End-to-End Memory Networks with Knowledge Carryover for Multi-Turn Spoken Language Understanding

Yun-Nung Chen†, Dilek Hakkani-Tür, Gokhan Tur, Jianfeng Gao, and Li Deng

National Taiwan University, Taipei, Taiwan†
Microsoft Research, Redmond, WA, USA
{y.v.chen†, dilek, gokhan.tur}@ieee.org, {jfgao, deng}@microsoft.com

Abstract

Spoken language understanding (SLU) is a core component of a spoken dialogue system. In the traditional architecture of dialogue systems, the SLU component treats each utterance independent of each other, and then the following components aggregate the multi-turn information in the separate phases. However, there are two challenges: 1) errors from previous turns may be propagated and then degrade the performance of the current turn; 2) knowledge mentioned in the long history may not be carried into the current turn. This paper addresses the above issues by proposing an architecture using end-to-end memory networks to model knowledge carryover in multi-turn conversations, where utterances encoded with intents and slots can be stored as embeddings in the memory and the decoding phase applies an attention model to leverage previously stored semantics for intent prediction and slot tagging simultaneously. The experiments on Microsoft Cortana conversational data show that the proposed memory network architecture can effectively extract salient semantics for modeling knowledge carryover in the multi-turn conversations and outperform the results using the state-of-the-art recurrent neural network framework (RNN) designed for single-turn SLU.

Index Terms: spoken language understanding, end-to-end, memory network, embedding

1. Introduction

In the past decades, goal-oriented spoken dialogue systems (SDS) are being incorporated in various devices and allow users to speak to systems in order to finish tasks more efficiently, for example, the virtual personal assistants Microsoft's Cortana and Apple's Siri. A key component of the understanding system is an spoken language understanding (SLU) module-it parses user utterances into semantic frames that capture the core meaning, where three main tasks of SLU are domain classification, intent determination, and slot filling [1]. A typical pipeline of SLU is to first decide the domain given the input utterance, and based on the domain, to predict the intent and to fill associated slots corresponding to a domain-specific semantic template. The upper block of Figure 1 shows a communication-related user utterance, "*just send email to bob about fishing this weekend*" and its semantic frame, send_email(contact_name="bob", subject="fishing this weekend") [2]. Traditionally, domain detection and intent prediction are framed as classification problems, where several classifiers such as support vector machines and maximum entropy are employed [3, 4, 5]. Then slot filling is framed as a sequence tagging task, where the IOB (in-out-begin) format is applied for representing slot tags as illustrated in Figure 1, and hidden Markov models (HMM) or conditional

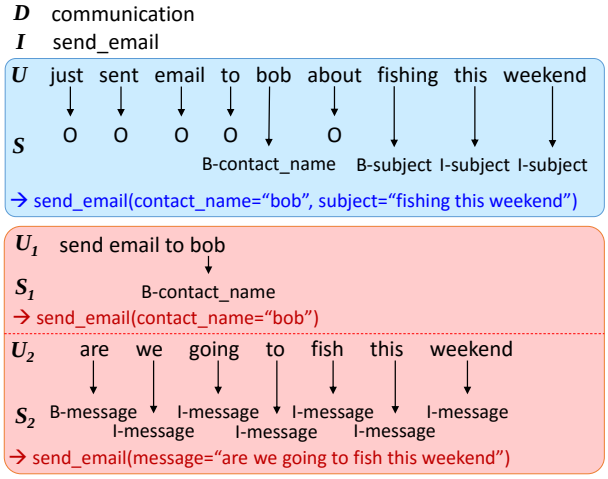

Figure 1: *Example utterances (U) annotated with its domain (D) and intent (I) and semantic slots in the IOB format (S). The upper block shows an example for a single-turn utterance, and the lower block shows similar utterances in the multi-turn scenario.*

random fields (CRF) have been employed for tagging [6, 7, 8].

With the advances on deep learning, deep belief networks (DBNs) with deep neural networks (DNNs) have been applied to for domain and intent classification [9, 10, 11]. Recently, Ravuri et al. proposed an RNN architecture for intent determination [12]. For slot filling, deep learning has been viewed as a feature generator and the neural architecture can be merged with CRFs [13]. Yao et al. and Mesnil et al. later employed RNNs for sequence labeling in order to perform slot filling [14, 15]. Recently, Hakkani-Tür proposed RNN-based joint semantic parsing for predicting intents and filling slots in the mean time [16]. However, above work focuses on SLU for single-turn interactions, where each utterance is treated independently.

