Learning Dialogue History for Spoken Language Understanding

Xiaodong Zhang, Dehong Ma, and Houfeng Wang(✉)

Institute of Computational Linguistics, Peking University, Beijing 100871, China
{zxdcs,madehong,wanghf}@pku.edu.cn

Abstract. In task-oriented dialogue systems, spoken language understanding (SLU) aims to convert users' queries expressed by natural language to structured representations. SLU usually consists of two parts, namely intent identification and slot filling. Although many methods have been proposed for SLU, these methods generally process each utterance individually, which loses context information in dialogues. In this paper, we propose a hierarchical LSTM based model for SLU. The dialogue history is memorized by a turn-level LSTM and it is used to assist the prediction of intent and slot tags. Consequently, the understanding of the current turn is dependent on the preceding turns. We conduct experiments on the NLPCC 2018 Shared Task 4 dataset. The results demonstrate that the dialogue history is effective for SLU and our model outperforms all baselines.

Keywords: Spoken language understanding · Dialogue history
Hierarchical LSTM

1 Introduction

In recent years, task-oriented dialogue systems have a rapid development and widespread application, e.g., voice assistant in mobiles and intelligent customer service. Spoken language understanding (SLU) plays an import role in task-oriented dialogue systems. An utterance of a user is often first transcribed to text by an automatic speech recognizer (ASR). The SLU then interprets the meaning of the utterance and convert the unstructured text to structured representations. The result of SLU is passed to dialogue management module to update dialogue state and make dialogue policy. Therefore, the performance of SLU is critical to a task-oriented dialogue system.

SLU usually consists of two parts, namely intent identification and slot filling. Intent identification can be viewed as an utterance classification problem, and slot filling can be viewed as a sequence labeling problem. Take the utterance in Table 1 as an example. The intent of the utterance is *navigation*. There are two slots in the utterance, i.e., the origin *the Forbidden City* and the destination *Peking University*. With the IOB (Inside, Outside, Beginning) annotation

M. Zhang et al. (Eds.): NLPCC 2018, LNAI 11108, pp. 120–132, 2018.
https://doi.org/10.1007/978-3-319-99495-6_11

Table 1. An example utterance for SLU.

Utterance	Go	to	Peking	University	from	the	Forbidden	City
Slots	O	O	B-dest	I-dest	O	B-orig	I-orig	I-orig
Intent	Navigation							

method, the slot labels for each word are listed in Table 1. The category of intent and slot is usually defined by domain experts.

For intent identification, lots of classifiers have been used by previous work, including support vector machines (SVM) [5], Adaboost [22], and convolutional neural networks (CNN) [24]. For slot filling, the popular methods include maximum entropy Markov models (MEMM) [12], conditional random fields (CRF) [17], and recurrent neural networks (RNN) [13]. In consideration of intent identification and slot filling are correlative, recent works have focused on the joint model of the two tasks [4, 10, 26].

Previous work mainly processes each utterance individually, which does not make use of dialogue context. In task-oriented dialogue systems, dialogue history is important to the understanding of the current utterance. As is shown in Table 2, the third turns in Dialogue 1 and 2 are both "Cancel". If only looking at this turn, it is impossible to identify whether the intent is to *cancel navigation* or *cancel phone call*. The same problem occurs in slot filling. If only looking at the second turns in Dialogue 3 and 4, it is hard to tell whether "Washington" is a name of a place or a person. Fortunately, with the preceding turns, it is possible to make the correct predictions. Therefore, how to model and use dialogue history for SLU is worth of research.

Table 2. Some sample dialogues.

ID	Dialogue
1	A: Go to Peking University B: Planning route to Peking University A: Cancel
2	A: Call Tom B: Calling Tom for you A: Cancel
3	A: Where are you going? B: Washington
4	A: Who would you like to call? B: Washington

For modeling dialogue history, Serban et al. propose a hierarchical recurrent encoder-decoder (HRED) model, which uses a high-level context RNN to keep track of past utterances [19]. However, HRED is only designed for non-task-oriented conversational dialogue systems. Following HRED, we propose a hierarchical long short-term memory based model for SLU (HLSTM-SLU). In HLSTM-SLU, a low-level LSTM is used to learn inner-turn representations. Max-pooling is used to obtain a fixed-length vector of a turn. A high-level LSTM reads the vectors iteratively and memorize useful dialogue history. The contextual representation of a turn is used in two aspects. On one hand, it is utilized to make prediction for the intent of the current turn. One the other hand, it is concatenated with low-level representation to predict the slot label for each word. Consequently, the dialogue history is available for both intent identification and slot filling and contributes to the two tasks.

