A Graph-to-Sequence Model for Joint Intent Detection and Slot Filling in Task-Oriented Dialogue Systems

Anonymous ACL submission

Abstract

Effectively decoding semantic frames in task-oriented dialogue systems remains a challenge, which typically includes intent detection and slot filling. Although RNN-based neural models show promising results by jointly learning of these two tasks, dominant RNNs are primarily focusing on modeling sequential dependencies. Rich graph structure information hidden in the dialogue context is seldomly explored. In this paper, we propose a novel Graph-to-Sequence model to tackle the spoken language understanding problem by modeling both temporal dependencies and structural information in a conversation. We introduce a new Graph Convolutional LSTM (GC-LSTM) encoder to learn the semantics contained in the dialogue dependency graph by incorporating a powerful graph convolutional operator. Our proposed GC-LSTM can not only capture the spatio-temporal semantic features in a dialogue, but also learn the co-occurrence relationship between intent detection and slot filling. Furthermore, a LSTM decoder is utilized to perform final decoding of both slot filling and intent detection, which mutually improves both tasks through global optimization. Experiments on benchmark ATIS and Snips datasets show that our model achieves state-of-the-art performance and outperforms existing models.

1 Introduction

Spoken language understanding (SLU) in task-oriented dialogue systems, including intent detection and slot filling (Tur and Mori, 2011), has been greatly advanced by deep learning techniques. It aims to parse users's utterances into semantic frames in order to capture a conversation's core meaning. The input of SLU is a sequence of words, whereas the output is a sequence of predefined slot types represented in In-Out-Begin (IOB) format. A specific intent label is also assigned for the whole sentence. For example, in Table 1, given an utterance "Flights from Charlotte to Miami", SLU is supposed to determine the users' intention as *Flight* and to map each word into predefined slots.

Sentence	Flights	from	Charlotte	to	Miami
Intent	Flight				
Slots	O	O	B-fromloc	O	B-toloc

Table 1: An example utterance annotated with its intent and semantic slots (IOB format).

Early studies modeled intent detection and slot filling separately in a pipelined manner, and were insufficient to take full advantage of all supervised signals, as they intrinsically shared semantic knowledge. What's more, in the pipelined architecture, errors made in upper stream modules may propagate and be amplified in downstream components, which, however, could possibly be eased in joint models (Zhang and Wang, 2016). Thus, jointly modeling intent detection and slot filling has attracted significant attention, and achieved promising results with recurrent neural networks (RNNs) (Liu and Lane, 2016; Goo et al., 2018; Li et al., 2018). However, these RNN-based models are primarily focusing on modeling sequential dependencies, and inherently unstable over long-time sequences as RNNs tend to focus more on short-term memories (Weston et al., 2014). Indeed, this weakness of sequential RNN-based models leads to a large portion of slot filling errors (Tur et al., 2010).

Subsequently, Zhang et al. (2020) attempted to address the limitation of sequential models by utilizing S-LSTM to learn the graph structure in dialogue utterances, and achieved promising improvement compared with sequential RNNs. Nevertheless, this model still suffers from three major issues: **1) Modeling dialogue graphs**. Although the n-gram context graph used in S-LSTM has to some extent captured the influence of neighboring words within a specific window, closely-related words, such as "parents", "children" and "siblings" in a dialogue graph can be outside this window and unfortunately neglected. Actually, these words should have substantial impact on slot tag decoding. Furthermore, unrelated words within the n-gram window are acting as noise, leading to more slot filling errors. **2) Learning spatial structures in dialogues**. S-LSTM is incapable of capturing spatial structures in a conversational context, and we observe that this property plays a vital role in modeling a fully graph-structured dialogue. **3) Jointly decoding intent detection and slot filling in a stand-alone decoder**. Zhang et al. (2020) utilized a S-LSTM to both encode dialogue states and decode final intents and slot tags. This puts too much burden on the S-LSTM and deteriorates SLU performance.

In this paper, we propose a novel Graph-to-Sequence (Graph-to-Seq) framework to perform joint intent detection and slot filling in task-oriented dialogue systems. The proposed model learns spatio-temporal semantic features hidden in dialogues by modeling dialogue structure as a dependency graph, and by employing a Graph Convolutional LSTM (GC-LSTM). Graph Convolutional operation further enables a deeper level of semantic modeling of the dialogue context. A separate SLU decoder is also used to jointly decode slot tags and intents in a globally optimal way.

In short, our contributions are threefold:

- To the best of our knowledge, our work is the first one that introduces spectral graph convolution to model the graph structures in SLU.

- We propose a novel GC-LSTM to effectively learn graph-structured representations in dialogues. We model a dialogue graph as an enhanced dependency tree by adding forward and backward edges between words in order to capture both sequential and structural information in dialogues.