The contextual information has been shown useful for SLU modeling [17, 18, 19, 20]. For example, the lower block of Figure 1 shows two utterances, where the latter containing the message content in the email, so it is more likely to estimate the semantic slot message with the same intent send_email if the contextual knowledge is kept. Bhargava et al. incorporated the information from previous intra-session utterances into the SLU tasks on a given utterance by applying SVM-HMMs to sequence tagging and obtained the improvement [17]. Also, contextual information has been incorporated into the recurrent neural network (RNN) for improved domain classification, in-

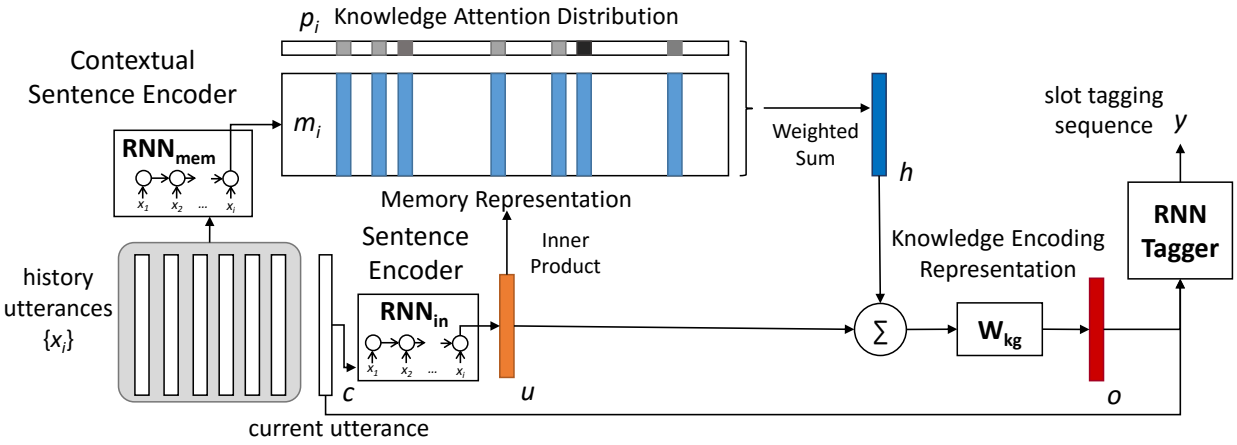

Figure 2: *The illustration of the proposed end-to-end memory network model for multi-turn SLU.*

tent prediction, and slot filling [18, 21]. However, most prior work exploited information only from the previous turn, ignoring the long-term contexts. Another constraint is that the models require supervision at each layer of the network, and there is also no unified architecture that can perform multi-turn SLU in an end-to-end framework.

Recently there has been a resurgence in computational models using explicit storage and a notion of attention [22, 23, 24, 25]; manipulating such a storage allows multiple computational steps and can model long-term dependencies in sequential utterances. Basically, the storage is endowed with a continuous representation modeled by neural networks, where the stored representations can be read and written to encode knowledge. Motivated by the idea, this paper presents a recurrent neural network (RNN) architecture where the recurrence reads from a possibly large external memory before tagging the current utterance. The model training does not require paired data for each layer of the network; that is, the proposed model can be trained end-to-end directly from input-output pairs. To the best of our knowledge, this is the first attempt of employing an end-to-end neural network model to model long-term knowledge carryover for multi-turn SLU.

神经图灵机
memory network

2. End-to-End Memory Networks

For the SLU task, our model takes a discrete set of history utterances $\{x_i\}$ that are stored in the memory, a current utterance $\boldsymbol{c} = w_1, ..., w_T$, and outputs corresponding semantic tags $\boldsymbol{y} = y_1, ..., y_T$, where a semantic tag consists intent and slot information. The proposed model is illustrated in Figure 2 and detailed below.

2.1. Architecture

The model embeds all utterances into a continuous space and stores all x's embedding to the memory. The representation of the current utterance is then compared with memory representations to encode carried knowledge via an attention mechanism. Then the encoded knowledge is taken together with the word sequence for estimating the semantic tags. Four main procedures are described below.

