SPEAKER-SENSITIVE DUAL MEMORY NETWORKS FOR MULTI-TURN SLOT TAGGING

Young-Bum Kim*, Sungjin Lee**, Ruhi Sarikaya*

Amazon Alexa Brain, Seattle, WA*,
Microsoft Research, Redmond, WA**

ABSTRACT

In multi-turn dialogs, natural language understanding models can introduce obvious errors by being blind to contextual information. To incorporate dialog history, we present a neural architecture with Speaker-Sensitive Dual Memory Networks which encode utterances differently depending on the speaker. This addresses the different extents of information available to the system - the system knows only the surface form of user utterances while it has the exact semantics of system output. We performed experiments on real user data from Microsoft Cortana, a commercial personal assistant. The result showed a significant performance improvement over the state-of-the-art slot tagging models using contextual information.

1. INTRODUCTION

Major technology companies have been investing in natural language understanding systems as effective tools for human-computer communication [1, 2, 3, 4, 5, 6, 7, 8]. Of particular interest are task-oriented personal assistants, e.g., Amazon’s Alexa, Apple Siri and Microsoft Cortana. In task-oriented dialogs, the main role of the system is to fill out the associated slots to the semantic frame of the requested task. As the demand for fulfilling more complex tasks with voice-interface agents keeps increasing, it becomes crucial to address some issues faced when handling multi-turn interaction. For example, in Table 1, the user’s utterance at turn 3 could mean either number_people or time without any context.

Traditionally, the natural language understanding (NLU) module [9] delegates the task of contextual interpretation to downstream modules such as dialog state tracker [10]. However, given that most dialog state trackers just exploit the slot-values recognized by the NLU module as is without further transformation based on context, the NLU errors due to the context-insensitivity are very likely to propagate through downstream modules. Thus it is critical to reduce the context-related NLU errors as early in the system pipeline as possible.

With this aim, a host of previous studies has attempted to incorporate dialog history for contextual interpretation [11, 12, 13]. A large body of work, however, exploited only previous turn information such as the NLU results at the previous turn. Not only are they limited in the range of history information but also they internally suffer the error propagation problems by directly using previous predictions. With the rise of end-to-end deep neural network models, Chen et al. [14] addressed such shortcomings with end-to-end memory networks [15] where the embeddings of all past user utterances are stored in a memory. In their work, however, no system’s information was used which will prove powerful in our experimental results.

In this paper we propose an end-to-end neural architecture for slot tagging that uses Speaker-Sensitive Dual Memory Networks (SSDMNs) to leverage both user and system information in a tailored manner. Specifically, since the system only gets to see the surface form of user’s utterance, we encode each word of all past user’s utterances using the Long Short-Term Memory (LSTM) model [16]. We store the encoded vectors of each word in a memory and use an attention mechanism [17] to selectively use only relevant words to the current user input. Whereas the system already knows the exact semantics of its past utterances, thus we directly encode the targeted slots of system’s question to avoid parsing noisy natural language expressions. Due to this different nature, we introduce a separate memory storing all system’s targeted slot embeddings. The attention mechanism again allows to selectively use relevant system information. Hori et al. [18] adopted a similar concept, role-dependent encoding, but their work is not aimed for slot tagging, nor does it exploit the system output semantics. Furthermore, we adopt an efficient pre-training method [19, 20, 21, 22] for system’s targeted slot encoding that further improves the
Table 1. An example dialog for restaurant reservation. The targeted slots are the slots that the system asks about at each turn.

<table>
<thead>
<tr>
<th>Turn</th>
<th>Speaker</th>
<th>Utterance</th>
<th>Targeted Slots</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>System</td>
<td>Where do you want to reserve?</td>
<td>place_name</td>
</tr>
<tr>
<td></td>
<td>User</td>
<td>Pizza Hut</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>System</td>
<td>For how many people?</td>
<td>number_people</td>
</tr>
<tr>
<td></td>
<td>User</td>
<td>Two</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>System</td>
<td>Okay. What time should I make the reservation for?</td>
<td>{time, date}</td>
</tr>
<tr>
<td></td>
<td>User</td>
<td>Actually it’s three</td>
<td></td>
</tr>
</tbody>
</table>

system’s performance. To verify the efficacy of the proposed model, we performed evaluation on a large amount of real data gathered from five different domains of commercial personal assistants. In comparison to previous approaches our method showed a large improvement.

The rest of the paper is structured as follows. In Section 2 we describe SSDMNs in depth. In Section 3 we discuss our experiments. We finish with conclusions in Section 4.