- We introduce a novel Graph-to-Seq framework to perform joint SLU. A stand-alone RNN decoder is greatly beneficial to improve SLU performance by relieving encoders from decoding burden, and by enabling the interaction between intent detection and slot filling.

2 Related Work

Joint Intent Detection and Slot Filling Zhang and Wang (2016) proposed a joint work using RNNs for learning the correlation between intents and slots. Hakkani-Tür et al. (2016) adopted a RNN for slot filling and the last hidden state of the RNN was used to predict the utterance intent. Liu and Lane (2016) introduced an attention-based RNN encoder-decoder model to jointly perform intent detection and slot filling.

Most recently, some work modeled the intent information for slot filling explicitly in joint models. Goo et al. (2018); Li et al. (2018) proposed a gating mechanism to explore incorporating the intent information for slot filling. However, as the sequence becomes longer, it is risky to simply rely on the gate function to sequentially summarize and compress all slots and context information in a single vector (Cheng et al., 2016). Zhang et al. (2018a) proposed a hierarchical capsule neural network to model the hierarchical relationship among words, slots, and intents in an utterance. Niu et al. (2019) introduced a SF-ID network to establish the interrelated mechanism for slot filling and intent detection tasks. However, these RNN-based models suffer from being weak at capturing long-range dependencies. Then Wu et al. (2021) explicitly modeled the long-term slot context via a key-value memory network beneficial to both slot filling and intent detection. Unfortunately, None of these models did take the rich structure information in dialogues into consideration. Subsequently, Zhang et al. (2020) attempted to address the limitation of sequential models by utilizing S-LSTM with a context-gated mechanism to learn the local context in dialogue utterances, and achieved promising improvement compared with sequential RNN models. Compared with this work, our work differs significantly by proposing a novel Graph-to-Seq framework with Graph Convolution LSTM on enhanced dialogue dependency graphs.

Graph Convolutional LSTM Graph convolutional networks (GCNs) generalize convolutional neural networks to graphs. Spectral GCNs transform graph signals on graph spectral domains and then apply spectral filtering on graph signals. Defferrard et al. (2016) proposed a spectral formulation for the convolutional operator on graph with fast localized convolutions. Kipf and Welling (2017) introduced Spectral GCNs for semi-supervised classification on graph-structured data. Subsequently, in order to incorporate temporal features, Seo et al. (2016) proposed a graph convolutional LSTM to capture the spatial-temporal features of graph structures. This was an extension of GCNs to have a recurrent architecture. Si et al. (2019) further improved graph convolutional LSTM network by introducing attention to effectively extract discriminative spatial and temporal features in Skeleton-Based Action Recognition.

3 Proposed Model

Given a sequence of words $w = (w_1, w_2, ..., w_n)$ in an utterance and an associated dialogue dependency relation graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, intent detection can be defined as a classification task that outputs an expected intent label o^I, where $\mathcal{V}$ and $\mathcal{E}$ denote the set of word nodes and relations in $\mathcal{G}$, and n is the utterance length. Slot filling can be seen as a sequence labeling task that maps the input utterance w into a predefined slot sequence $o^S = (o^S_1, o^S_2 ..., o^S_n)$.

3.1 Model Overview

We propose a novel Graph-to-Seq framework to combine the merits of S-LSTM and GCNs to effectively learn the spatio-temporal representation of the dialogue context. Our proposed model is composed of two major components: a GC-LSTM encoder, and a SLU decoder, as shown in Figure 1. The GC-LSTM encoder learns fixed-length vectors to represent the dialogue context structurally. Not only can it model spatial graph structure information, but also it learns the semantic correlation between slots and intents. Message passing in our graph convolutional operation improves on capturing the long-range dependencies. We further add a context gate to improve our model's ability to utilize local context information. In our decoder, a dedicated LSTM is employed to integrate hidden states of the GC-LSTM for generating slot tagging and final intents. Our decoder first decodes slot tags and then outputs an expected intent. We intentionally choose this mechanism to improve intent accuracy since the slot information is beneficial to intent detection. Both intent detection and slot filling are optimized simultaneously via joint learning. In the following sections we detail each component thoroughly.