Memory Representation: To store the knowledge in the previous turns, we convert each utterance from previous turns, x_i, into a memory vectors m_i with dimension d by embedding the utterances in a continuous space through an RNN. The current utterance c is also embedded to a vector u with the same dimension.

$$m_i = \text{RNN}_{\text{mem}}(x_i), \tag{1}$$

$$u = \text{RNN}_{\text{in}}(c), \tag{2}$$

where RNN_{mem} and RNN_{in} are tied together for encoding contexts and the current utterance consistently. Therefore, the sequential information can be kept for better representations [16].

Knowledge Attention Distribution: In the embedding space, we compute the match between the current utterance u and each memory vector m_i by taking the inner product followed by a softmax.

$$p_i = \text{softmax}(u^T m_i), \tag{3}$$

where $\text{softmax}(z_i) = e^{z_i} / \sum_j e^{z_j}$ and p_i can be viewed as attention distribution for modeling knowledge carryover in order to understand the current utterance.

Knowledge Encoding Representation: In order to encode the knowledge from history, a history vector h is a sum over the memory embeddings weighted by the attention distribution.

$$h = \sum_i p_i m_i. \tag{4}$$

Because the function from input to output is smooth, we can easily compute gradients and back propagate through it. Then the sum of the memory vector h and the current input embedding u are then passed through a weight matrix W_{kg} to generate an output knowledge encoding vector o,

$$o = W_{\text{kg}}(h + u), \tag{5}$$

where W_{kg} is a fully-connected neural network for encoding carried knowledge.

Sequence Tagging: Different from the classification task most of work focused on [25], our memory architecture is to provide additional knowledge for improving tagging performance. Therefore, to estimate the tag sequence $\boldsymbol{y}$ corresponding to an input word sequence $\boldsymbol{c}$, we use an RNN module for training a slot tagger, where the encoded knowledge o is fed into the input of the model in order to model knowledge carryover:

$$\boldsymbol{y} = \text{RNN}(o, \boldsymbol{c}). \tag{6}$$

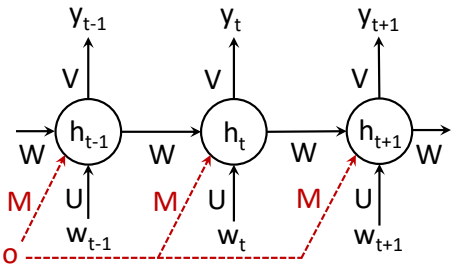

Figure 3: *The RNN model architecture for tagging. The dotted red lines show the encoded knowledge from history in the multi-turn interactions.*

2.2. Recurrent Neural Network (RNN) Tagger

The goal of the SLU model is to assign a semantic tag for each word in the current utterance. That is, given $\boldsymbol{c} = w_1, ..., w_n$, the model is to predict $\boldsymbol{y} = y_1, ..., y_n$ where each tag y_i is aligned with the word w_i. We use the Elman RNN architecture, consisting of an input layer, a hidden layer, and an output layer [26]. The input, hidden and output layers consist of a set of neurons representing the input, hidden and output at each time step t (w_t, h_t, and y_t) respectively. The solid black part in the Figure 3 illustrates the architecture of the vanilla RNN model.

$$h_t = \phi(Ww_t + Uh_{t-1}), \quad (7)$$

$$\hat{y_t} = \text{softmax}(Vh_t), \quad (8)$$

where ϕ is a smooth bounded function such as tanh, and $\hat{y_t}$ is the probability distribution over of semantic tags given the current hidden state h_t. The sequence probability can be formulated as

$$p(\boldsymbol{y} \mid \boldsymbol{c}) = p(\boldsymbol{y} \mid w_1, ..., w_T) = \prod_i p(y_i \mid w_1, ..., w_i). \quad (9)$$

The model can be trained using backpropagation to maximize the conditional likelihood of the training set labels.

To overcome the frequent gradient vanishing issue when modeling long-term dependencies, gated RNN was designed to use a more sophisticated activation function than a usual activation function, consisting of affine transformation followed by a simple element-wise nonlinearity by using gating units [27], such as long short-term memory (LSTM) and gated recurrent unit (GRU) [28, 29]. RNNs employing either of these recurrent units have been shown to perform well in tasks that require capturing long-term dependencies [15, 30, 31, 32]. In this paper, we use RNN with GRUs to allow each recurrent unit to adaptively capture dependencies of different time scales.

