Multi-Domain Joint Semantic Frame Parsing using Bi-directional RNN-LSTM

Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Chen†, Jianfeng Gao, Li Deng, and Ye-Yi Wang

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

Abstract

Sequence-to-sequence deep learning has recently emerged as a new paradigm in supervised learning for spoken language understanding. However, most of the previous studies explored this framework for building single domain models for each task, such as slot filling or domain classification, comparing deep learning based approaches with conventional ones like conditional random fields. This paper proposes a holistic multi-domain, multi-task (i.e. slot filling, domain and intent detection) modeling approach to estimate complete semantic frames for all user utterances addressed to a conversational system, demonstrating the distinctive power of deep learning methods, namely bi-directional recurrent neural network (RNN) with long-short term memory (LSTM) cells (RNN-LSTM) to handle such complexity. The contributions of the presented work are three-fold: (i) we propose an RNN-LSTM architecture for joint modeling of slot filling, intent determination, and domain classification; (ii) we build a joint multi-domain model enabling multi-task deep learning where the data from each domain reinforces each other; (iii) we investigate alternative architectures for modeling lexical context in spoken language understanding. In addition to the simplicity of the single model framework, experimental results show the power of such an approach on Microsoft Cortana real user data over alternative methods based on single domain/task deep learning.

Index Terms: recurrent neural networks, long short term memory, multi-domain language understanding, joint modeling

1. Introduction

In the last decade, a variety of practical goal-oriented conversation understanding systems have been built for a number of domains, such as the virtual personal assistants Microsoft's Cortana and Apple's Siri. Three key tasks in such targeted understanding applications are domain classification, intent determination and slot filling [1], aiming to form a semantic frame that captures the semantics of user utterances/queries. Domain classification is often completed first in spoken language understanding (SLU) systems, serving as a top-level triage for subsequent processing. Intent determination and slot filling are then run for each domain to fill a domain specific semantic template. An example semantic frame for a movie-related utterance, *"find recent comedies by James Cameron"*, is shown in Figure 1.

This modular design approach (i.e., modeling SLU as 3 tasks) has the advantage of flexibility; specific modifications (e.g., insertions, deletions) to a domain can be implemented without requiring changes to other domains. Another advantage is that, in this approach, one can use task/domain specific features, which often significantly improve the accuracy of these task/domain specific models. Also, this approach often yields more focused understanding in each domain since the intent determination only needs to consider a relatively small set of intent and slot classes over a single (or limited set) of domains, and model parameters could be optimized for the specific set of intent and slots. However, this approach also has disadvantages: First of all, one needs to train these models for each domain. This is an error-prone process, requiring careful engineering to insure consistency in processing across domains. Also, during run-time, such pipelining of tasks results in transfer of errors from one task to the following tasks. Furthermore, there is no data or feature sharing between the individual domain models, resulting in data fragmentation, whereas, some semantic intents (such as, finding or buying a domain specific entity) and slots (such as, dates, times, and locations) could actually be common to many domains [2, 3]. Finally, the users may not know which domains are covered by the system and to what extent, so this issue results in interactions where the users do not know what to expect and hence resulting in user dissatisfaction [4, 5].

W	find	recent	comedies	by	james	cameron
	↓	↓	↓	↓	↓	↓
S	O	B-date	B-genre	O	B-dir	I-dir
D	movies					
I	find_movie					

Figure 1: *An example utterance with annotations of semantic slots in IOB format (S), domain (D), and intent (I), B-dir and I-dir denote the director name.*

We propose a single recurrent neural network (RNN) architecture that integrates the three tasks of domain detection, intent detection and slot filling for multiple domains in a single SLU model. This model is trained using all available utterances from all domains, paired with their semantic frames. The input of this RNN is the input sequence of words (e.g., user queries) and the output is the full semantic frame, including domain, intent, and slots, as shown in Figure 1. Since the dependency between the words is important for SLU tasks, we investigate alternative architectures for integrating lexical context and dependencies. We compare the single model approach to alternative ways of building models for multi-task, multi-domain scenarios.

The next section sets the baseline RNN-LSTM architecture based on the slot filling task [6], and explores various architectures for exploiting lexical contexts. In Section 3, we extend this architecture to model domains and intents of user utterances in addition to slot filling, and propose a multi-domain multi-task architecture for SLU. In the experiments, we first investigate the performance of alternative architectures on the benchmark ATIS data set [7], and then on the Microsoft Cortana muilti-domain data. We show that the single multi-domain, joint model approach is not only simpler, but also results in the best F-measure

in experimental results.

