Spoken Language Understanding for Task-oriented Dialogue Systems with Augmented Memory Networks

Jie Wu
University of California Irvine
wuj8@uci.edu

Ian G. Harris
University of California Irvine
harris@ics.uci.edu

Hongzhi Zhao
Beijing Jiaotong University
hzzhao@bjtu.edu.cn

Abstract

Spoken language understanding, usually including intent detection and slot filling, is a core component to build a spoken dialog system. Recent research shows promising results by jointly learning of those two tasks based on the fact that slot filling and intent detection are sharing semantic knowledge. Furthermore, attention mechanism boosts joint learning to achieve state-of-the-art results. However, current joint learning models ignore the following important facts: 1. Long-term slot context is not traced effectively, which is crucial for future slot filling. 2. Slot tagging and intent detection could be mutually rewarding, but bidirectional interaction between slot filling and intent detection remains seldom explored. In this paper, we propose a novel approach to model long-term slot context and to fully utilize the semantic correlation between slots and intents. We adopt a key-value memory network to model slot context dynamically and to track more important slot tags decoded before, which are then fed into our decoder for slot tagging. Furthermore, gated memory information is utilized to perform intent detection, mutually improving both tasks through global optimization. Experiments on benchmark ATIS and Snips datasets show that our model achieves state-of-the-art performance and outperforms other methods, especially for the slot filling task.

1 Introduction

Task-oriented dialogue systems have attracted significant attention, which have been greatly advanced by deep learning techniques. Traditionally, these dialog systems have been built as a pipeline, with modules including spoken language understanding (SLU), dialog state tracking, action selection and language generation. Among these problems, SLU, including intention detection and slot filling (Tur and Mori, 2011), is a key yet challenging problem to parse users' utterances into semantic frames in order to capture a conversation's core meaning. Traditionally, intention detection is treated as a classification problem, whereas slot filling is usually defined as sequence labeling problem, where In-Out-Begin (IOB) format is applied for representing slot tags as illustrated in Table 1. Given an utterance, SLU determines users' intention and maps it into predefined semantic slots. The input is a sequence of words, and the output is a sequence of predefined slot IDs. A specific intent is assigned for the whole sentence.

Sentence	Flights	from	Irvine	to	Seattle
Intent	Flight				
Slots	O	O	B-fromloc	O	B-toloc

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

In the traditional pipeline approach, intent detection and slot filling are implemented separately. However, separate modeling of those two tasks is insufficient to take full advantage of all supervised signals, as they share semantic knowledge. For example, if the intent of an utterance is "find_a_flight", it is more likely to contain slots "departure_city" and "arrival_city" rather than "restaurant_name". Another drawback of the pipeline method is that errors made in upper stream modules may propagate and be amplified in downstream components, which however could possibly be eased in joint model (Zhang and Wang, 2016).

Recently, joint model for intent detection and slot filling has been proposed and achieved promising results (Liu and Lane, 2016; Goo et al., 2018; Li et al., 2018). Though achieving promising performance, their models suffer from two major issues: **1)** Modeling of slot context. Though the latent memory of RNNs can model history information, they are inherently unstable over long time sequences because the memories are the RNN hidden states. (Weston et al., 2014) observes that

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 797–806
June 6–11, 2021. ©2021 Association for Computational Linguistics

RNNs tend to focus more on short-term memories and forcefully compress historical records into one hidden state vector. Thus, simple RNNs cannot preserve long-term slot context of the conversation, which is crucial to future slot tagging. **2)** Bi-directional interaction between slot filling and intent detection. The majority of joint modeling work has studied how to utilize intent information to improve slot filling performance. However, the beneficial impact of slot information on intent detection is mostly ignored. In fact, slots and intents are closely correlative, thus mutually reinforcing each other.

In this paper, we propose a new framework to jointly model intent detection and slot filling in order to achieve a deeper level of semantic modeling. Specifically, our model is distinguished from previous work primarily in two ways.