2.2.1. Gated Recurrent Units (GRU)

A GRU has two gates, a *reset gate* r, and an *update gate* z [29, 27]. The reset gate determines the combination between the new input and the previous memory, and the update gate decides how much the unit updates its activation, or content.

$$r = \sigma(W^r w_t + U^r h_{t-1}), \quad (10)$$

$$z = \sigma(W^z w_t + U^z h_{t-1}), \quad (11)$$

where σ is a logistic sigmoid function.

Then the final activation of the GRU at time t, h_t, is a linear interpolation between the previous activation h_{t-1} and the candidate activation $\tilde{h_t}$:

$$h_t = (1 - z) \odot \tilde{h_t} + z \odot h_{t-1}, \quad (12)$$

$$\tilde{h_t} = \phi(W^h w_t + U^h(h_{t-1} \odot r))), \quad (13)$$

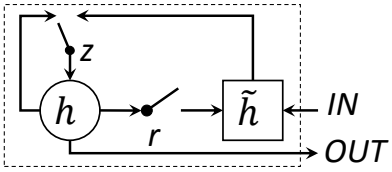

Figure 4: *The illustration of a GRU cell as depicted in [27].*

where $\odot$ is an element-wise multiplication. When the reset gate off, it effectively makes the unit act as if it is reading the first symbol of an input sequence, allowing it to forget the previously computed state. Figure 4 shows the gating mechanism of a GRU cell. RNN-GRU can yield comparable performance as RNN-LSTM with need of fewer parameters and less data for generalization [27, 33]. Therefore, this paper employs GRUs for all RNN models in the experiments.

2.2.2. Knowledge Carryover

In order to model the encoded knowledge from previous turns, for each time step t, the knowledge encoding vector o in (5) is fed into the RNN model together with the word w_t. Therefore, the hidden layer in the RNN model can be formulated as

$$h_t = \phi(Mo + Ww_t + Uh_{t-1}) \quad (14)$$

to replace (7) as illustrated in Figure 3, where the dotted red lines indicate the carried knowledge. RNN-GRU can incorporate the encoded knowledge in the similar way, where Mo can be added into (10), (11) and (12) for modeling contextual knowledge similarly.

2.3. Model Training

To train the RNN-GRU with knowledge carryover, the weights of M, W, and U for both gates and the weights of RNN_{mem}, RNN_{in}, and W_{kg} can be jointly updated via backpropagation from the RNN tagger.

3. Experiments

3.1. Dataset

The data is collected from Microsoft Cortana, where we extract multi-turn interactions (#turn $\geq$ 5) in the communication domain for experiments. There are 32 semantic tags (the concatenation of intents and slots). The number of multi-turn utterances for training is 1,005, one for testing is 1,001, and one for development is 207. There are 13,779 single-turn utterances, which are used to train the baseline SLU model for comparison.

3.2. Implementation Setting

For training models, we use mini-batch stochastic gradient descent with batch size 50 and the adam optimizer with default parameters (a fixed learning rate 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e^{-08}$) [34]. The number of iterations per batch is set to be 50 in the experiments. The dimension of word and memory embeddings is set as 150 and the size of the hidden layer in the RNN is set as 100. The memory size is 20 to store carried knowledge from previous 20 turns. The dropout rate is set to be 0.5 to avoid overfitting.

Table 1: *The performance of multi-turn SLU in terms of first turn only, other turns, and overall results (%).*

	Model	Training	Knowledge Encoding	First Turn			Other Turns			Overall		
				P	R	F1	P	R	F1	P	R	F1
(a)	RNN Tagger	single-turn	✗	53.6	**69.8**	60.6	14.3	18.8	16.2	22.5	29.5	25.5
(b)		multi-turn	✗	70.4	46.3	55.8	41.5	50.8	45.7	45.1	49.9	47.4
(c)	Encoder-Tagger	multi-turn	current utterance (c)	74.5	47.0	57.6	54.8	57.3	56.0	57.5	55.1	56.3
(d)		multi-turn	history + current (x, c)	78.3	63.1	69.9	60.3	61.2	60.8	63.5	61.6	62.5
(e)	Memory Network	multi-turn	history + current (x, c)	**79.5**	67.8	**73.2**	**65.1**	**66.2**	**65.7**	**67.8**	**66.5**	**67.1**