2. Deep Learning for SLU

A major task in spoken language understanding in goal-oriented human-machine conversational understanding systems is to automatically classify the domain of a user query along with domain specific intents and fill in a set of arguments or "slots" to form a semantic frame. In this study, we follow the popular IOB (in-out-begin) format for representing the slot tags as shown in Figure 1.

A detailed survey of pre-deep learning era approaches for domain detection, intent determination, and slot filling can be found in [1]. Basically, domain detection and intent determination tasks are framed as classification problems, for which researchers have employed support vector machines [8], maximum entropy classifiers [9], or boosting based classifiers [10, 11]. Similarly, slot filling is framed as a sequence classification problem and hidden Markov models [12] and conditional random fields [13, 14] have been employed.

With the advances on deep learning, deep belief networks (DBNs) with deep neural networks (DNNs) have first been employed for intent determination in call centers [15], and later for domain classification in personal assistants [16, 17, 18]. More recently, an RNN architecture with LSTM cells have been employed for intent classification [19].

For slot filling, deep learning research has started as extensions of DNNs and DBNs (e.g., [20]) and is sometimes merged with CRFs [21]. One notable extension is the use of recursive neural networks, framing the problem as semantic parsing [22]. To the best of our knowledge RNNs have first been employed for slot filling by Yao et al. [23] and Mesnil et al. [24] concurrently. We have compiled a comprehensive review of RNN based slot filling approaches in [6].

Especially with the re-discovery of LSTM cells [25] for RNNs, this architecture has started to emerge [26]. LSTM cells are shown to have superior properties, such as faster convergence and elimination of the problem of vanishing or exploding gradients in sequence via self-regularization, as presented below. As a result, LSTM is more robust than RNN in capturing long-span dependencies.

2.1. RNN with LSTM cells for slot filling

To estimate the sequence of tags $Y = y_1, ..., y_n$ corresponding to an input sequence of tokens $X = x_1, ..., x_n$, we use the Elman RNN architecture [27], consisting of an input layer, a hidden layer (for the single layer version), and an output layer. The input, hidden and output layers consist of a set of neurons representing the input, hidden, and output at each time step t, x_t, h_t, and y_t, respectively. The input is typically represented by 1-hot vector or word level embeddings. Given the input layer x_t at time t, and hidden state from the previous time step h_{t-1}, the hidden and output layers for the current time step are computed as follows:

$$h_t = \phi(W_{xh}\begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}) \quad (1)$$

$$p_t = \text{softmax}(W_{hy}h_t) \quad (2)$$

$$\hat{y_t} = \text{argmax}\, p_t \quad (3)$$

where W_{xh} and W_{hy} are the matrices that denote the weights between the input and hidden layers and hidden and output layers, respectively. ϕ denotes the activation function, i.e., `tanh` or `sigm`. The softmax is defined as: $\text{softmax}(z_m) = e^{z_m} / \sum_i e^{z_i}$. The weights of the model are trained using backpropagation to maximize the conditional likelihood of the training set labels:

$$\prod_t p(y_t \mid x_1, ..., x_t). \quad (4)$$

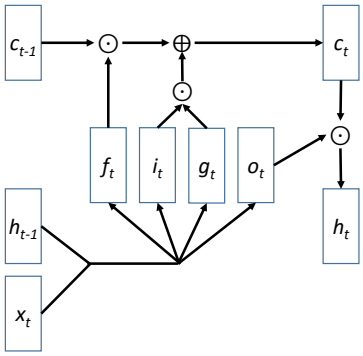

Figure 2: *LSTM cell, as depicted in [31].*

Previous work [28] has shown that training model parameters with backpropagation over time could result in exploding or vanishing gradients. Exploding gradients could be alleviated by gradient clipping [29], but this does not help vanishing gradients. LSTM cells [25] were designed to mitigate the vanishing gradient problem. In addition to the hidden layer vector h_t, LSTMs maintain a memory vector, c_t, which it can choose to read from, write to or reset using a gating mechanism and sigmoid functions. The input gate, i_t is used to scale down the input; the forget gate, f_t is used to scale down the memory vector c_t; the output gate, o_t is used to scale down the output to reach the final h_t. Following the precise formulation of [30], these gates in LSTMs are computed as follows, as also shown in Figure 2:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ g_t \end{bmatrix} = \begin{pmatrix} \texttt{sigm} \\ \texttt{sigm} \\ \texttt{sigm} \\ \texttt{tanh} \end{pmatrix} W_t \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix}, \quad (5)$$

where the `sigm` sand `tanh` are applied element-wise, W_t is the weight matrix, and