- **Model slot context dynamically with Key-Value Memory Networks (KV-MNs).** The majority of existing work use RNNs to track slot values mentioned in previous utterances. However, RNNs tend to focus more on short-term memories. We propose to use a memory network to model slot context information as external knowledge which is acting a global information to guide slot tagging. Instead of relying on the compressed vector in RNN, KV-MNs store different historical slot tag information separately in different memory slots, which enriches the representation capacity compared with RNNs. Furthermore, slot values mentioned in the utterance are dynamically tracked, which is beneficial for subsequent slot tagging at each timestamp. Lastly, slot-level attention can model more accurately the contribution of each word in an utterance to slot tagging.

- **Model the mutual interaction between intent detection and slot filling.** The fact that intent detection and slot filling are semantically related is well-observed and how to use intent information to boost slot filling is widely explored. However, slot filling is beneficial to intent detection as well, and these benefits are yet to be explored. We propose a gating mechanism between intents and slots based on KV-MNs in order to model the interaction between intent detection and slot filling.

2 Related Works

Since intent detection can be treated as an utterance classification problem, different classification methods, such as support vector machines (SVM) and RNNs (Haffner et al., 2003; Sarikaya et al., 2011), have proposed to solve it. On the other hand, for slot filling, hidden markov models (HMM) and conditional random fields (CRF) (Lee et al., 1992; Ye-Yi Wang et al., 2005; Raymond and Riccardi, 2007) were used to solve slot filling problem. Later RNN based methods had become popular. For example, Yao et al. (2013); Mesnil et al. (2015) employed RNNs for sequence labeling in order to perform slot filling.

Alternatively, intent detection and slot filling can be done jointly to overcome the error propagation. Zhang and Wang (2016) first proposed joint work using RNNs for learning the correlation between intent 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. An attention weighted sum of all encoded hidden states was used to predict the utterance intent. All those models outperform the pipeline models via mutual enhancement between two tasks.

Most recently, some work tries to model the intent information for slot filling explicitly in the joint model. Goo et al. (2018); Li et al. (2018) proposed the gate 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). Wang et al. (2018) proposed the bi-model to consider the cross-impact between the intent and slots and achieve state-of-the-art results. Zhang et al. (2018) proposed a hierarchical capsule neural network to model the hierarchical relationship among word, slot, and intent in an utterance. Niu et al. (2019) introduces a SF-ID network to establish the interrelated mechanism for slot filling and intent detection tasks. Compared with their work, our method explicitly models the long-term slot context knowledge which is beneficial to both slot filling and intent detection.

Memory network provides a principled approach for modeling long-range dependency which has advanced many NLP tasks such as machine translation (Wang et al., 2016) and question answering (Sukhbaatar et al., 2015). The initial framework of memory networks was proposed by Weston et al. (2014). Following the idea, Sukhbaatar et al. (2015) proposed an end-to-end memory augmented model that significantly reduced the requirement of supervision during training. Key-value memory network (Miller et al., 2016) encoded prior knowledge by introducing a key memory structure which stored facts to address to the relevant memory value.

None of them is to model slot context information dynamically especially in single turn conversational systems. In this paper, we demonstrate how memory networks can be used to model long-term slot context knowledge and the interaction between intent detection and slot filling.

3 Proposed Model

Memory networks show promising results on learning long-range dependency, but they are insensitive to represent temporal dependencies between memories (Wu et al., 2018). RNNs tend to be opposite. Thus, it makes sense for us to combine those networks together to model long-term slot context information. In this section, we present a specific key-value dynamic memory module to collect and remember slot clues in the dialog context. Then context memory is used to enhance the Encoder-Decoder based model to perform slot filling and intent detection.