2. SPEAKER-SENSITIVE DUAL MEMORY NETWORKS

At turn $t$, the multi-turn slot tagging model takes a user input $u_{i-1}$ and system output $m_{i-1}$ by considering the contextual information, i.e., a sequence of user’s past utterances $\{u_1, \ldots, u_{i-1}\}$ and the corresponding system outputs $\{m_1, \ldots, m_{i-1}\}$.

2.1. Overall architecture

The overall SSDMNs architecture is illustrated in Figure 1. The model first encodes the last user input and system output to produce vector embeddings. Then it updates the dual memories with the generated vectors. After that, as it encodes the current user utterance, at each word position, it computes the attention weights to find relevant information. Finally it makes prediction for each word by jointly considering the encoded vector of each word and the memory content averaged with respect to the attention weights. The main steps are described below.

2.2. User utterance encoding

In order to obtain a more contextful vector representation of the $j$-th word $u_i^j$ of the user utterance at turn $i$, we adopt a stateful LSTM instead of a vanilla LSTM encoder - the LSTM output for the last word in an utterance is fed as the initial state into the LSTM network for the next utterance while all LSTM weights are shared. The stateful LSTM runs over both past user utterances and the current input:

$$\tilde{u}_i^j = \text{StatefulLSTM}(u_i^j), i < t$$

$$h_i^j = \text{StatefulLSTM}(u_i^j)$$

2.3. System output encoding

Since the system has the exact semantics of system utterances, we directly encode the targeted slots that usually have strong influence on the following user’s response. Concretely, for each system output, we are given a k-hot vector $m_i^j \in \mathbb{R}^l$, where $l$ is the number of slot tags, which indicates which slot is targeted by the system’s output. Then we use a trainable projection matrix $P \in \mathbb{R}^{d \times l}$ to map the k-hot vector to a $d$-dimensional dense vector $\tilde{m}_i^j$:

$$\tilde{m}_i^j = Pm_i^j$$

Note that since $m_i^j$ is a k-hot vector, $\tilde{m}_i^j$ is the sum of the targeted slot embeddings. It has been observed that pre-trained embeddings can serve as good initialization values that help avoid converging to bad local optima and speed up model training. Inspired by [19], we adopt Canonical Correlation Analysis (CCA) to efficiently pre-train the projection matrix. The CCA algorithm finds a lower dimensional projection that maximizes the correlation between a set of random vector pairs:

- $\{a_i^j\}$ is a set of zero vectors in which the entry corresponding to the slot tag of the $i$-th sentence is set to 1.
- $\{b_i^j\}$ is a set of zero vectors in which the entries corresponding to words spanned by the tag are set to 1.

2.4. Memory update

Although an LSTM network can capture long-term dependencies, it has been argued that LSTM networks’
fixed-size memory may not be enough to contain a large amount of information encoded in variable-sized inputs [17]. Thus before performing the tag prediction we update the dual memories with the encoded vectors of each word of the user utterances ($\tilde{u}_k^i$) and system output vectors ($\tilde{m}_i^i$) encoded in the manner described above, respectively.

2.5. Attention mechanism

Since not every piece of history information is helpful in making prediction for a current word, we selectively use only relevant information. To this end, we first compute the attention weights for both memories:

$$
\alpha_i^j = \text{softmax}(W_\alpha(\tilde{u}_i^j \circ \tilde{h}_k^i) + b_\alpha)
$$

$$
\beta_i^i = \text{softmax}(W_\beta(\tilde{m}_i^i \circ \tilde{h}_k^i) + b_\beta)
$$

where $\tilde{h}_k^i$ is the LSTM output for $u_k^i$, $W_\alpha$ and $W_\beta$ are weight matrices, $b_\alpha$ and $b_\beta$ are bias vectors, $\circ$ denotes element-wise vector multiplication, and $\text{softmax}(x_i^j) = e^{x_i^j}/\sum_{k,l} e^{x_k^l}$. Then we compute the weighted sum of memory entries:

$$
\tilde{u} = \sum_{i,j} \alpha_i^j \tilde{u}_i^j
$$

$$
\tilde{m} = \sum_{i} \beta_i^i \tilde{m}_i^i
$$

2.6. Tag prediction

Finally we use $\tilde{u}$ and $\tilde{m}$ along with $\tilde{h}_k^i$ to make the tag prediction:

$$
\hat{y}_{k,l} = \exp \{z_{k,l}^t\} / \sum_{l'} \exp \{z_{k,l'}^t\}
$$

where $z_{k,l}^t = W_h \tilde{h}_k^i + W_u \tilde{u} + W_m \tilde{m}$ and $W_s$ are all trainable weight matrices.