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t, \quad (6)$$

$$h_t = o \odot \texttt{tanh}(c_t). \quad (7)$$

2.2. Integration of context

In SLU, word tags are not only determined by the associated terms, but also contexts [32]. For example, in ATIS data, the city name *Boston* could be tagged as originating or destination city, according to the lexical context it appears in. For capturing such dependencies, we investigated two extensions to the RNN-LSTM architecture (Figure 3.(a)): look-around LSTM (LSTM-LA) and bi-directional LSTM (bLSTM) [33].

At each time step, in addition to x_t, LSTM-LA (Figure 3.(b)) considers the following and preceding words as part of the input, by concatenating the input vectors for the neighboring words. In this work, our input at time t consisted of a single vector formed by concatenating x_{t-1}, x_t, x_{t+1}.

In bLSTM (Figure 3.(c)), two LSTM architectures are traversed in a left-to-right and right-to-left manner, and their hid-

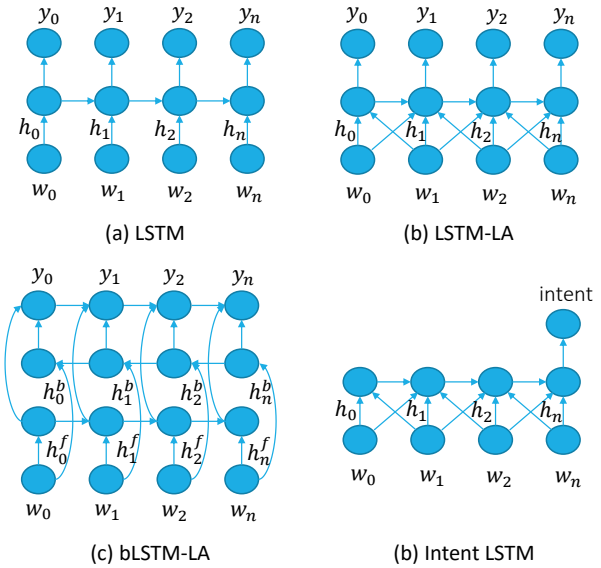

Figure 3: *RNN-LSTM architectures used in this work.*

den layers are concatenated when computing the output sequence (we use the superscripts b and f for denoting parameters for the backward and forward directions):

$$p_t = \text{softmax}(W^f_{hy} h^f_t + W^b_{hy} h^b_t), \tag{8}$$

where forward and backward gates are computed respectively as follows:

$$\begin{bmatrix} i^f_t \\ f^f_t \\ o^f_t \\ g^f_t \end{bmatrix} = \begin{pmatrix} \texttt{sigm} \\ \texttt{sigm} \\ \texttt{sigm} \\ \texttt{tanh} \end{pmatrix} W^f_t \begin{bmatrix} x_t \\ h^f_{t-1} \end{bmatrix}, \tag{9}$$

$$\begin{bmatrix} i^b_t \\ f^b_t \\ o^b_t \\ g^b_t \end{bmatrix} = \begin{pmatrix} \texttt{sigm} \\ \texttt{sigm} \\ \texttt{sigm} \\ \texttt{tanh} \end{pmatrix} W^b_t \begin{bmatrix} x_t \\ h^b_{t+1} \end{bmatrix}. \tag{10}$$

In order to make the implementation more efficient, many of the shared computations are done once such as input vector preparation or top level gradient computation, $p_t - \text{truth}_t$, where truth_t is the 1-hot vector for the target tag.

Figure 3 depicts these three architectures, as well as the intent LSTM architecture of [19] that we used for modeling of intents and domains in isolation as the baseline.

3. Joint, Multi-Domain Modeling of Domain, Intent and Slots

A commonly used approach to represent slot tags for slot filling is associating each input word w_t of utterance k with an IOB-style tag as exemplified in Figure 1, hence the input sequence X is $w_1, ..., w_n$ and the output is the sequence of slot tags $s_1, ..., s_n$. We follow this approach and associate a slot tag with each word.