The rest of the paper is organized as follows. The related work is surveyed in Sect. 2. In Sect. 3, we introduce the proposed HLSTM-SLU model. Section 4 discusses the experimental setup and results on the NLPCC dataset. The conclusion is given in Sect. 5.

2 Related Work

The study of SLU surged in the 1990s with the Air Travel Information System (ATIS) project [16]. The early systems are basically rule-based [23]. Developers are required to write quantities of syntactic or semantic grammars. Due to the informality of spoken language and ASR errors, rule-based methods have run into the bottleneck and data-driven statistical approaches have become the mainstream.

For intent identification, word n-grams are typically used as features with generic entities, such as dates, locations. Many classifiers have been used by prior work, such as SVM [5] and Adaboost [22]. Some researchers tried to use syntactic information. Hakkani et al. presented an approach populating heterogeneous features from syntactic and semantic graphs of utterance for call classification [6]. Tur et al. proposed a dependency parsing based sentence simplification approach that extracts a set of keywords and uses those in addition to entire utterances for completing SLU tasks [22]. For slot filling, Moschitti et al. employed syntactic features via syntactic tree kernels with SVM [15]. Many works used CRF because it is a well-performed model for sequence labelling [17]. Jeong and Lee proposed triangular CRF, which coupled an additional random variable for intent on top of a standard CRF [8]. Mairesse et al. presented an efficient technique that learned discriminative semantic concept classifiers whose output was used to recursively construct a semantic tree, resulting in both slot and intent labels [11].

Recently, various deep learning models have been explored in SLU. The initial try is deep belief networks (DBN), which have been used in call routing classification [18] and slot filling [3]. Tur et al. used deep convex networks (DCN) for domain classification and produced higher accuracy than a boosting-based classifier [21]. RNN has shown excellent performance on slot filling and outperformed CRF [13,14]. Yao et al. improved RNN by using transition features and the sequence-level optimization criterion of CRF to explicitly model dependencies of output labels [25]. Many joint models are proposed to leverage the correlation of the two tasks. Xu and Sarikaya improved the triangular CRF model by using convolutional neural networks (CNN) to extract features automatically [24]. Zhang and Wang used representations learned by Grated Recurrent Unit (GRU) to predict slot labels and used a vector obtained by max-pooling of these representations to predict intent labels [26]. Liu and Lane proposed a conditional RNN model that can be used to jointly perform SLU and language modeling [10].

To model dialogue history, Serban et al. propose a HRED model, which uses a high-level context RNN to keep track of past utterances [19]. Furthermore, Serban et al. extended HRED by introducing Variational Autoencoder (VAE) and proposed VHRED model [20]. VHRED can generate more meaningful and diverse responses. HRED and VHRED are both used in conversational dialogue systems, while our propose HLSTM-SLU is designed for task-oriented dialogue systems.

3 The Proposed Method

In this section, we describe the HLSTM-SLU model in detail. Due to the dataset used in the paper is a Chinese dataset, we use Chinese dialogues as the input to introduce our model. The model can be applied to English dialogues with minor modifications. We consider a dialogue $D = \{T_1, \cdots, T_m\}$ as a sequence of m turns. Each T_i is a sequence of n Chinese characters, i.e., $T_i = \{c_{i,1}, \cdots, c_{i,n}\}$. The outputs of the model for T_i are intent label $\hat{y}_i^T$ and slot labels $\hat{y}_i^c = \{\hat{y}_{i,1}^c, \cdots, \hat{y}_{i,n}^c\}$. An overview of the model is illustrated in Fig. 1.