3.2 Spectral Graph Convolutions

Graph convolutional neural networks are an effective framework for learning the representation of graph structured data. As it is difficult to express a meaningful translation operator in the vertex domain, Defferrard et al. (2016) proposed a spectral formulation for the convolutional operator on graph $*_\mathcal{G}$. Based on this definition, a spectral convolution on graphs is defined as the multiplication of a graph signal $x \in \mathbb{R}^N$ (a scalar for every node) with a non-parametric kernel filter $g_\theta = \mathrm{diag}(\theta)$ parameterized by $\theta \in \mathbb{R}^N$ in Fourier domain, where N is the number of vertices, as follows:

$$g_\theta *_\mathcal{G} x = U g_\theta U^T x \tag{1}$$

where $U \in \mathbb{R}^{N \times N}$ is the matrix of eigenvectors of the normalized graph Laplacian $L = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^T$ with a diagonal matrix of its eigenvalues Λ and $U^T x$ being the graph Fourier transform of x. g_θ can be thought as a function of eigenvalues of L, i.e. $g_\theta(\Lambda)$. However, evaluating Equation (1) is computationally expensive as the multiplication with U is $\mathcal{O}(N^2)$. What's more, calculating the eigendecomposition of L might be prohibitively expensive for large graphs. Thus, Defferrard et al. (2016) parameterized g_θ as a truncated expansion, up to $(K-1)^{th}$ order of Chebyshev polynomials T_k such that:

$$g_\theta(\Lambda) \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}) \tag{2}$$

where the parameter $\theta \in \mathbb{R}^K$ is the vector of Chebyshev coefficients and $T_k(\tilde{\Lambda})$ is the Chebyshev polynomial of order k evaluated at $\tilde{\Lambda} = 2\Lambda/\lambda_{max} - I_N$. λ_{max} denotes the largest eigenvalue of L. The graph filtering operation can then be written as

$$g_\theta *_\mathcal{G} x \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L}) x \tag{3}$$

where $\tilde{L} = 2L/\lambda_{max} - I_N$. Equation (3) can be evaluated by using the stable recurrent relation $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0 = 1$ and $T_1 = x$ in $\mathcal{O}(K|\mathcal{E}|)$ operations, where $\mathcal{E}$ is the number of edges. As pointed out by Defferrard et al. (2016), when the filtering operation Equation (3) is an order K polynomial of the Laplacian, it is K-localized and depends only on nodes that are maximum K hops away from the central node, that is, the K-neighborhood.

3.3 Graph Convolutional LSTM Encoder

Based on the graph convolutional operation defined in Equation (3), we propose a GC-LSTM encoder

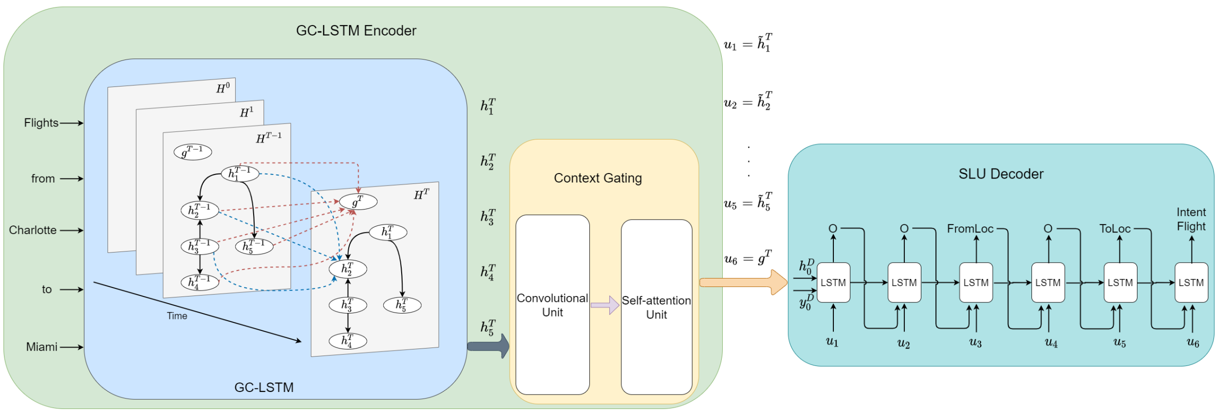

Figure 1: Graph-to-Sequence model for joint intent detection and slot filling.

to encode graph structures in dialogue utterances. Utterance words are firstly transformed to word embeddings $e = (e_1, e_2, ..., e_n)$ by using the pre-trained ELMo embeddings (Peters et al., 2018). They are then fed into the GC-LSTM at each time step. After T steps, our GC-LSTM generates word-level hidden states for decoding slot labels, and sentence-level hidden states for predicting intents.

Dialogue Graph We model word relationships by using dependency trees, as dependency links are close to the semantic relationships for the next stage of interpretation. In order to enable learning various relationships of words such as dependency relations, we first use the off-the-shelf parsing tool called Spacy[1] to extract dependency relation graph $\mathcal{G}$ among words in dialog utterances as shown in Figure 2. To further support bi-directional information flow, we explicitly add reverse edges and sequential relations (i.e., Next and Prev) as well. Enhanced dependency graphs allow information flow between dependent words and head words bidirectionally, enabling the learning process to capture the rich semantic representation between them.