For joint modeling of domain, intent, and slots, we assume an additional token at the end of each input utterance k, <EOS>, and associate a combination of domain and intent tags d_k and i_k to this sentence final token by concatenating these tags. Hence, the new input and output sequence are :

$$\begin{aligned} X &= w_1, ..., w_n, \text{<EOS>} \\ Y &= s_1, ..., s_n, d_k\_i_k \end{aligned}$$

The main rationale of this idea is similar to the sequence-to-sequence modeling approach, as used in machine translation [34] or chit-chat [35] systems approaches. The last hidden layer of the query is supposed to contain a latent semantic representation of the whole input utterance, so that it can be utilized for domain and intent prediction ($d_k\_i_k$).

4. Experiments

For training all architectures, we used mini-batch stochastic gradient descent with a batch size of 10 examples and adagrad [36]. We experimented with different hidden layer sizes in {50, 75, 100, 125, 150} and a fixed learning rate in {0.01, 0.05, 0.1} in all of the experiments. We used only lexical features (i.e., no dictionaries), and represented input with 1-hot word vectors, including all the vocabulary terms. In addition to 1-hot word vectors, we experimented with word2vec [37] and Senna [38] embeddings, and did not observe significant performance improvement, hence only results with 1-hot vectors are reported. All parameters were uniformly initialized in $[-0.01, 0.01]$.

4.1. Data sets

For investigating the integration of contexts for slot filling, we have experimented with the benchmark ATIS data set [7] for the air travel domain. For experiments related to joint domain, intent, and slot modeling, four domains are chosen: alarm, calendar, communication and technical, to create a diverse set in terms of vocabulary size, number of intents and slots. The number of training, development and test utterances, vocabulary size, number of intents and slots for each of these data sets are listed in Table 4. As seen in the last row of this table, the number of intents and slots in the joined data set is less than the sum of the number of intents and slots in individual domains, this is because some of these are shared across different domains.

4.2. Slot Filling Experiments

ATIS data set comes with a commonly used training and test split [7]. For tuning parameters, we further split the training set into 90% training and 10% development set. After choosing the parameters that maximize the F-measure on the development set, we retrained the model with all of the training data with the optimum parameter set with 10 different initializations and averaged F-measures. The maximum F-measure (best F) is computed on the test set when 90% of the training examples were used and the average F-measure (avg. F) is computed by averaging F-measure from the 10 runs when all the training examples are used with the optimum parameters. These results are shown in Table 2. We get the best F-measure with the bi-directional LSTM architecture (though comparable with LSTM-LA), the relative performances of RNN, LSTM, and LSTM-LA are in parallel with our earlier work [39], though F-measure is slightly lower due to differences in normalization.

4.3. Multi-Domain, Joint Model Experiments

Following the slot filling experiments, we used bi-directional LSTM for modeling slots alone and jointly modeling intent and slots, and following [19], we use LSTM for modeling intents.

Table 1: *Data sets used in the experiments. For each domain, the number of examples in the training, dev, and test sets, input vocabulary size of the training set, and number of unique intents and slots.*

Data Set	# Train	# Dev	# Test	$\|V\|$	# Intents	# Slots
ATIS	4,978	-	893	900	17	79
Alarm	8,096	1,057	846	433	16	8
Calendar	21,695	3,626	2,555	1,832	20	18
Communication	13,779	2,662	1,529	4,336	25	20
Techincal	7,687	993	867	2,180	5	18
4 domains	51,257	8,338	5,797	6,680	59	42

Table 2: *F-measure results using ATIS data. The first column shows the best F-measure on the test set, when the model was trained with 90% of the training examples, the second column shows F-measure averaged over 10 random initializations with parameters optimized in the development set (10%).*

Model	best F	avg. F
RNN	93.06%	92.09%
LSTM	93.80%	93.09%
LSTM-LA	95.12%	94.68%
bLSTM	95.48%	94.70%

We experimented with 4 settings, and report slot F-measure (SLOT F, Table 3), intent accuracy (INTENT A, Table 3 and overall frame error rate (OVERALL E, Table 4) for each of these:

- **SD-Sep**: For each domain, a separate intent detection and slot filling model was trained, resulting in $2 \times |D|$ classifiers, where $|D|$ is the number of domains. Optimum parameters were found on the development set for each experiment and used for computing performance on the test set. The output of all the classifiers were joined for overall error rates.
- **SD-Joint**: For each domain, a single model that estimates both intent and sequence of slots was used, resulting in $|D|$ classifiers.
- **MD-Sep**: An intent detection model and a slot filling model were trained using data from all the domains, resulting in 2 classifiers. The output of intent detection was merged with the output of slot filling for computing overall template error rates.
- **MD-Joint**: A single classifier for estimating the full semantic frame that includes domain, intent, and slots for each utterance was trained using all the data.