Table 2: *The performance of multi-turn SLU in terms of intent and slot results (%).*

Model	Intent			Slot		
	P	R	F1	P	R	F1
(a)	34.0	31.1	32.5	29.2	38.3	33.1
(b)	84.1	82.8	83.4	63.7	66.5	65.1
(c)	78.3	74.0	76.1	68.5	62.0	65.1
(d)	**91.5**	86.7	**89.0**	68.7	66.0	67.3
(e)	87.6	**87.3**	87.5	**73.7**	**70.8**	**72.2**

3.3. Results

In order to evaluate the proposed model for multi-turn SLU, we compare the performance of following model architectures in Table 1 and Table 2.

- **RNN Tagger** treats each test utterance independently and performs sequence tagging via RNN-GRUs, where the training set comes from single-turn interactions (row (a)) or multi-turn interactions (row (b)).
- **Encoder-Tagger** encodes the knowledge before prediction using RNN-GRUs, and then estimates each tag by considering not only the current input word but also the encoded information via another RNN-GRUs, where we encode knowledge using the current utterance only (row (c)) or entire history together with the current utterance (row (d)). Note that there is no attention and memory mechanisms in this model. The entire history is concatenated together for modeling the knowledge.
- **Memory Network** takes history and current utterances into account for encoding knowledge with attention and memory mechanisms in an end-to-end fashion, and then performs sequence tagging using RNN-GRUs as described in Section 2 (row (e)).

The evaluation metrics are precision (P), recall (R) and F-measure (F1) for semantic tags, where each tagging result is considered correct if the word-beginning and the word-inside and the word-outside predictions are all correct (including both intents and slots). For the evaluation results, we show the performance of the testing set in terms of (1) first turn only, (2) other turns, and (3) all turns from a full dialogue in Table 1. Furthermore, we show the performance of evaluating intents and slots separately in Table 2.

3.3.1. Comparing between Single-Turn and Multi-Turn Data

For the rows (a) and (b) in Table 1, training on single-turn data may work well when testing the first-turn utterances, achieving 60.6% on F1. However, for other turns, its performance is much worse due to lack of modeling contextual knowledge in the multi-turn interactions and mismatch between training and testing data. Treating each utterance from multi-turn interactions independently performs 55.8% on F1 for the first-turn utterances, even though the size of training data is smaller. The reason is probably that there is no mismatch between training and testing.

3.3.2. Comparing between Encoded Knowledge

For employing encoder-tagger models (rows (c) and (d)), additionally encoding the history utterances improves the tagging results for both first-turn and following turns. Also, the encoder-tagger significantly outperform the RNN tagger (47.4% to 62.5%), showing that encoder-tagger is able to capture clues from long-term dependencies.

3.3.3. Effectiveness of the Proposed Model

From Table 1, we find that the best overall performance comes from the proposed memory networks (row (e)), which achieves 67.1% on the overall F1 score and shows the effectiveness of modeling long-term knowledge for SLU. The proposed model works well when tagging the turns with previous contexts, where the F1 score of SLU is about 65.7%. Interestingly, the performance of first-turn utterances is also better than all baselines, probably because the capability of modeling following turns can benefit the performance of first-turn utterances. Comparing with the row (d), memory network is able to effectively capture salient knowledge for better tagging. Also, in terms of efficiency, the proposed model is more than 10x faster than the encoder-tagger model, because each utterance is modeled separately and stored in the memory for later reuse.

3.3.4. Analysis of Intent and Slot Performance

Table 2 presents intent-only and slot-only performance of the same set of models. For both intent and slot performance, the model trained on single-turn utterances performs worst. For intent performance, encoder-tagger with history (row (d)) and memory network (row (e)) significantly outperform others. The difference between the rows (d) and (e) may not be significant. On the other hand, in terms of slot performance, the best result is from the memory network, demonstrating that knowledge carryover modeled by the proposed model can benefit inference of semantic intents and slots in the multi-turn scenario.

4. Conclusions

This paper proposes end-to-end memory networks to store contextual knowledge, which can be exploited dynamically during testing for manipulating knowledge carryover in order to model long-term knowledge for multi-turn understanding. The model embeds history utterances into a continuous space and store them in the memory. The decoding phase applies an attention model to encode the carried knowledge and then perform multi-turn SLU. The experiments show the feasibility and robustness of modeling knowledge carryover through memory networks.