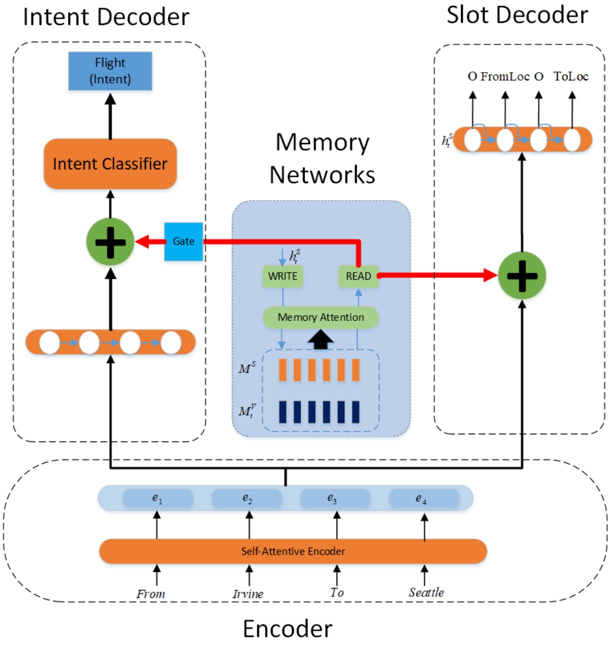

Figure 1: Framework of the proposed model

As illustrated in Figure 1, our proposed model is composed of an Encoder-Decoder, and a Key-Value Memory Module including KEY-MEMORY, VALUE-MEMORY, a memory read unit, and a memory write unit. Given a single-turn dialog, the Encoder transforms a word in user utterances into a dense vector by using a shared self-attentive encoder. Then the memory network encodes long-term slot context information by incorporating historical slot tags through memory attention and WRITE operations of the memory network. The slot decoder integrates short-term hidden state of self-attention encoder and the long-term slot context generated by attentively reading the VALUE-MEMORY to generate slot tagging at each timestamp. Later, intent decoder performs token level intent detection, which is seen as a coarse-grained intent detection result. Finally, a fine-grained intent detection is produced by gating memory modules. Both intent detection and slot filling are optimized simultaneously via a joint learning scheme.

3.1 Self-Attentive Encoder

Given an input utterance $\mathbf{X} = (x_1, x_2, \ldots, x_T)$ of T words, where each word is initially represented by a vector of dimension d, the BiLSTM (Hochreiter and Schmidhuber, 1997) is applied to learn representations of each word by reading the input utterance forward and backward to produce context sensitive hidden states $\mathbf{H} = (h_1, h_2, \ldots, h_T)$:

$$h_t = \mathbf{BiLSTM}(x_t, h_{t-1}) \tag{1}$$

Then, we use self-attention mechanism to capture the contextual information for each token. We adopt the method proposed by (Vaswani et al., 2017), where we first map the matrix of input vectors $\mathbf{X} \in \mathbb{R}^{T \times d}$ to queries ($\mathbf{Q}$), keys ($\tilde{\mathbf{K}}$) and values ($\tilde{\mathbf{V}}$) matrices by using different linear projections and the self-attention output $\mathbf{C} \in \mathbb{R}^{T \times d_1}$ is:

$$\mathbf{C} = \text{softmax}\left(\frac{\mathbf{Q}\tilde{\mathbf{K}}^{\top}}{\sqrt{d_2}}\right)\tilde{\mathbf{V}} \tag{2}$$

where d_1 and d_2 represents self-attention dimension and keys'dimension. We concatenate the output of self-attention and BiLSTM as the final encoding representation as shown in Qin et al. (2019):

$$\mathbf{E} = \mathbf{H} \oplus \mathbf{C} \tag{3}$$

where $\mathbf{E} = (e_1, \ldots, e_T) \in \mathbb{R}^{T \times (d+d_1)}$ and $\oplus$ is a concatenation operation.

3.2 Slot Decoder

Our slot deocder consists of two components: 1) the key-value memory-augmented attention model which generates slot context representation of users' utterance, and 2) the unidirectional LSTM decoder, which predicts the next slot tag step by step.

3.2.1 Dynamic Key Value Memory Network

To overcome the shortcomings of RNNs in capturing semantic clues over the long-term, we design a memory network that can preserve fine-grained semantic information of long-term slot context. We adopt a key-value memory network, which memorizes information by using a large array of external memory slots. The external memories enrich the representation capability compared with hidden vectors of RNNs and enable the KV-MNs to capture long-term data characteristics (Liu and Perez, 2017). We aim to incorporate the knowledge contained in the historical slot tags into the memory slots. The KV-MNs decompose slot semantics in an utterance into different slot categories and thus preserves more fine-grained information. In KV-MNs, a memory slot is represented by a key vector and an associated value vector.

