adj=1* gets a drop of 1.1% on Overall (Acc) while our model get 3.0% improvements. The similar results are also shown in the ATIS dataset. We hypothesize that this is because filling the adjacency matrix with all 1s essentially builds a fully-connected graph, which induces tremendous noise to the model. By contrast, integrating the syntactic information makes a word focus more on its dependency words as constraints. The experiment results confirm that incorporating syntactic dependency benefits the understanding of natural language.

4.2 Effect of Sentence Length

To understand what our syntax GCN model captures and how it differs from the previous baselines such as Stack-Propagation, we compare their performance over examples with different ranges of sentence length in the Fig 3. Specifically, for each model, we train it on the same training set and report their overall accuracy on examples with different sentence lengths of the test set.

Fig 3 shows that our proposed syntax GCN model outperforms Stack-Propagation with notable improvements at handling long sentences. We believe our model can better resolve issues of long-term dependencies via the explicit syntactic information. Besides, compared to the model w/o GCN, our model consistently achieves superior performance, which demonstrates the effectiveness of the feature aggregation layer.

4.3 Effect of Feature Aggregation

We further explore the benefits of our feature aggregation mechanism in our model. We conduct comparison experiments with the same architecture except for the feature aggregation layer. Table 3 shows the overall results of different feature aggregation methods, including Add, Concat, Gate, Full and our local-to-global multi-head attention mechanism. Given the syntactic representation $h_i^{(L)}$ and contextual representation $h_i^{(0)}$ of the i-th word, we define the Gate as $o_i = \alpha * h_i^{(L)} + (1-\alpha) * h_i^{(0)}$where $\alpha = W_1 h_i^{(L)} + W_2 h_i^{(0)}$, and the Full as $o_i = Concat([h_i^{(L)}; h_i^{(0)}; |h_i^{(L)} - h_i^{(0)}|; h_i^{(L)} * h_i^{(0)}])$.

Compared to the base RNN, all the aggregation methods, Add, Concat, Gate, Full, achieve $1\% \sim 2\%$

Model	SNIPS			ATIS		
	Slot (F1)	Intent (Acc)	Overall (Acc)	Slot (F1)	Intent (Acc)	Overall (Acc)
RNN	90.7	96.9	80.7	94.3	95.3	82.5
Add	91.7	97.4	82.3	94.8	95.6	84.4
Concat	91.5	98.1	82.1	94.9	94.4	83.5
Gate	92.0	97.7	82.6	94.9	95.3	83.9
Full	91.5	97.9	81.6	94.9	94.2	83.5
Local Integration	93.4	97.9	85.9	95.2	96.1	85.1
Global Matching	94.1	98.0	86.7	95.3	97.1	86.2
Our model	**94.8**	**98.2**	**87.6**	**95.7**	**97.2**	**86.9**

Table 3: Effect of Feature Aggregation.

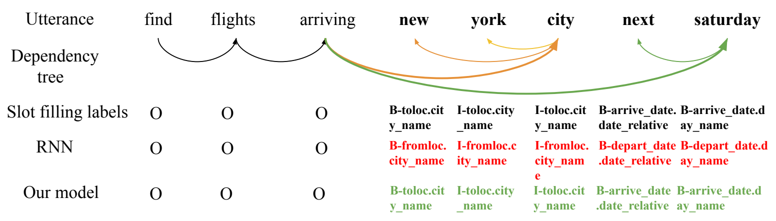


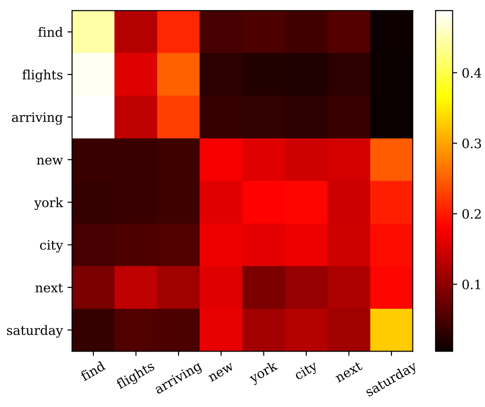
Figure 4: Case study of RNN and our model. The [GREEN] ([RED]) highlight indicates a correct (incorrect) tag.

improvements in both datasets, which demonstrates the effectiveness of incorporating syntactic structure via GCN. Further, our proposed local-to-global aggregation layer outperforms these methods with a statistically significant margin. The results confirm feature aggregation mechanism plays a vital role in the integration of contextual representation and syntactic information.