3. EXPERIMENTS

3.1. Models

To evaluate the SSDMNs model, we conducted comparative experiments with the following systems:

- **LSTM** uses the stateful LSTM only with no contextual information used.
- **+LS** additionally uses the last system output.
- **+PreLS** uses the pre-trained system tag projection matrix for encoding the last system output to show the efficacy of the CCA pre-training scheme.

### Table 2. Data description

<table>
<thead>
<tr>
<th>Domain</th>
<th>Train</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orderfood</td>
<td>5,993</td>
<td>1,333</td>
<td>1,714</td>
</tr>
<tr>
<td>Reservation</td>
<td>5,130</td>
<td>1,215</td>
<td>1,339</td>
</tr>
<tr>
<td>Taxi</td>
<td>6,120</td>
<td>1,216</td>
<td>1,414</td>
</tr>
<tr>
<td>Events</td>
<td>4,347</td>
<td>822</td>
<td>850</td>
</tr>
<tr>
<td>Movieticket</td>
<td>3,781</td>
<td>712</td>
<td>738</td>
</tr>
<tr>
<td>Average</td>
<td>5,074</td>
<td>1,059</td>
<td>1,211</td>
</tr>
<tr>
<td>Domain</td>
<td>LSTM</td>
<td>+LS</td>
<td>+PreLS</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>-----</td>
<td>--------</td>
</tr>
<tr>
<td>Orderfood</td>
<td>87.38</td>
<td>93.70</td>
<td>94.10</td>
</tr>
<tr>
<td>Reservation</td>
<td>86.29</td>
<td>91.96</td>
<td>92.81</td>
</tr>
<tr>
<td>Taxi</td>
<td>89.46</td>
<td>93.77</td>
<td>94.58</td>
</tr>
<tr>
<td>Events</td>
<td>85.17</td>
<td>91.39</td>
<td>90.80</td>
</tr>
<tr>
<td>Movieticket</td>
<td>88.06</td>
<td>93.76</td>
<td>94.53</td>
</tr>
<tr>
<td>Average</td>
<td>87.27</td>
<td>92.92</td>
<td>93.36</td>
</tr>
</tbody>
</table>

Table 3. Comparative performance in F1 score on five domains.

- MNs similarly to the Memory Networks model in Chen et al. [14], the system does not use system’s output information.
- MNs+S MNs with system’s output in surface form.

3.2. Data

The data is collected from five domains of Microsoft Cortana. Table 2 shows the number of utterances of the training, development and test datasets of each domain. All datasets were labeled by expert editors.

3.3. Training setting

In our experiments, all the models were implemented using Dynet [23] and were trained using Stochastic Gradient Descent (SGD) with Adam [24]—an adaptive learning rate algorithm. We used the initial learning rate of $4 \times 10^{-4}$ and left all the other hyper parameters as suggested in [24]. Each SGD update was computed with Intel MKL (Math Kernel Library) without minibatching. We used the dropout regularization [25] with the keep probability of 0.4. Both dimensions of the input and output of the stateful LSTM model were 100. To initialize word embedding, we used the pre-trained GloVe model [26]. For system slot tag embedding, we used CCA to induce 100 dimensional vector representation from 20 personal assistant domains consisting of 3M utterances and 130 distinct tags.

3.4. Results

To compute slot F1-score, we used the standard CoNLL evaluation script. In Table 3, the stateful LSTM-only model (LSTM) achieved 87.27% in averaged F1 score. The improved performance of the +LS model, 92.92%, confirms the same findings of previous studies on the effectiveness of previous turn information for contextual interpretation. The further performance increase of the +PreLS model, achieving 93.36%, demonstrates the possible gain with the proposed pre-training scheme. Interestingly the MNs model couldn’t show a better performance than the +LS model, even after incorporating system’s output in surface form, justifying our novel contribution on the dual encoding of system output. This claim is also supported by the best performance of the SSDMNs model reaching 95.17%.

4. CONCLUSION

We presented a novel Speaker-Sensitive Dual Memory Networks model for the multi-turn slot tagging task which extends the state-of-the-art Memory Networks-based approach with tailored memories to exploit the exact semantic knowledge on system output. We also described an efficient pre-training scheme to obtain better slot tag embeddings. The large performance improvement on real data demonstrates the efficacy of SSDMNs. Future work includes a detailed analysis on various reasoning behaviors of the model depending on different contexts. Also we plan to incorporate more elements of dialog state beyond system output. Another interesting direction is to perform multiple memory references to handle more complex discourse phenomena such as ellipsis, anaphora resolution.

5. REFERENCES