- **KEY-MEMORY:** The KEY-MEMORY $\mathbf{K} \in \mathbb{R}^{d_k \times n}$ learns latent correlation between utterance words and slot tags, where n is the number of memory slots and d_k is the dimension of each slot. Each column vector, that is, i-th key vector $k_i \in \mathbb{R}^{d_k}$ is set to the i-th column of the KEY-MEMORY $\mathbf{K}$, which is shared by all conversation turns and fixed during the processing of word sequences.

- **VALUE-MEMORY:** Both the KEY-MEMORY and VALUE-MEMORY have the same number of memory slots. Each value memory vector stores the value of slot tag mentioned in the utterance. We form a value memory matrix $\mathbf{V}_t \in \mathbb{R}^{d_v \times n}$ by combining all n value slots. Different from KEY-MEMORY $\mathbf{K}$, VALUE-MEMORY $\mathbf{V}_t$ is word-specific and is continuously updated according to the input word sequence. During the conversation, the value of a new slot tag may be added into the VALUE-MEMORY, and an old value can be erased. In this way, we can adequately capture the slot context information on each mentioned slot. Two types of operations, **READ** and **WRITE**, are designed to manipulate the value memories.

3.2.2 Memory-augmented Decoder

As shown in Figure 1, the decoder uses the aligned BiLSTM hidden state h_t as a query to address the KEY-MEMORY looking for an attention vector a_t, and attentively reads the VALUE-MEMORY to generate slot context representation c_t.

First, we use h_t to address the KEY-MEMORY to find an accurate attention vector a_t.

$$a_t = \textbf{Address}\left(h_t, \mathbf{K}\right) \tag{4}$$

a_t is subsequently used as the guidance for reading the VALUE-MEMORY $\mathbf{V}_{t-1}$ to get the slot context representation c_t.

$$c_t = \textbf{Read}(a_t, \mathbf{V}_{t-1}) \tag{5}$$

c_t works together with the aligned encoder hidden state e_t to generate the new decoder state at the decoding step t,

$$h_t^S = \textbf{LSTM}\left(h_{t-1}^S, y_{t-1}^S, e_t \oplus c_t\right) \tag{6}$$

where h_{t-1}^S is the previous slot decoder state and y_{t-1}^S is the previous emitted slot lable distribution. After that, we use the slot decoder hidden state h_t^S to update $\mathbf{V}_t$:

$$\mathbf{V}_t = \textbf{Write}\left(h_t^S, \mathbf{V}_{t-1}\right) \tag{7}$$

Finally, the decoder state h_t^S is utilized for slot filling:

$$y_t^S = \text{softmax}\left(\mathbf{W}_h^S h_t^S\right) \tag{8}$$

$$o_t^S = \text{argmax}\left(y_t^S\right) \tag{9}$$

where $\mathbf{W}_h^S$ are trainable parameters and o_t^S is the slot label of the word at timestamp t in the utterance.

3.3 Intent Detection Decoder

Different than most existing work where intent information is used to do slot filling, our framework is directly leveraging the explicit slot context information to help intent detection. Furthermore, a gated mechanism is used in order to effectively incorporate slot memory information into intent detection. By performing gated intent detection, there are two advantages:

1. Sharing slot context information with intent detection improves intent detection performance since those two tasks are related. Furthermore, a gating mechanism which combines the intent detection information and slot context retrieved from key-value memory, regulates the degree of enhancement of intent detection to prevent information overload.

2. Through shared key-value memory, the interaction between intent detection and slot filling can be effectively modeled and executed. Plus, by jointly training those two tasks, not only can intent detection performance be improved by slot context knowledge, but also slot filling is enhanced by minimizing intent detection objective function. In other words, by learning optimal parameters of shared key-value memory, slot filling and intent detection interact in a more effective and deeper way.