The first two settings assume that the correct domain for each example in the test set is provided. To estimate such higher level domain estimation, we trained an LSTM model for domain detection using all the data, the accyracy of the domain detection is 95.5% on the test set. Table 3 shows results for intent detection and slot filling when the true domain is known for the first two settings, hence the performances of these two settings seem higher, however, Table 4 shows overall frame error rates when the domain estimation is integrated in the decision of the final frame. In both single-domain and multi-domain settings, intent detection accuracy improves with joint training (although small), but slot filling degrades. On the overall, we achieve the lowest error with the single model approach. The 13.4% semantic frame error rate on all the data is significantly better than the commonly used SD-Sep.

Table 3: *Slot F-measure and intent accuracy results in single domain (SD) and multi domain (MD) joint and separate modeling experiments.*

SLOT F	SD-Sep	SD-Joint	MD-Sep	MD-Joint
Alarm	95.9%	93.9%	94.5%	94.3%
Cal.	94.5%	93.7%	92.6%	92.4%
Comm.	86.4%	83.8%	85.1%	82.7%
Tech.	90.4%	89.8%	89.6%	88.3%
All	91.8%	90.5%	90.0%	89.4%
INTENT A	**SD-Sep**	**SD-Joint**	**MD-Sep**	**MD-Joint**
Alarm	96.5%	96.2%	94.9%	94.3%
Cal.	97.2%	97.6%	94.2%	94.3%
Comm.	96.1%	95.8%	94.0%	95.4%
Tech.	94.6%	95.9%	93.9%	95.3%
All	96.4%	96.7%	94.1%	94.6%

Table 4: *Overall frame level error rates.*

Overall E	SD-Sep	SD-Joint	MD-Sep	MD-Joint
Alarm	9.5%	9.8%	9.1%	9.2%
Cal.	10.7%	11.1%	11.3%	10.1%
Comm.	19.8%	20.6%	16.3%	17.3%
Tech.	20.4%	20.6%	21.4%	20.2%
All	14.4%	14.9%	13.7%	**13.4%**

5. Conclusions

We propose a multi-domain, multi-task (i.e. domain and intent detection and slot filling) sequence tagging approach to estimate complete semantic frames for user utterances addressed to a conversational system. First, we investigate alternative architectures for modeling lexical context for spoken language understanding. Then we present our approach that jointly models slot filling, intent determination, and domain classification in a single bi-directional RNN with LSTM cells. User queries from multiple domains are combined in a single model enabling multi-task deep learning. We empirically show improvements with the proposed approach in experimental results, over alternatives. In addition to the simplicity of the single model framework for SLU, as our future research, such an architecture opens way to handling belief state update, other non-lexical contexts, such as user contacts or dialogue history in one holistic model [32]. Furthermore, an RNN-LSTM based language generation system [40] can be jointly trained enabling the end-to-end conversational understanding framework.

6. Acknowledgments

Authors would like to thank Nikhil Ramesh, Bin Cao, Derek Liu and reviewers for useful discussions and feedback.

7. References

[1] G. Tur and R. D. Mori, Eds., *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. New York, NY: John Wiley and Sons, 2011.

[2] Y.-B. Kim, K. Stratos, R. Sarikaya, and M. Jeong, "New transfer learning techniques for disparate label sets," in *Proceedings of ACL-IJCNLP*. ACL, 2015.

[3] Y.-N. Chen, D. Hakkani-Tur, and X. He, "Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models," in *Proceedings of ICASSP*. IEEE, 2016.

[4] Y.-N. Chen, W. Y. Wang, and A. I. Rudnicky, "Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing," in *Proceedings of ASRU*. IEEE, 2013, pp. 120–125.

[5] Y.-N. Chen, W. Y. Wang, A. Gershman, and A. I. Rudnicky, "Matrix factorization with knowledge graph propagation for unsupervised spoken language understanding," in *Proceedings of ACL-IJCNLP*. ACL, 2015, pp. 483–494.

[6] 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.