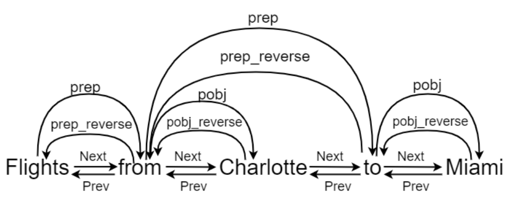

Figure 2: An example of our dialogue utterance graph

Graph Convolutional LSTM We introduce a new GC-LSTM by extending S-LSTM (Zhang et al., 2018b) to include a powerful graph convolutional operator in order to better learn graph-structured semantics in a dialogue. GC-LSTM views the whole sentence as a single graph $\mathcal{G}$, consisting of word-level nodes h_i and a sentence-level node g. At each time step t, the graph state is represented as: $H^t = (h_1^t, h_2^t, ..., h_n^t, g_t)$.

Like S-LSTM, GC-LSTM uses a recurrent state transition process to model information between sub states, which enriches state representations incrementally. Unlike S-LSTM, GC-LSTM updates its word hidden states using the graph convolution operation in order to capture spatial features of the semantics. The graph state transition from H^{t-1} to H^t consists of word-level node state change from h^{t-1} to h^t, and sentence-level state transition from g^{t-1} to g^t. We set initial state $h_i^0 = g^0 = h^0$ in H^0, where h^0 is a parameter. The hidden state h_i^t is a function of word embedding e_i, its neighboring node hidden state h_j^{t-1}, and sentence-level state g^{t-1}, where $j \in \mathcal{N}(i)$ and $\mathcal{N}(i)$ is the neighbor nodes of word node i. This updating function is formulated as follow:

$$
\begin{aligned}
\hat{i}^t &= \delta(W_i *_{\mathcal{G}} h^{t-1} + U_i e^t + V_i g^{t-1} + b_i) \\
\hat{f}^t &= \delta(W_f *_{\mathcal{G}} h^{t-1} + U_f e^t + V_f g^{t-1} + b_f) \\
\hat{s}^t &= \delta(W_s *_{\mathcal{G}} h^{t-1} + U_s e^t + V_s g^{t-1} + b_s) \\
o^t &= \delta(W_o *_{\mathcal{G}} h^{t-1} + U_o e^t + V_o g^{t-1} + b_o) \\
u^t &= \delta(W_u *_{\mathcal{G}} h^{t-1} + U_u e^t + V_u g^{t-1} + b_u) \\
i^t, f^t, s^t &= \mathrm{softmax}(\hat{i}^t, \hat{f}^t, \hat{s}^t) \\
c^t &= f^t \odot c^{t-1} + s^t \odot c_g^{t-1} + i^t \odot u^t \\
h^t &= o^t \odot \tanh(c^t)
\end{aligned}
\tag{4}
$$

where W_x, U_x, V_x and b_x are model parameters ($x \in \{i, f, s, o, u\}$), δ is the sigmoid function, and $\odot$ denotes Hadamard product.

[1] https://spacy.io/

Different than S-LSTM, GC-LSTM only contains three gates: an input gate i^t, a forget gate f^t, and an output gate o^t. But similar to S-LSTM, GC-LSTM also has one sentence gate s_t controlling information from sentence cell c_g^{t-1}. These control gates are normalized by a softmax function and then served as probability weights to regulate new cell states c_t and hidden states h^t.

Following Zhang et al. (2018b), the sentence-level node g^t is updated based on all word-level nodes:

$$
\begin{aligned}
&\bar{h} = \text{avg}(h_1^{t-1}, h_2^{t-1}, ..., h_n^{t-1}) \\
&\tilde{f}_g^t = \delta(W_g g^{t-1} + U_g \bar{h} + b_g) \\
&\tilde{f}_i^t = \delta(W_f g^{t-1} + U_f h_i^{t-1} + b_{\tilde{f}}) \\
&\tilde{o}_g^t = \delta(W_{og} g^{t-1} + U_{og} \bar{h} + b_{og}) \\
&f'^t_1, ..., f'^t_n, f'^t_g = \text{softmax}(\tilde{f}_1^t, ..., \tilde{f}_n^t, \tilde{f}_g^t) \\
&c_g^t = f'^t_g \odot c_g^{t-1} + \sum_{i=1}^{n} f'^t_i \odot c_i^{t-1} \\
&g^t = o_g^t \odot \tanh(c_g^t)
\end{aligned}
\tag{5}
$$

where W_x, U_x and b_x are model parameters ($x \in \{g, f, og\}$). $f'^t_1, ..., f'^t_n$ and f'^t_g are gates controlling information from $c_1^{t-1}, ..., c_n^{t-1}$ and c_g^{t-1}, whereas o_g^t is an output gate regulating the recurrent cell c_g^t to g_t.