Intent Detection Decoder: For intent detection, we use another uni-directional LSTM as the intent detection network. At each decode step t, the decoder state h^I_t is generated by the previous decoder state h^I_{t-1}, the previous emitted intent label distribution y^I_{t-1} and the aligned encoder hidden e_t.

$$h^I_t = \textbf{LSTM}\left(h^I_{t-1}, y^I_{t-1}, e_t\right) \tag{10}$$

Then the intent decoder state h^I_t together with the slot context c_t is utilized for final intent detection.

Gated Memory: We propose a gated mechanism to integrate slot context with intent detection. The gate regulates the degree of slot context information to feed into the intent detection task and prevent information from overloading. As shown in Figure 2, the gate **G** is a trainable fully connected network with sigmoid activation.

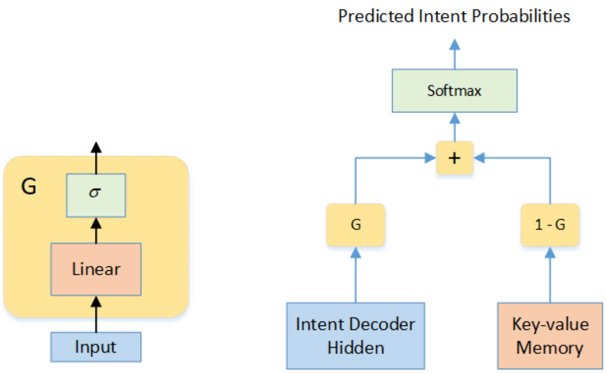

Figure 2: Intent detection with gated memory

$$h'^I_t = g_t \cdot h^I_t + (1 - g_t) \cdot c_t \tag{11}$$

where $g_t = \text{sigmoid}\left(W_t[h^I_t \bigoplus c_t] + b_t\right)$. Then, the output of gated decoder state h'^I_t is utilized for intent detection:

$$y^I_t = \text{softmax}\left(W^I_h h'^I_t\right) \tag{12}$$

$$o^I_t = \text{argmax}(y^I_t) \tag{13}$$

where y^I_t is the intent output distribution of the t-th token in the utterance, o^I_t represents the intent lable of t-th token and W^I_h are trainable parameters of the model.

The final utterance result O^I is generated by voting from all token intent results as illustrated in Qin et al. (2019).

3.4 Memory Access Operation

In this section, we detail how to access key-value memory at the decoding time step t.

KEY-MEMORY Address: $\mathbf{K} \in \mathbb{R}^{d_k \times n}$ denotes the KEY-MEMORY at decoding time step t. The addressed attention vector is given by

$$a_t = \textbf{Address}\left(h_t, \mathbf{K}\right) \tag{14}$$

where $a_t \in \mathbb{R}^n$ specifies the normalized weights assigned to the slots in **K**, with j-th slot being k_j. The attention weights $a_{t,j}$ are calculated based on the correlation between h_t and k_j:

$$a_{t,j} = \frac{exp(e_{t,j})}{\sum_{i=1}^{n} exp(e_{t,i})} \tag{15}$$

where $e_{t,j} = k_j^\top \left(\mathbf{W}_a h_t + b_a\right)$

VALUE-MEMORY Read: $\mathbf{V}_t \in \mathbb{R}^{d_v \times n}$ denotes the VALUE-MEMORY at decoding time step t. The output of reading the value memory $\mathbf{V}_t$ is given by

$$c_t = \sum_{j=1}^{n} a_{t,j} \mathrm{v}_{t,j} \tag{16}$$

VALUE-MEMORY Write: Similar to the attentive writing operation of neural turing machines (Graves et al., 2014), we define two types of operation for updating the VALUE-MEMORY: FORGET and ADD.