5. References

[1] G. Tur and R. De Mori, *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons, 2011.

[2] Y.-N. Chen, W. Y. Wang, A. Gershman, and A. I. Rudnicky, "Matrix factorization with knowledge graph propagation for unsupervised spoken language understanding," *Proceedings of ACL-IJCNLP*, 2015.

[3] P. Haffner, G. Tur, and J. H. Wright, "Optimizing svms for complex call classification," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP)*, vol. 1. IEEE, 2003, pp. I–632.

[4] C. Chelba, M. Mahajan, and A. Acero, "Speech utterance classification," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP)*, vol. 1. IEEE, 2003, pp. I–280.

[5] Y.-N. Chen, D. Hakkani-Tur, and G. Tur, "Deriving local relational surface forms from dependency-based entity embeddings for unsupervised spoken language understanding," in *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2014, pp. 242–247.

[6] R. Pieraccini, E. Tzoukermann, Z. Gorelov, J.-L. Gauvain, E. Levin, C.-H. Lee, and J. G. Wilpon, "A speech understanding system based on statistical representation of semantics," in *1992 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1. IEEE, 1992, pp. 193–196.

[7] Y.-Y. Wang, L. Deng, and A. Acero, "Spoken language understanding," *IEEE Signal Processing Magazine*, vol. 22, no. 5, pp. 16–31, 2005.

[8] C. Raymond and G. Riccardi, "Generative and discriminative algorithms for spoken language understanding." in *INTERSPEECH*, 2007, pp. 1605–1608.

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

[10] G. Tur, L. Deng, D. Hakkani-Tür, and X. He, "Towards deeper understanding: Deep convex networks for semantic utterance classification," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 5045–5048.

[11] R. Sarikaya, G. E. Hinton, and A. Deoras, "Application of deep belief networks for natural language understanding," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 4, pp. 778–784, 2014.

[12] S. Ravuri and A. Stolcke, "Recurrent neural network and lstm models for lexical utterance classification," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[13] P. Xu and R. Sarikaya, "Convolutional neural network based triangular CRF for joint intent detection and slot filling," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2013, pp. 78–83.

[14] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, "Recurrent neural networks for language understanding." in *INTERSPEECH*, 2013, pp. 2524–2528.

[15] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu *et al.*, "Using recurrent neural networks for slot filling in spoken language understanding," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 530–539, 2015.

[16] D. Hakkani-Tür, G. Tur, A. Celikyilmaz, Y.-N. Chen, J. Gao, D. Li, and Y.-Y. Wang, "Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM," in *INTERSPEECH*, 2016.

[17] A. Bhargava, A. Celikyilmaz, D. Hakkani-Tur, and R. Sarikaya, "Easy contextual intent prediction and slot detection," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 8337–8341.

[18] P. Xu and R. Sarikaya, "Contextual domain classification in spoken language understanding systems using recurrent neural network," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 136–140.

[19] Y.-N. Chen, M. Sun, A. I. Rudnicky, and A. Gershman, "Leveraging behavioral patterns of mobile applications for personalized spoken language understanding," in *Proceedings of 17th ACM International Conference on Multimodel Interaction (ICMI)*. ACM, 2015, pp. 83–86.

[20] M. Sun, Y.-N. Chen, and A. I. Rudnicky, "An intelligent assistant for high-level task understanding," in *Proceedings of The 21st Annual Meeting of the Intelligent Interfaces Community (IUI)*, 2016, pp. 169–174.

[21] Y. Shi, K. Yao, H. Chen, Y.-C. Pan, M.-Y. Hwang, and B. Peng, "Contextual spoken language understanding using recurrent neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5271–5275.

[22] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *International Conference on Learning Representations (ICLR)*, 2015.

[23] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[24] J. Weston, S. Chopra, and A. Bordesa, "Memory networks," in *International Conference on Learning Representations (ICLR)*, 2015.

[25] S. Sukhbaatar, J. Weston, R. Fergus *et al.*, "End-to-end memory networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2431–2439.

[26] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[27] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[29] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[30] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, "Spoken language understanding using long short-term memory neural networks," in *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2014, pp. 189–194.

[31] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 6645–6649.

[32] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[33] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 2342–2350.

[34] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.