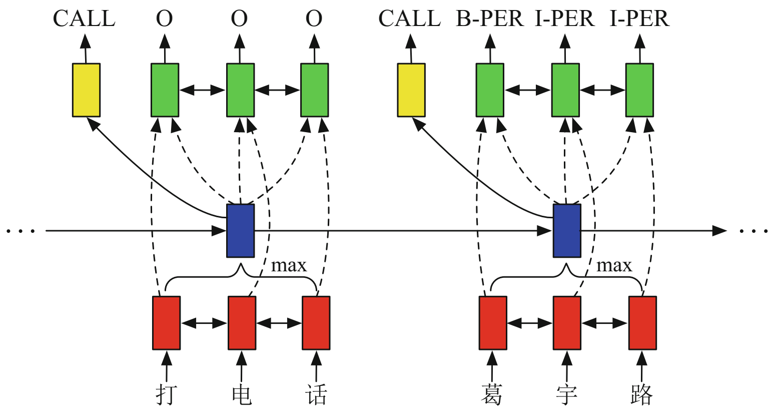

Fig. 1. The structure of HLSTM-SLU. (Color figure online)

3.1 Inputs

HLSTM-SLU processes Chinese text at the character level. The input of the model contains two parts, i.e., characters and additional features. The j^{th} character in the i^{th} turn $c_{i,j}$ is mapped to a low-dimensional dense vector $e_{i,j} \in \mathbb{R}^{d_e}$ via a lookup table, where d_e is the dimension of character embeddings. The additional feature of $c_{i,j}$ is a 0–1 sparse vector $a_{i,j} \in \mathbb{R}^{d_a}$, where d_a is the total number of features. We use two kinds of features, namely part-of-speech and domain lexicons. Domain lexicons contain collected words for several domains, e.g., singers, songs, and so on.

Table 3 lists feature vectors of an example utterance " 小苹果 " (little apple). Feature "B-A" and "I-A" denote the beginning and following characters of adjectives respectively. Similarly, "B-N" and "I-N" are the beginning and following characters of nouns, and "B-song" and "I-song" are the matching result of the song lexicon. The omitted features are all 0.

Table 3. An example of additional feature vectors.

	...	B-A	I-A	B-N	I-N	...	B-song	I-song	...
小	...	1	0	0	0	...	1	0	...
苹	...	0	0	1	0	...	0	1	...
果	...	0	0	0	1	...	0	1	...

The sparse feature vector is convert to a low-dimensional dense vector via a linear transformation. Formally, the dense feature vector $\overline{a}_{i,j}$ for $a_{i,j}$ is computed by

$$\overline{a}_{i,j} = W_a a_{i,j} \tag{1}$$

where $W_a \in \mathbb{R}^{d_{\overline{a}} \times d_a}$ is the transformation matrix and $d_{\overline{a}}$ is the dimension of the dense feature vector.

The input $x_{i,j} \in \mathbb{R}^{d_e + d_{\overline{a}}}$ for LSTM layer is the concatenation of the character embedding and the dense representation of features, i.e.,

$$x_{i,j} = [c_{i,j}; \overline{a}_{i,j}] \tag{2}$$

where $[\cdot;\cdot]$ denotes concatenation of two vectors. With this representation as the input, the LSTM can obtain lexical information to cover the shortage of sequence modeling at the character level.

3.2 Hierarchical LSTM

RNN is a family of neural network that can process variable-length sequences. It uses a recurrent hidden state to take into account the influence of past states. Concretely, it takes a sequence of vectors $X = \{x_1, \cdots, x_n\}$ as input and outputs a sequence $H = \{h_1, \cdots, h_n\}$ that is the representation of the sequence at each time step.

It was hard to for the vanilla RNN to capture long-term dependencies because the gradients tend to vanish or explode [1]. Some more sophisticated activation functions with gating units were designed, among which the most representative variant is long short-term memory (LSTM) [7]. LSTM uses several gates to control the proportion of the input and the proportion of the previous state to forget. Formally, the computation of LSTM is as follows:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{3}$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{4}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{5}$$
$$\hat{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{6}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t \tag{7}$$
$$h_t = o_t \odot tanh(c_t) \tag{8}$$

where i_t, f_t, o_t are input gate, forget gate and output gate respectively, σ is a sigmoid function, W_i, W_f, W_o, W_c, U_i, U_f, U_o, U_c are weight matrices, b_i, b_f, b_c are biases, and x_t is the input at the time step t.