FORGET determines the content to be removed from memory slots. More specifically, the vector $\mathbf{F}_t \in \mathbb{R}^{d_v}$ specifies the values to be forgotten or removed on each dimension in memory slots, which is then assigned to each memory slot through normalized weights a_t. We use the slot decoder hidden state h_t^S to update $\mathbf{V}_{t-1}$. Formally, the memory after FORGET operation is given by

$$\tilde{\mathrm{v}}_{t,i} = \mathrm{v}_{t-1,i}(\mathbf{1} - a_{t,i} \cdot \mathbf{F}_t), i = 1, 2, \ldots, n \quad (17)$$

where

- $\mathbf{F}_t = \sigma(\mathbf{W}_F, h_t^S)$ is parameterized with $\mathbf{W}_F \in \mathbb{R}^{d_v \times d_h}$, and δ stands for the *Sigmoid* activation function, and $\mathbf{F}_t \in \mathbb{R}^{d_v}$;
- $a_t \in \mathbb{R}^n$ specifies the normalized weights assigned to the key memory slots in $\mathbf{K}$, and $a_{t,i}$ represents the weight associated with the i-th memory slot.

ADD decides how much current information should be written to the memory as the added content:

$$\mathrm{v}_{t,i} = \tilde{\mathrm{v}}_{t,i} + a_{t,i} \cdot \mathbf{A}_t, i = 1, 2, \ldots, n \quad (18)$$

where $\mathbf{A}_t = \sigma(\mathbf{W}_A, h_t^S)$ is parameterized with $\mathbf{W}_A \in \mathbb{R}^{d_v \times d_h}$ and $\mathbf{A}_t \in \mathbb{R}^{d_v}$. By learning the parameters of FORGET and ADD layers, our model can automatically determine which signal to weaken or strengthen based on input utterance words.

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:

$$\mathcal{L}_1 \triangleq -\sum_{j=1}^{m} \sum_{i=1}^{n_I} \hat{y}_j^{I,i} \log \left(y_j^{I,i} \right) \quad (19)$$

and

$$\mathcal{L}_2 \triangleq -\sum_{j=1}^{m} \sum_{i=1}^{n_S} \hat{y}_j^{S,i} \log \left(y_j^{S,i} \right) \quad (20)$$

where $\hat{y}_j^{I,i}$ and $\hat{y}_j^{S,i}$ are the gold intent label and gold slot label respectively, m is the number of words in a word sequence, 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 weighted-sum of these two loss functions using hyper-parameters α and β:

$$\mathcal{L}_\theta = \alpha \mathcal{L}_1 + \beta \mathcal{L}_2 \quad (21)$$

Through joint training, the key-value memory shared by those two tasks can learn the shared representations and interactions between them, thus further promoting each other's performance and easing the error propagation compared with pipeline models.

4 Experiments

4.1 Setup

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

To justify the generalization of our proposed mode, we also execute our experiment on another NLU dataset collected by Snips (Coucke et al., 2018) [1]. This data is collected from the Snips personal voice assistant, where the number of samples for each intent is approximately the same. 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 single-domain ATIS dataset, Snips is more complicated mainly due to the intent diversity and large vocabulary (Goo et al., 2018). For example, GetWeather and BookRestaurant in Snips are from different topics, resulting in a larger vocabulary. On the other hand, intents in ATIS are all about flight information with similar vocabularies.

In our experiments, we set the dimension of word embedding to 256 for ATIS and 200 for Snips dataset. L2 reularization used in our model is 1×10^{-6} and dropout ratio is set to 0.4 for reducing overfit. The number of memory columns is set to 20 for both datasets, and the dimensions of memory column vectors are set to 64 for ATIS, and to 200 for Snips. The optimizer is Adam (Kingma and Ba, 2014). During our experiments, we select the model which works the best on the development set, and then evaluate it on the test set.

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

We carefully choose some representative works, for example, 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) and Stack-Propagation (Qin et al., 2019) as our baselines. When doing the comparison, we adopt the reported results from those papers directly.

4.2 Results

In order to have fair comparison with others' work, we adopt the same metrics to evaluate our model. That is, we evaluate slot filling using F1 score, intent prediction using accuracy, and sentence-level semantic frame parsing using whole frame accuracy.