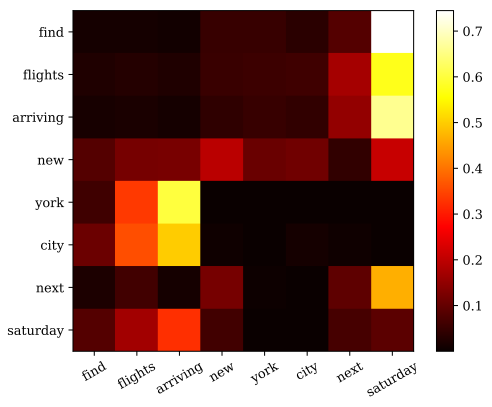
4.4 Case Study

We display two samples from basic RNN and our model in Fig 4. Given the same input *"find flights arriving new york city next saturday"*, RNN can not distinguish the *from_loc* from *to_loc* because it can not explicitly model the relationships between *new york city* and *arriving*. By contrast, our model leverages the syntactic structure to make *new york city* focus more on its dependency word *arriving*. This example illustrates the syntactic structure could enhance the contextual representation to facilitate the SLU tasks by modeling direct relations between words.

4.5 Visualization Analysis



(a) The model without syntactic GCN



(b) Our proposed model

Figure 5: Visualization of attention distributions of the self-attention layer of global matching in our syntactic GCN model(right) and the variant without GCN(left).

To have a quick grasp of how syntactic information works, we perform visualization analysis of attention distributions of the self-attention layer of global matching in our syntactic GCN model and the variant without GCN, as shown in Fig 5. Weights of attention are selected from the first head of the self-attention layer. After integrating syntactic knowledge, the word ”city” focuses more on its dependency word ”arriving”, which convincingly indicates that ”new york city” is an entity of arrival city but departure city. The visualization confirms that syntactic knowledge makes a word attentively select the relevant words and enhance contextual representations to distinguish subtle differences. Compared to (Zhang et al., 2019) which restrains the scope of attention only between word and all of its ancestor head words, we incorporate the syntactic dependency tree as a soft mask. We believe this soft mask can alleviate errors caused by incomplete dependency parser.

4.6 Ablation Study

Model	**SNIPS**			**ATIS**		
	Slot (F1)	Intent (Acc)	Overall (Acc)	Slot (F1)	Intent (Acc)	Overall (Acc)
RNN	90.7	96.9	80.7	94.3	95.3	82.5
GCN*	83.6	97.4	66.8	81.9	95.1	54.5
RNN+GCN**	91.7	97.4	82.3	94.8	95.6	84.4
RNN+GCN+Local Integration	93.4	97.9	85.9	95.2	96.1	85.1
RNN+GCN+Global Matching	94.1	98.0	86.7	95.3	97.1	86.2
Our model	**94.8**	**98.2**	**87.6**	**95.7**	**97.2**	**86.9**

Table 4: Performance of different model variants. * indicates that the *GCN* model initializes the first GCN layer inputs $h^{(0)}$ with word embeddings. ** indicates that the *RNN+GCN* model simply sums contextual embeddings and GCN outputs.

To study the effect of each component of our method, we conduct ablation analysis (Table 4). In the ATIS and Snips dataset, the basic RNN model achieves 82.5 and 80.7 on overall accuracy respectively, which are much higher performance than vanilla GCN, 54.6 and 66.8. These results indicate that SLU tasks need contextual word representation, especially for slot filling task, while the syntactic structure could enhance the RNN model as supplementary knowledge. We can see that the simple RNN+GCN achieves 1.9% improvements in the ATIS dataset and 1.6% improvements in the Snips dataset.

On the other hand, although simply inducing syntactic structure(RNN+GCN) helps improve the basic RNN model, both Local Integration and Global Matching further improve the whole performance. We can see that Local Integration and Global Matching achieve 0.7% and 1.8% improvements in the ATIS dataset, 2.4% and 4.4% improvements in the Snips dataset, compared to the RNN+GCN. The full local-to-global aggregation further achieves 2.5% and 5.3% improvements respectively. The results demonstrate the effectiveness of the feature aggregation mechanism since the syntactic representation and contextual word representation can complement each other. Hence, our proposed local-to-global multi-head attention achieves the best performance.

4.7 Effect of BERT

Model	**SNIPS**			**ATIS**		
	Slot (F1)	Intent (Acc)	Overall (Acc)	Slot (F1)	Intent (Acc)	Overall (Acc)
Our model	94.8	98.2	87.6	95.7	97.2	86.9
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
Stack-Propagation + BERT (Qin et al., 2019)	97.0	**99.0**	92.9	96.1	97.5	88.6
Our model + BERT	**97.1**	**99.0**	**93.0**	**96.2**	**97.8**	**88.7**

Table 5: The SLU performance on BERT-based model on two datasets.

Considering the performance with the fine-tuning approach, we also conduct experiments that we replace the contextualized BiLSTM by the BERT (Devlin et al., 2019) in our framework and keep the same architecture in rest of our model. The results of BERT model on ATIS and SNIPS datasets are shown in Table 5. From the Table 5, we can observe our model utilizing BERT achieves a new state-of-the-art performance. These results indicate a strong pre-trained model can further improve the performance for our model on SLU tasks. Our model + BERT outperforms Stack-Propagation + BERT which indicate that our framework is more effective with BERT than baseline models.