At each time step, word-level nodes capture an increasingly larger scope of the dialogue graph, building up knowledge incrementally. On the other hand, the sentence-level node gathers information from all the word-level nodes to refine the whole utterance representation. Slot nodes and the intent node are interacting with each other via Equations (4) and (5), which learns the correlation between intent detection and slot filling. Unlike an LSTM which uses only one state to represent the utterances sequentially, our GC-LSTM uses multiple states (i.e. n word-level states and 1 sentence-level state) to learn deeper context information incrementally with the aid of our graph convolutional operation. In this way, our GC-LSTM can capture long-range dependencies. Finally, after T time steps, word-level hidden states h^T and the sentence-level hidden state g^T are used for predicting slot labels and an expected intent.

Gated Context In order to make our encoder fully context-aware, we introduce a context gate to capture the contextual information for each token like Zhang et al. (2020). The context gate includes a convolution unit and a self-attention unit. The convolution unit captures local context information, such as the internal correlation of phrase structure. Multi-head self-attention (Vaswani et al., 2017) is used to capture diverse global contextual information. The context gating is expressed as follow with details in Zhang et al. (2020):

$$Z = \text{ContextGating}(H^T) \tag{6}$$

Encoder Output The final output of our GC-LSTM encoder with context gating is:

$$
\begin{aligned}
&\tilde{h}_i^T = h_i^T \odot \delta(z_i) \\
&U = (u_1, ...u_{n+1}) = [\tilde{h}_1^T, ..., \tilde{h}_n^T || g^T]
\end{aligned}
\tag{7}
$$

3.4 SLU Decoder

Different than most existing joint models where intent detection and slot filling are decoded separately, our framework decodes them in a shared LSTM. We directly leverage the explicit slot decoding context to help intent detection. By performing a joint SLU decoding in a stand-alone LSTM, there are mainly two advantages:

1. The architecture such as Zhang et al. (2020) puts too much burden on one Graph LSTM encoder as it is playing a dual role in both encoding and decoding. We observe that separating decoding from encoding can be beneficial to overall performance improvement. Dominant seq-to-seq models are primarily relying on an independent autoregressive decoder to generate slot tokens one-by-one conditioned on all previously generated tokens.

2. Sharing slot decoding context with intent detection improves intent detection performance since those two tasks are related. This is substantiated by our following experimental results. With shared decoding states, the interaction between intent detection and slot filling can be further modeled and executed.

We use one unidirectional LSTM as a SLU decoder. At the decoding step $i \in [1, n+1]$, the decoder state h_i^D can be formalized as:

$$h_i^D = \text{LSTM}(h_{i-1}^D, y_{i-1}^D, u_i) \tag{8}$$

where $h_0^D = \tanh(W_0^D u_n)$, h_{i-1}^D is the previous decoder state, y_{i-1}^D is the previous emitted slot label distribution, and W_0^D is the model parameter.

Finally, the slot filling is given by:

$$\begin{aligned} y_i^S &= \text{softmax}(W_h^S h_i^D) \quad i \in [1, n] \\ o_i^S &= \text{argmax}(y_i^S) \end{aligned} \tag{9}$$

The intent detection is defined as follows:

$$\begin{aligned} y^I &= \text{softmax}(W_h^I h_{n+1}^D) \\ o^I &= \text{argmax}(y^I) \end{aligned} \tag{10}$$

3.5 Joint Training

The loss function for intent detection is $\mathcal{L}_1$, and that for slot filling is $\mathcal{L}_2$, which are defined as cross entropy:

$$\mathcal{L}_1 \triangleq -\sum_{i=1}^{n_I} \hat{y}^{I,i} \log\left(y^{I,i}\right) \tag{11}$$

and

$$\mathcal{L}_2 \triangleq -\sum_{j=1}^{n}\sum_{i=1}^{n_S} \hat{y}_j^{S,i} \log\left(y_j^{S,i}\right) \tag{12}$$

where $\hat{y}^{I,i}$ and $\hat{y}_j^{S,i}$ are the gold intent label and gold slot label respectively, and n_I and n_S are the number of intent label types and the number of slot tag types, respectively.

Finally the joint objective is formulated as follows by using hyper-parameters α:

$$\mathcal{L}_\theta = \alpha\mathcal{L}_1 + \mathcal{L}_2 \tag{13}$$

4 Experimental Setup

4.1 Datasets

To evaluate our proposed model, we conduct experiments on two widely used benchmark datasets: ATIS (Airline Travel Information System) and Snips. Both datesets follow the same format and partition in Goo et al. (2018).