Table 2 shows the experiment results of the proposed model on ATIS and Snips datasets. From the table, we can see that our model outperforms all the baselines in all three aspects: slot filling (F1), intent detection (Acc) and setence accurancy (Acc), demonstrating that explicitly modeling slot context and strong relationships between slots and intent can benefit SLU effectively from the key-value memory. In the ATIS dataset, compared with the best prior joint work Stack-Propagation (Qin et al., 2019), we achieve F1 score as 96.13 which is even slightly better than Stack-propagation's F1 score (96.10) with BERT model. This signifies that our key-value memory can not only capture long-term slot context, but also model correlation between slot filling and intent detection, which can be further optimized by joint training. What's more, in the Snips dataset, our model achieves good results in both slot filling and overall sentence. Specifically, slot filling was improved by almost 1.0%, and sentence accuracy by 1.4%. Generally, ATIS dataset is a simpler SLU task than Snips, and so the room to be improved is relatively small. On the other hand, Snips is more complex so that it needs more complicated model to capture long-term context and share the knowledge across different topics.

4.3 Analysis

From Section 4.2, we can see good improvements on both datasets, but we want to know how each component impacts SLU performance.

4.3.1 Ablation Study

In this section, we explore how each component contributes to our full model. Specifically, we ablate three important scenarios and conduct them in this experiment. Note that all the variants are based on joint learning.

- Without key-value memory and gating architecture for integrating slot context information with intent detection. This is the model similar to Qin et al. (2019).
- Only with key-value memory, but without sharing slot context information with intent detection.
- With key-value memory and sharing, but without gating architecture, where only key-value memory is applied to model slot context and that information is directly fed into intent detection.

Table 3 shows the joint learning performance of our model on ATIS and Snips datasets by removing one component at one time. First, if we remove key-value memory and gating architecture, the performance drops dramatically compared with our proposed model. This is expected as it does not have any of our improvements. Then we only consider key-value memory to model slot context. From Table 3, we can see that key-value memory does improve performance in a large scale. The result can be interpreted as indicating that key-value memory learns long-term slot context representation effectively, which does compensate the weakness of RNN. In the following, we apply key-value memory and also share it with intent detection without gating. It is noticeable that SLU performance is enhanced further. Sharing slot context information with intent detection not only improves intent accuracy, but also betters slot filling through joint optimization. Finally, when we add gating mechanism, the performance improves further. We attribute this to gating mechanism that regulates the degree of slot context information to feed into intent detection task and prevent information from overloading.

We also study how the number of memory slots and the dimension of memory slots impacts SLU performance. Figure 3 shows the performance change with different hyper-parameters. We found that the optimal size of memory slots for ATIS and Snips dataset is 20, whereas the optimal dimension of memory slots is 64 for ATIS and 200 for Snips respectively.

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
Stack-Propagation(Qin et al., 2019)	95.90	96.90	86.50	94.20	98.0	86.90
Our model	**96.13**	**97.20**	**87.12**	**95.13**	**98.14**	**88.14**

Table 2: SLU Performance comparison on ATIS and Snips datasets (%). The improved results are written in bold.

Model	ATIS Dataset			Snips Dataset		
	Slot(F1)	Intent(Acc)	Sent.(Acc)	Slot(F1)	Intent(Acc)	Sent.(Acc)
Without K-V memory and sharing	95.72	96.64	85.78	94.08	97.42	86.42
With K-V memory without sharing with intent	95.95	96.66	86.56	94.46	98.09	87.0
With K-V memory and sharing without gate	96.08	96.86	87.0	94.76	98.0	87.28
Full Model	**96.13**	**97.20**	**87.12**	**95.13**	**98.14**	**88.14**