5 Related work

Slot filling and intent detection are two major tasks for SLU. Recently, the typical pipeline methods build slot filling and intent detection separately, where slot filling is implemented as a standard sequence labeling task (Yao et al., 2014) and intent detection is built as a classification task (Lai et al., 2015), respectively. More recent work (Hakkani-Tür et al., 2016; Liu and Lane, 2016; Goo et al., 2018; Li et al., 2018; Wang et al., 2018; Zhang et al., 2018a; E et al., 2019; Qin et al., 2019) implement slot filling and intent detection as a joint model to eliminate the error propagation without any linguistic knowledge. Instead, our work apply the linguistic knowledge (i.e. dependency tree) as a prior to guide the learning slot filling and intent detection jointly.

Dependency tree, as a important linguistic knowledge, is applied to recent natural language processing tasks. In relation extraction and machine translation, many studies (Xu et al., 2015; Liu et al., 2015; Miwa and Bansal, 2016; Chen et al., 2018) have show that the dependency trees can capture long-distance relations effectively. In machine reading comprehension, (Zhang et al., 2019) use syntax to guide the text modeling by incorporating explicit syntactic constraints into attention mechanism for better linguistically motivated word representations and achieve promising results on both SQuAD 2.0 and RACE datasets. Inspired by (Zhang et al., 2019), our work utilize the dependency tree to guide the joint model for slot filling and intent detection. Different from (Zhang et al., 2019), our work apply the representation over dependency tree as a inner feature instead of syntactic constraints into attention mechanism.

Graph Convolution Network (GCN) also have been utilized for many natural language processing tasks. (Vashishth et al., 2019) propose a flexible graph convolution based method for learning word embeddings. (Marcheggiani and Titov, 2017) apply a GCN as sentence encoders to produce latent feature representations of words in a sentence for semantic role labeling task. (Bastings et al., 2017) present a simple and effective approach to incorporate syntactic structure by the way of GCN into the encoder-decoder model for machine translation. (Yao et al., 2019) build a single text graph for a corpus based on word co-occurrence and document word relations, then learn a text GCN for the corpus to classify the text. Different from above work, our work propose a model that applies a GCN over dependency trees to integrate the syntactic structure for joint learning slot filling and intent detection.

6 Conclusion

In this paper, we propose a novel joint model that applies a graph convolution network over dependency trees to integrate the syntactic structure for learning slot filling and intent detection jointly. In addition, we utilize multi-head attention to fuse syntactic representation with contextual word representation to access complementary representation for SLU task. Experimental results show that our proposed model outperforms strong baseline models and achieves state-of-the-art performance on both ATIS and Snips datasets in overall accuracy metric. Finally, we apply the BERT model to our framework and experiments demonstrate that our proposed model integrating BERT model can improve the performance on both slot filling and intent detection more obviously.

Acknowledgments

The work was done when the first author was an intern at Meituan Dialogue Group. We thank Xiaojie Wang, Jiangnan Xia and Hengtong Lu for the discussion. We thank all anonymous reviewers for their constructive feedback.

References

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *EMNLP*, pages 1957–1967.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*.

Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2018. Syntax-directed attention for neural machine translation. In *AAAI*.

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. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Haihong E, Peiqing Niu, Zhongfu Chen, and Meina Song. 2019. A novel bi-directional interrelated model for joint intent detection and slot filling. In *ACL*, pages 5467–5471.

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 *NAACL*, pages 753–757.

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*, pages 715–719.

Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language*.

Dan Hendrycks and Kevin Gimpel. 2017. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *ArXiv*, abs/1606.08415.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

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

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*.

Changliang Li, Liang Li, and Ji Qi. 2018. A self-attentive model with gate mechanism for spoken language understanding. In *EMNLP*, pages 3824–3833.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *Interspeech 2016*, pages 685–689.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and WANG Houfeng. 2015. A dependency-based neural network for relation classification. In *ACL-IJCNLP*, pages 285–290.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *EMNLP*, pages 1506–1515.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *ACL*, pages 1105–1116.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Libo Qin, Wanxiang Che, Yangming Li, Haoyang Wen, and Ting Liu. 2019. A stack-propagation framework with token-level intent detection for spoken language understanding. In *EMNLP-IJCNLP*, pages 2078–2087.

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

Shikhar Vashishth, Manik Bhandari, Prateek Yadav, Piyush Rai, Chiranjib Bhattacharyya, and Partha Talukdar. 2019. Incorporating syntactic and semantic information in word embeddings using graph convolutional networks. In *ACL*, pages 3308–3318.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz 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 *NAACL*, pages 309–314.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *EMNLP*, pages 1785–1794.

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*, pages 189–194. IEEE.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *AAAI*, volume 33, pages 7370–7377.

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip S Yu. 2018a. Joint slot filling and intent detection via capsule neural networks. *arXiv preprint arXiv:1812.09471*.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018b. Graph convolution over pruned dependency trees improves relation extraction. In *EMNLP*.

Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, and Hai Zhao. 2019. Sg-net: Syntax-guided machine reading comprehension. *arXiv preprint arXiv:1908.05147*.