ATIS This dataset (Hemphill et al., 1990) contains audio recordings of people making flight reservations. The training set has 4,478 utterances and the test set contains 893 utterances. We use another 500 utterances for the development set. There are 120 slot labels and 21 intent types in the training sets.

Snips Dataset Snips (Coucke et al., 2018) is collected from the Snips personal voice assistant. The training set contains 13,804 utterances and the test set contains 700 utterances. We use another 700 utterances as the development set. There are 72 slot labels and 7 intent types. Compared to the single-domain ATIS dataset, Snips is more complicated mainly due to its large vocabulary and the diversity of intents and slots (Goo et al., 2018).

4.2 Training Details

We implement our model in Pytorch, and trained it on NVIDIA GeForce RTX 2080 Ti. In our experiments, we set the dimension of GC-LSTM hidden state to 200, and that of ELMo embedding to 1024. During training, ELMo parameters are not updated in order to reduce training time. The decoder hidden state dimension is set to 124 for Snips, and to 90 for ATIS. Dropout ratio is set to 0.5 to preventing overfitting, and the batch size is set to 32. The model is trained end-to-end using Adam optimizer (Kingma and Ba, 2014) to minimize the cross-entropy loss, with learning rate $= 1e^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 1e^{-9}$. Finally our graph convolution operation is approximated by $1st$-order Chebyshev polynomials.

4.3 Baselines

We adopt three most popular evaluation metrics in SLU studies: slot filling using F1 score, intent prediction using accuracy, and sentence-level semantic frame parsing using whole frame accuracy.

Baselines models are from some typical works such as Joint Seq. (Hakkani-Tür et al., 2016), Attention BiRNN (Liu and Lane, 2016), Sloted-Gated (Goo et al., 2018), CAPSULE-NLU (Zhang et al., 2019), SF-ID Network (Niu et al., 2019), Key-value Memory (Wu et al., 2021), Unsupervised Transfer + ELMo (Siddhant et al., 2019) and Graph-LSTM (Zhang et al., 2020).

5 Experimental Results

5.1 Automatic Evaluation Results

Table 2 shows the experimental results of our proposed model on ATIS and Snips datasets. On the ATIS dataset, our model substantially outperforms all the baselines by a noticeable margin in all three aspects: slot filling (F1), intent detection (Acc) and sentence accuracy (Acc), demonstrating that explicitly modeling graph-structured dialogue context and the correlation between slots and intents can benefit SLU effectively via GC-LSTM. Compared with the prior joint work Graph-LSTM (Zhang et al., 2020), we achieve F1 score as 96.37% and intent Acc 97.88%, a significant improvement over 95.91% and 97.2%. This performance promotion signifies that our GC-LSTM can effectively model graph-structured dialogue context, and that our Graph-to-Seq framework captures long-term dependencies and models the correlation between slot filling and intent detection.

Model	ATIS Dataset			Snips Dataset		
	Slot(F1)	Intent(Acc)	Sent.(Acc)	Slot(F1)	Intent(Acc)	Sent.(Acc)
Joint Seq.(Hakkani-Tür et al., 2016)	94.30	92.60	80.70	87.30	96.90	73.20
Attention BiRNN(Liu and Lane, 2016)	94.20	91.10	78.90	87.80	96.70	74.10
Sloted-Gated(Goo et al., 2018)	95.42	95.41	83.73	89.27	96.86	76.43
CAPSULE-NLU(Zhang et al., 2019)	95.20	95.0	83.40	91.80	97.30	80.90
SF-ID Network(Niu et al., 2019)	95.58	96.58	86.00	90.46	97.0	78.37
Key-value Memory(Wu et al., 2021)	96.13	97.20	87.12	95.13	98.14	88.14
Unsupervised Transfer + ELMo(Siddhant et al., 2019)	95.62	97.42	87.35	93.90	**99.29**	85.43
Graph-LSTM(Zhang et al., 2020)	95.91	97.20	87.57	95.30	98.29	89.71
Graph-to-Seq	**96.37**	**97.88**	**88.69**	**95.89**	98.43	**90.57**

Table 2: SLU Performance evaluation results on ATIS and Snips datasets (%).

On the Snips dataset, our model also achieves good results in almost all cases, which indicates our model has a better generalization capability than baseline models. Specifically, for slot filling, we achieve a F1 score of 95.89%, a salient enhancement compared with 95.3% (Zhang et al., 2020), and our sentence accuracy reaches at 90.57%. The gain further demonstrates the effectiveness of our proposed Graph-to-Seq framework. Although the intent Acc. of Unsupervised Transfer + ELMo model is slightly higher than ours, this is at the cost of slot filling performance.