A bidirectional LSTM consists of a forward and a backward LSTM. The forward LSTM reads the input sequence as it is ordered and calculates forward hidden states $(\overrightarrow{h}_1, ..., \overrightarrow{h}_n)$. The backward LSTM reads the sequence in the reserve order and calculates backward hidden states $(\overleftarrow{h}_1, ..., \overleftarrow{h}_n)$. The bidirectional hidden state is obtained by concatenating the forward and backward hidden state, i.e.,

$$h_t = [\overrightarrow{h}_t; \overleftarrow{h}_t] \tag{9}$$

A dialogue can be considered as a sequence at two levels, i.e., a sequence of characters for each turn and a sequence of turns. Intuitively, HLSTM-SLU models a dialogue with two LSTMs: one at the character level and one at the turn level. The character-level LSTM is bidirectional. The hidden states $h^c_{i,j}$ for the j^{th} character in the i^{th} turn is calculated by

$$h^c_{i,j} = BLSTM(x_{i,j}, h^c_{i,j-1}, h^c_{i,j+1}) \tag{10}$$

where $BLSTM(x_t, h_{t-1}, h_{t+1})$ is an abbreviation for Eq. (9), and $x_{i,j}$ is as described in Eq. (2). The character-level LSTM is illustrated by the red part in Fig. 1.

Then, a fixed-length representation for the i^{th} turn is obtained by the max-pooling of all hidden states in the turn, i.e.,

$$r^T_i = \max_{k=1}^{n} h^c_{i,k} \tag{11}$$

where the max function is an element-wise function, and n is the number of characters in the turn.

The turn-level LSTM is unidirectional. This is because only preceding turns are available in an online system. Therefore, the representation of the i^{th} turn with dialogue history r_i^D is calculated by

$$r_i^D = LSTM(r_i^T, r_{i-1}^D) \tag{12}$$

where $LSTM(x_t, h_{t-1})$ is an abbreviation for Eqs. (3)–(8). The turn-level LSTM is illustrated by the blue part in Fig. 1. In this way, r_i^D can learn important information from preceding turns, which is useful for making predictions at the current turn.

3.3 Intent and Slot Tagger

The r_i^D is used for predicting intent labels, which is illustrated by the yellow part in Fig. 1. Instead of making individual prediction for each turn, we select the best label sequence of all turns with a CRF, which has demonstrated good performance on many sequence labeling tasks [2,9]. The CRF layer is not illustrated in Fig. 1 for simplification. Concretely, for a dialogue $D = \{T_1, \cdots, T_m\}$, the unnormalized score for intent tags at T_i is calculated by a fully-connected network with r_i^D as input, i.e.,

$$s_i^T = W_s^T r_i^D + b_s^T \tag{13}$$

where W_s^T is a weight matrix, b_s^T is a bias vector, s_i^T is the score vector and $s_{i,k}^T$ is the score of the k^{th} intent tag for the i^{th} turn.

For a sequence of tags $y^T = \{y_1^T, \cdots, y_m^T\}$, the score of the sequence is

$$S(y^T) = \sum_{i=1}^{m} s_{i,y_i^T}^T + \sum_{i=1}^{m-1} A_{y_i^T, y_{i+1}^T} \tag{14}$$

where A is a matrix of transition scores for intent tags and $A_{j,k}$ represents the transition score from the tag j to tag k.

The probability for the sequence y^T is computed by a softmax over all possible tag sequences, i.e.,

$$p(y^T) = \frac{e^{S(y^T)}}{\sum_{\tilde{y}^T \in Y^T} e^{S(\tilde{y}^T)}} \tag{15}$$

where Y^T is all possible intent tag sequences.

During training, the objective is to maximize the log-probability of the correct tag sequence. During decoding, the predicted intent tag sequence is the one with the maximum score:

$$\hat{y}^T = \underset{\tilde{y}^T \in Y^T}{\operatorname{argmax}} S(\tilde{y}^T) \tag{16}$$

Although all possible sequences are enumerated in Eqs. (15) and (16), they can be computed efficiently using dynamic programming.