[7] P. J. Price, "Evaluation of spoken language systems: The ATIS domain," in *Proceedings of the DARPA Workshop on Speech and Natural Language*, Hidden Valley, PA, June 1990.

[8] P. Haffner, G. Tur, and J. Wright, "Optimizing SVMs for complex call classification," in *Proceedings of the ICASSP*, Hong Kong, April 2003.

[9] C. Chelba, M. Mahajan, and A. Acero, "Speech utterance classification," in *Proceedings of the ICASSP*, Hong Kong, May 2003.

[10] R. E. Schapire and Y. Singer, "Boostexter: A boosting-based system for text categorization," *Machine Learning*, vol. 39, no. 2/3, pp. 135–168, 2000.

[11] B. Favre, D. Hakkani-Tür, and S. Cuendet, "Icsiboost," http://code.google.come/p/icsiboost, 2007.

[12] 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 *Proceedings of the ICASSP*, San Francisco, CA, March 1992.

[13] Y.-Y. Wang, L. Deng, and A. Acero, "Spoken language understanding - an introduction to the statistical framework," *IEEE Signal Processing Magazine*, vol. 22, no. 5, pp. 16–31, September 2005.

[14] C. Raymond and G. Riccardi, "Generative and discriminative algorithms for spoken language understanding," in *Proceedings of the Interspeech*, Antwerp, Belgium, 2007.

[15] R. Sarikaya, G. E. Hinton, and B. Ramabhadran, "Deep belief nets for natural language call-routing," in *Proceedings of the ICASSP*, Prague, Czech Republic, 2011.

[16] G. Tur, L. Deng, D. Hakkani-Tür, and X. He, "Towards deeper understanding deep convex networks for semantic utterance classification," in *In Proceedings of the ICASSP*, Kyoto, Japan, March 2012.

[17] L. Deng, G. Tur, X. He, and D. Hakkani-Tür, "Use of kernel deep convex networks and end-to-end learning for spoken language understanding," in *In Prooceedings of the IEEE SLT Workshop*, Miami, FL, December 2012.

[18] R. Sarikaya, G. E. Hinton, and A. Deoras, "Application of deep belief networks for natural language understanding," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 4, April 2014.

[19] S. Ravuri and A. Stolcke, "Recurrent neural network and LSTM models for lexical utterance classification," in *Interspeech*, 2015.

[20] A. Deoras and R. Sarikaya, "Deep belief network based semantic taggers for spoken language understanding," in *In Prooceedings of the Interspeech*, Lyon, France, August 2013.

[21] P. Xu and R. Sarikaya, "Convolutional neural network based triangular CRF for joint intent detection and slot filling," in *Proceedings of the IEEE ASRU*, 2013.

[22] D. Guo, G. Tur, W.-T. Yih, and G. Zweig, "Joint semantic utterance classification and slot filling with recursive neural networks," in *Proceedings of the IEEE SLT Workshop*, 2014.

[23] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, "Recurrent neural networks for language understanding," in *In Prooceedings of the Interspeech*, Lyon, France, August 2013.

[24] G. Mesnil, X. He, L. Deng, and Y. Bengio, "Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding," in *Proceedings of Interspeech*, 2013.

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

[26] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, "Spoken language understanding using long short-term memory neural networks," in *Proceedings of the IEEE SLT Workshop*, 2014.

[27] J. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, 1990.

[28] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[29] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *arXiv preprint arXiv:1211.5063*, 2012.

[30] A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and understanding recurrent networks," *arXiv preprint arXiv:1506.02078*, November 2015.

[31] 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.

[32] Y.-N. Chen, D. Hakkani-Tür, G. Tur, J. Gao, and D. Li, "End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding," in *Proceedings of Interspeech*, 2016.

[33] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[34] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., 2014, pp. 3104–3112.

[35] O. Vinyals and Q. V. Le, "A neural conversational model," in *ICML Deep Learning Workshop*, 2015.

[36] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, no. 12, pp. 2121–2159, 2011.

[37] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Workshop at ICLR*, 2013.

[38] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, no. 12, pp. 2483–2537, 2011.

[39] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu, and G. Zweig, "Using recurrent neural networks for slot filling in spoken language understanding," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, March 2015.

[40] T.-H. Wen, M. Gasic, N. Mrksic, P.-H. Su, D. Vandyke, and S. Young, "Semantically conditioned LSTM-based natural language generation for spoken dialogue systems," *arXiv preprint arXiv:1508.01745*, 2015.