Generally, the ATIS dataset is a simpler SLU task than Snips, so the room to be improved is relatively small. However, we still obtain noticeable improvement and set a new state-of-the-art result. On the other hand, Snips dataset is more complex crossing multiple domains. Thus, it is not surprising that most of baseline models are doing poorly especially on slot filling. Surprisingly, our model achieves a great performance jump especially on slot filling. Again we attribute this to our Graph-to-seq framework and GC-LSTM.

5.2 Ablation Study

In this section, we explore how each component contributes to our full model by conducting three important scenarios: (1) **With only GC-LSTM**. In this case, we directly compare the performance between GC-LSTM and S-LSTM (Zhang et al., 2020) to verify the effectiveness of our GC-LSTM. (2) **With GC-LSTM and LSTM decoder but without decoding intent**. This is to verify the effectiveness of a LSTM decoder. (3) **With full Graph-to-Seq framework**.

Table 3 shows the SLU performance variance on these scenarios. First, we only consider GC-LSTM to model spatio-temporal features of dialogue utterances by replacing S-LSTM in Zhang et al. (2020). From Table 3, we can see that GC-LSTM does improve performance in almost all the cases especially on slot filling and shows its superiority over S-LSTM. The result can be interpreted as that GC-LSTM demonstrates great capability to model spatial and temporal dependencies among the dialogue context globally, whereas S-LSTM is more focused on the local context. We then apply a stand-alone LSTM decoder to perform slot decoding. It is noticeable that slot filling is enhanced further, though intent detection deteriorates. It is explainable that using an autonomous autoregressive decoder to generate slot tags token by token not only reduces decoding errors by conditioning on all previously generated tokens, but also alleviates the encoder's burden. However, this model unintentionally puts too much weight on slot filling with the sacrificing of intent detection performance, thus leading to this unbalanced result. Finally, when we jointly perform decoding of slot filling and intent detection, the performance further improves. We attribute this to the fact that sharing slot decoding context not only improves intent detection accuracy, but is also beneficial to slot filling by minimizing intent detection objective function via joint training. To sum up, in a joint SLU model leaning too much on one task potentially worsens the other. Nevertheless, it is salient that our model achieves a trade-off to balance those two tasks.

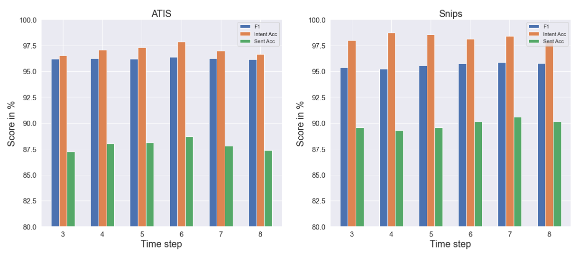

Figure 3: SLU performance on various time steps.

Furthermore, we also study how the parameter time step in GC-LSTM impacts SLU performance.

Model	ATIS Dataset			Snips Dataset		
	Slot(F1)	Intent(Acc)	Sent.(Acc)	Slot(F1)	Intent(Acc)	Sent.(Acc)
Graph-LSTM baseline (Zhang et al., 2020)	95.91	97.20	87.57	95.30	98.29	89.71
With only GC-LSTM	96.21	97.01	87.68	95.40	98.43	89.71
With GC-LSTM and LSTM Decoder no decoding intent	96.30	96.75	87.23	95.68	98.0	89.57
Our full model: Graph-to-Seq	**96.37**	**97.88**	**88.69**	**95.89**	**98.43**	**90.57**

Table 3: Feature ablation study on our proposed model on ATIS and Snips datasets (%).

Figure 3 shows the performance change with different time steps. It is easily observed that as the time steps go up, the sentence-level accuracy increases as well until reaching its peak. This is due to the message passing mechanism trying to enable word-level nodes to involve information spanning the whole dialogue graph. We find that the optimal time step for ATIS and snips datasets is 6 and 7, respectively.

5.3 Dialogue Dependency Graph vs N-gram Context Graph

Model	ATIS Dataset		
	Slot(F1)	Intent(Acc)	Sent.(Acc)
N-gram graph with window 1	96.12	97.09	87.46
N-gram graph with window 2	96.27	96.64	87.57
N-gram graph with window 3	96.28	96.42	87.35
Our dependency graph	**96.37**	**97.88**	**88.69**

Table 4: Performance comparison of dialogue dependency graph and n-gram context graph on ATIS (%).