As for slot filling, to make use of the dialogue history, the model rereads a turn with the contextual turn representation, which is illustrated by the green part in Fig. 1. Concretely, for a turn T_i, the hidden states of the character-level LSTM

$h^c_{i,j}$ and the turn representation with dialogue history r^D_i, another bidirectional LSTM is used to learn the character representation with dialogue history $h^d_{i,j}$. Formally, we have

$$h^d_{i,j} = BLSTM(x^d_{i,j}, h^d_{i,j-1}, h^d_{i,j+1}) \tag{17}$$

$$x^d_{i,j} = [h^c_{i,j}; r^D_i] \tag{18}$$

We also use the CRF layer for predict slot tags for each turn. The calculation is similar to Eqs. (13)–(16) and we describe it briefly. For each turn T_i, we have

$$s^c_{i,j} = W^c_s h^d_{i,j} + b^c_s \tag{19}$$

$$S(y^c_i) = \sum_{j=1}^{n} s^c_{i,j,y^c_{i,j}} + \sum_{i=1}^{n-1} B_{y^c_{i,j}, y^c_{i,j+1}} \tag{20}$$

$$p(y^c_i) = \frac{e^{S(y^c_i)}}{\sum_{\tilde{y}^c_i \in Y^c_i} e^{S(\tilde{y}^c_i)}} \tag{21}$$

$$\hat{y}^c_i = \underset{\tilde{y}^c_i \in Y^c_i}{\text{argmax}}\, S(\tilde{y}^c_i) \tag{22}$$

The loss function of the model is the sum of negative log-probability of the correct tag sequence for both intent and slot, i.e.,

$$L = -\frac{1}{m}\left(\log(p(y^T)) + \sum_{i=1}^{m} \log(p(y^c_i))\right) \tag{23}$$

4 Experiments

4.1 Dataset

We use NLPCC 2018 Shared Task 4 dataset[1] in our experiment. The dataset is a sample of the real query log from a Chinese commercial task-oriented dialog system. It includes three domains, namely *music*, *navigation* and *phone call*, and an additional domain *OTHERS*, which is the data not covered by the three domains. The dataset contains 4705 dialogues with 21352 turns in the training set and 1177 dialogues with 5350 turns in the testing set. The statistics of the intents and slots in the dataset is listed in Table 4.

We use two metrics given by the task organizer. The first metric is macro F1 of intents and the second one is the precision of both intents and slots. The detailed equations can be found at the guideline[2].

[1] http://tcci.ccf.org.cn/conference/2018/taskdata.php.
[2] http://tcci.ccf.org.cn/conference/2018/dldoc/taskgline04.pdf.

Table 4. The number of intents and slots in the NLPCC dataset.

Intent	Train	Test	Slot	Train	Test
music.play	6425	1641	Song	3941	983
			Singer	1745	473
			Theme	191	45
			Style	69	26
			Age	48	15
			Toplist	44	16
			Emotion	38	2
			Language	29	2
			Instrument	14	7
			Scene	7	0
music.pause	300	75	-		
music.prev	5	4	-		
music.next	132	34	-		
navigation.navigation	3961	1039	Destination	3805	1014
			custom_destination	132	17
			Origin	14	9
navigation.open	245	56	-		
navigation.start_navigation	33	4	-		
navigation.cancel_navigation	835	206	-		
phone_call.make_a_phone_call	2796	674	phone_num	1100	256
			contact_name	779	224
phone_call.cancel	22	18	-		
OTHERS	6598	1599	-		
Total	21352	5350		11956	3089

4.2 Experimental Setup

We compare our model with the following baselines:

- **BLSTM** A bidirectional LSTM model with only character embeddings as input.
- **BLSTM-FT:** A bidirectional LSTM model with character embeddings and additional features as input.
- **BLSTM-CRF:** Compared to BLSTM-FT, it adds a CRF layer for decoding slot tag sequences.
- **HLSTM-Intent:** The difference between HLSTM-Intent and the proposed HLSTM-SLU is that it only uses dialogue history for intent identification, but not for slot filling.