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

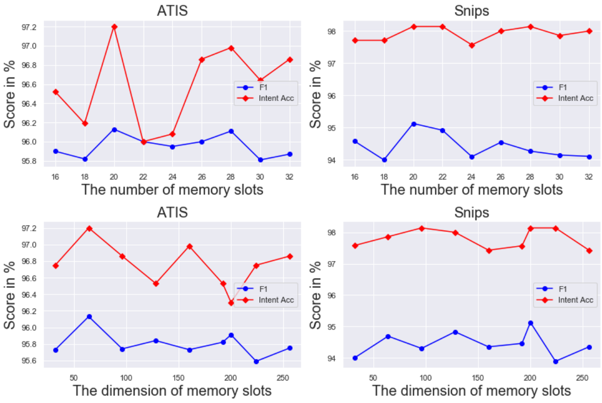

Figure 3: SLU performance on different hyper-parameters in key-value memory networks

4.3.2 Memory Attention

Analyzing the attention weights has been frequently used to show the memory read-out, since it is an intuitive way to understand the model dynamics. Figure 4 shows the attention vector for each decoded slot, where each row represents attention vector a_t. Our model has a sharp distribution over the memory, which implies that it is able to select the most related memory slots from the value memory. For example, when decoding "san", our model selects memory slot 1, 7, 8,15 from the value memory to read context information, where memory slot 7 and 15 are representing word "from" and memory slot 1 representing word "flight". In other words, words "flight" and "from" contribute more than other previous words in order to decode "san" to B-fromloc.city_name.

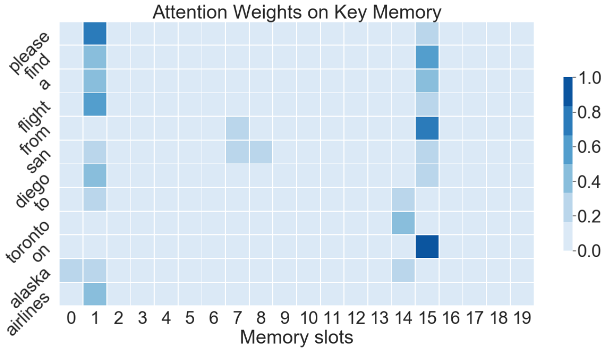

Figure 4: Key memory attention visualization from the ATIS dataset

5 Conclusion

In this paper, we propose a joint model to perform spoken language understanding with an augmented key-value memory to model slot context in order to capture long-term slot information. In addition, we adopt a gating mechanism to incorporate slot context information for intent classification to improve intent detection performance. Reciprocally, joint optimization promotes slot filling performance further by memory sharing between those two tasks. 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*.

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.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

P. Haffner, G. Tur, and J. H. Wright. 2003. Optimizing svms for complex call classification. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).*, volume 1, pages I–I.

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*.

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*.

C. Lee, R. Pieraccini, E. Tzoukermann, J. Gauvain, E. Levin, J. Wilpon, and Z. Gorelov. 1992. A speech understanding system based on statistical representation of semantics. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, volume 1, pages 193–196, Los Alamitos, CA, USA. IEEE Computer Society.

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*.

Fei Liu and Julien Perez. 2017. Gated end-to-end memory networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1–10.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

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*.

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. *arXiv preprint arXiv:1909.02188*.

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

R. Sarikaya, G. E. Hinton, and B. Ramabhadran. 2011. Deep belief nets for natural language call-routing. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5680–5683.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

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, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2016. Memory-enhanced decoder for neural machine translation. *arXiv preprint arXiv:1606.02003*.

Yu Wang, Yilin Shen, and Hongxia Jin. 2018. A bi-model based rnn semantic frame parsing model for intent detection and slot filling. *arXiv preprint arXiv:1812.10235*.

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

Chien-Sheng Wu, Andrea Madotto, Genta Indra Winata, and Pascale Fung. 2018. End-to-end dynamic query memory network for entity-value independent task-oriented dialog. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6154–6158. IEEE.

Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu. 2013. Recurrent neural networks for language understanding. In *Interspeech*, pages 2524–2528.

Ye-Yi Wang, Li Deng, and A. Acero. 2005. Spoken language understanding. *IEEE Signal Processing Magazine*, 22(5):16–31.

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

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip S. Yu. 2019. Joint slot filling and intent detection via capsule neural networks.

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.