Model	Snips Dataset		
	Slot(F1)	Intent(Acc)	Sent.(Acc)
N-gram graph with window 1	95.77	98.00	90.29
N-gram graph with window 2	95.61	97.71	89.71
N-gram graph with window 3	95.25	98.0	88.71
Our dependency graph	**95.89**	**98.43**	**90.57**

Table 5: Performance comparison of dialogue dependency graph and n-gram context graph on Snips (%).

We argue that modeling dialogue structural information by using our enhanced dependency graph is superior to the use of the n-gram context graphs. In order to verify this, we design some experiments to only replace our dialogue dependency graph with n-graph context graph with window size 1,2,3, respectively. From Tables 4 and 5, it is noticeable that our dependency graph constantly outperforms the n-gram context graph with variable window sizes in all cases. We attribute this to the fact that modeling dialogue structural dependencies by our enhanced dependency graph captures spatial features globally, whereas the n-gram context graph is more focusing on limited local context. Anything outside the n-gram window has no impact on the decision being made.

5.4 Joint Model vs Separate Model

One of our main contributions is explicitly modeling the correlation and interaction of slots and intents by our GC-LSTM and joint decoding. Theoretically, this explicit interaction between them eventually promotes each other by achieving a trade-off. To verify this conclusion, we compare the SLU performance between the joint model and separate models. The former is our proposed model, whereas the latter is solely focusing on one task, thus without any interaction between intent detection and slot filling. It is easily observed from Table 6 that the joint model generally performs much better than two separate models. This further buttresses our claim.

Model	ATIS Dataset		Snips Dataset	
	Slot(F1)	Intent(Acc)	Slot(F1)	Intent(Acc)
Slot filling model	96.05	-	95.38	-
Intent detection model	-	97.20	-	98.0
Joint model	**96.37**	**97.88**	**95.89**	**98.43**

Table 6: Comparison between our joint model and separate models (%).

6 Conclusion

In this chapter, we propose a novel Graph-to-Seq framework to jointly perform intent detection and slot filling. The Graph Convolutional LSTM encoder not only captures the spatio-temporal sematic features in dialogue utterances, but also learns the co-occurrence relationship between intent detection and slot filling. In addition, a LSTM decoder is employed to perform final decoding of both slot filling and intent detection to alleviate GC-LSTM's burden and to fully exploring the interaction between these two tasks. On one hand, slot decoding context promotes intent detection accuracy. On the other hand, reciprocally, joint optimization also enhances slot filling performance further by optimizing intent detection objective. Experiments on two public datasets show the effectiveness of our proposed model and achieve state-of-the-arts results.

References

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

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*.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852.

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 *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757, New Orleans, Louisiana. Association for Computational Linguistics.

Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Vivian Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Proceedings of The 17th Annual Meeting of the International Speech Communication Association (INTERSPEECH 2016)*. ISCA.

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*.

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 *International Conference on Learning Representations (ICLR)*.

Changliang Li, Liang Li, and Ji Qi. 2018. A self-attentive model with gate mechanism for spoken language understanding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3824–3833, Brussels, Belgium. Association for Computational Linguistics.

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

Peiqing Niu, Zhongfu Chen, Meina Song, et al. 2019. A novel bi-directional interrelated model for joint intent detection and slot filling. *arXiv preprint arXiv:1907.00390*.

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

Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2016. Structured sequence modeling with graph convolutional recurrent networks.

Chenyang Si, Wentao Chen, Wei Wang, Liang Wang, and Tieniu Tan. 2019. An attention enhanced graph convolutional lstm network for skeleton-based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1227–1236.

Aditya Siddhant, Anuj Goyal, and Angeliki Metallinou. 2019. Unsupervised transfer learning for spoken language understanding in intelligent agents. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4959–4966.

Gokhan Tur, Dilek Hakkani-Tür, and Larry Heck. 2010. What is left to be understood in atis? In *2010 IEEE Spoken Language Technology Workshop*, pages 19–24.

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, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Jie Wu, Ian Harris, and Hongzhi Zhao. 2021. Spoken language understanding for task-oriented dialogue systems with augmented memory networks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 797–806, Online. Association for Computational Linguistics.

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip Yu. 2019. Joint slot filling and intent detection via capsule neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5259–5267, Florence, Italy. Association for Computational Linguistics.

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*.

Linhao Zhang, Dehong Ma, Xiaodong Zhang, Xiaohui Yan, and Houfeng Wang. 2020. Graph lstm with context-gated mechanism for spoken language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9539–9546.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *IJCAI*, volume 16, pages 2993–2999.

Yue Zhang, Qi Liu, and Linfeng Song. 2018b. Sentence-state LSTM for text representation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 317–327, Melbourne, Australia. Association for Computational Linguistics.