We only use the official provided data for training without using additional data. We shuffle the data randomly at dialogue granularity and set apart

10% dialogues as development set. All hyper-parameters are tuned on the development set. Jieba[3] is used for part-of-speech tagging. The neural networks are implemented using TensorFlow. The dimension of character embeddings and dense feature vector are both 100. The character embeddings are pretrained by GloVe[4] with a large unlabeled Chinese corpus. The dimension of character-level LSTM is 300 (each direction is 150), and the dimension of turn-level LSTM is 300. Models are trained using Adam optimization method using the learning rate set to 0.001.

As shown in Table 4, the data is imbalanced for different intents, which does harm to the performance of the model. We solve the problem from two aspects. First, we use over sampling method. The dialogues that contains the scarce intents are copied for many times. Second, we write some simple rules to identify some intents.

To correct ASR errors, we perform error corrections for song slot using the song lexicon in the dataset. If a song predicted by the model is not found in the lexicon, we try to find a song from the lexicon that has the same pinyin with the predicted song. If still not found, we try to find a song of which the Levenshtein distance with the predicted song is equal to 1. If multiple candidates exist, we randomly select one.

4.3 Experimental Results

The results are demonstrated in Table 5. We can see that the BLSTM preforms worst on both metrics. With additional features, the BLSTM-FT outperforms BLSTM by a large margin. This is because the part-of-speech and lexicon matching information is useful for predicting slot tags. The BLSTM-CRF improves the results further. The reason is that the CRF layer model the transition of tags explicitly, which reduces the situation of incomplete or redundant slot values and illegal tag sequences. HLSTM-Intent uses dialogue history for intent identification, which improves the score of intent significantly. Our HLSTM-SLU model uses dialogue history not only for intent identification but also for slot filling and obtains the best performance on the two metrics. The results support our argument that the dialogue history is important for the two tasks in SLU.

4.4 Case Study

We give the result of three models for an example dialogue to show the difference of the models intuitively. The dialogue is selected from the testing set. It contains two turns, as follows.

- **Turn 1:** 蓝牙打电话 (Make a call with Bluetooth.)
- **Turn 2:** 张解 (Zhang Jie.)

[3] https://github.com/fxsjy/jieba.
[4] https://nlp.stanford.edu/projects/glove/.

Table 5. Results of HLSTM-SLU and baselines.

Method	F1 for Intent	Precision for intent & Slot
BLSTM	88.56	83.38
BLSTM-FT	89.24	86.95
BLSTM-CRF	89.31	88.62
HLSTM-Intent	93.61	90.21
HLSTM-SLU	94.19	90.84

The results of three models are demonstrated in Table 6. For BLSTM-CRF, the Turn 2 is misclassified as *OTHERS*. It is because the BLSTM-CRF process each turn individually. The obvious clue of making a phone call in Turn 1 is not utilized when processing Turn 2. The HLSTM-Intent uses dialogue history for intent prediction and therefore it predicts the intent of Turn 2 correctly. However, it does not identify the contact name. The HLSTM-SLU uses dialogue history for both tasks and the prediction is totally correct.

Table 6. The comparative result with baselines on an example dialogue.

Model	**Dialogue**	**Intent**	**Slot**
BLSTM-CRF	Turn 1	phone_call.make_a_phone_call	-
	Turn 2	OTHERS	-
HLSTM-Intent	Turn 1	phone_call.make_a_phone_call	-
	Turn 2	phone_call.make_a_phone_call	-
HLSTM-SLU	Turn 1	phone_call.make_a_phone_call	-
	Turn 2	phone_call.make_a_phone_call	contact_name = 张解

5 Conclusion

Dialogue history provides import information for SLU in dialogues. In this paper, we propose a HLSTM-SLU model to represent dialogue history and use it for SLU. The HLSTM-SLU uses a character-level LSTM to learn inner-turn representation. Then, the dialogue history is memorized by a turn-level LSTM and it is used to assist the prediction of intent and slot tags. The experimental results demonstrate that the dialogue history is effective for SLU and our model outperforms all baselines. In future work, we will test HLSTM-SLU on English datasets to investigate the generalization of the model.

Acknowledgments. Our work is supported by National Natural Science Foundation of China (No. 61433015).

References

1. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994)
2. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**(Aug), 2493–2537 (2011)
3. Deoras, A., Sarikaya, R.: Deep belief network based semantic taggers for spoken language understanding. In: INTERSPEECH, pp. 2713–2717 (2013)
4. Guo, D., Tur, G., Yih, W.t., Zweig, G.: Joint semantic utterance classification and slot filling with recursive neural networks. In: SLT, pp. 554–559. IEEE (2014)
5. Haffner, P., Tur, G., Wright, J.H.: Optimizing SVMs for complex call classification. In: ICASSP, vol. 1, pp. 632–635. IEEE (2003)
6. Hakkani-Tür, D., Tur, G., Chotimongkol, A.: Using syntactic and semantic graphs for call classification. In: Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing (2005)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
8. Jeong, M., Lee, G.G.: Triangular-chain conditional random fields. IEEE Trans. Audio Speech Lang. Process. **16**(7), 1287–1302 (2008)
9. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. In: NAACL-HLT, pp. 260–270 (2016)
10. Liu, B., Lane, I.: Joint online spoken language understanding and language modeling with recurrent neural networks. In: SIGDIAL, p. 22 (2016)
11. Mairesse, F., et al.: Spoken language understanding from unaligned data using discriminative classification models. In: ICASSP, pp. 4749–4752. IEEE (2009)
12. McCallum, A., Freitag, D., Pereira, F.C.: Maximum entropy Markov models for information extraction and segmentation. In: lCML, vol. 17, pp. 591–598 (2000)
13. Mensil, G., et al.: Using recurrent neural networks for slot filling in spoken language understanding. IEEE/ACM Trans. Audio Speech Lang. Process. **23**(3), 530–539 (2015)
14. Mesnil, G., He, X., Deng, L., Bengio, Y.: Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In: INTERSPEECH, pp. 3771–3775 (2013)
15. Moschitti, A., Riccardi, G., Raymond, C.: Spoken language understanding with Kernels for syntactic/semantic structures. In: ASRU, pp. 183–188. IEEE (2007)
16. Price, P.J.: Evaluation of spoken language systems: the ATIS domain. In: Speech and Natural Language (1990)
17. Raymond, C., Riccardi, G.: Generative and discriminative algorithms for spoken language understanding. In: Eighth Annual Conference of the International Speech Communication Association (2007)
18. Sarikaya, R., Hinton, G.E., Ramabhadran, B.: Deep belief nets for natural language call-routing. In: ICASSP, pp. 5680–5683. IEEE (2011)
19. Serban, I.V., Sordoni, A., Bengio, Y., Courville, A.C., Pineau, J.: Building end-to-end dialogue systems using generative hierarchical neural network models. AAAI **16**, 3776–3784 (2016)
20. Serban, I.V., et al.: A hierarchical latent variable encoder-decoder model for generating dialogues. In: AAAI, pp. 3295–3301 (2017)
21. Tur, G., Deng, L., Hakkani-Tür, D., He, X.: Towards deeper understanding: deep convex networks for semantic utterance classification. In: ICASSP, pp. 5045–5048. IEEE (2012)

22. Tur, G., Hakkani-Tür, D., Heck, L., Parthasarathy, S.: Sentence simplification for spoken language understanding. In: ICASSP, pp. 5628–5631. IEEE (2011)
23. Ward, W., Issar, S.: Recent improvements in the CMU spoken language understanding system. In: Proceedings of the Workshop on Human Language Technology, pp. 213–216 (1994)
24. Xu, P., Sarikaya, R.: Convolutional neural network based triangular CRF for joint intent detection and slot filling. In: ASRU, pp. 78–83. IEEE (2013)
25. Yao, K., Peng, B., Zweig, G., Yu, D., Li, X., Gao, F.: Recurrent conditional random field for language understanding. In: ICASSP, pp. 4077–4081. IEEE (2014)
26. Zhang, X., Wang, H.: A joint model of intent determination and slot filling for spoken language understanding. In: IJCAI, pp. 2993–2999 (2016)