Deep Contextual Language Understanding in Spoken Dialogue Systems

Chunxi Liu[1]*, Puyang Xu*[2]*, Ruhi Sarikaya*[2]

[1]The Johns Hopkins University, Baltimore, MD USA
[2]Microsoft Corporation, Redmond, WA USA
chunxi@jhu.edu, {puyangxu, ruhi.sarikaya}@microsoft.com

Abstract

We describe a unified multi-turn multi-task spoken language understanding (SLU) solution capable of handling multiple context sensitive classification (intent determination) and sequence labeling (slot filling) tasks simultaneously. The proposed architecture is based on recurrent convolutional neural networks (RCNN) with shared feature layers and globally normalized sequence modeling components. The temporal dependencies within and across different tasks are encoded succinctly as recurrent connections. The dialog system responses beyond SLU component are also exploited as effective external features. We show with extensive experiments on a number of datasets that the proposed joint learning framework generates state-of-the-art results for both classification and tagging, and the contextual modeling based on recurrent and external features significantly improves the context sensitivity of SLU models.

Index Terms: convolutional neural networks, recurrent neural networks, spoken language understanding

1. Introduction

Building intelligent spoken dialog systems, which automatically and coherently interact with people using natural languages, has been a grand challenge for both academic research and commercial applications. There has been significant progress in this area as spoken dialog systems with various capabilities are already deployed in Siri, Google Now and Cortana and have been used by millions of users everyday. A typical dialog system is designed to execute the following components: (i) automatic speech recognition (ASR) converts a spoken query into transcription, (ii) spoken language understanding (SLU) component analyzes the transcription to extract semantic representations, (iii) dialog manager (DM) interprets the semantic information and decides the best system action, according to which the system response is further generated either as a natural language output or a result page [1].

In this paper, we consider SLU as the scope of our work, which typically involves several well-established components. In a typical SLU design, domain determination is implemented to classify a user query into one of the supported domains, followed by a more refined domain-specific semantic analysis such as intent detection, and slot filling for extracting entities [2]. Standard data-driven statistical modeling approaches have been proven effective for these tasks. For domain and intent classification, support vector machines (SVMs) [3] are heavily used. Slot filling is often formulated as a sequence labeling problem for which conditional random fields (CRFs) [4] have been shown to generate robust and competitive results.

This work was done while the first author was interning at Microsoft.

One key motivation for our work is that, the task completion with spoken dialog systems tends to span multiple turns, and the number of back and forth (i.e. turns) between user and system increases as the complexity of the scenarios grows. Accurate language understanding (i.e. domain, intent, slots) often requires reasoning from its previous conversational history, to which we refer as context. However, conventional SLU processing for a given query often takes into account only the current turn, failing to consider the contextual information, which may result in incorrect interpretation of the user's request. Below are two example scenarios:

Turn 1: what's the weather like today?
Turn 2: and for the weekend?
Turn 1: postpone my meeting with John.
Turn 2: from 2 pm to 4 pm.

In the first two-turn conversation session, the second turn query is supposed to carry over the domain and intent from the previous turn, which is not possible to achieve without context sensitive models. In the second session, the slot interpretation of 2 pm and 4 pm depends entirely on the previous user intent – whether the user requested to schedule an appointment or change an appointment at the first turn yields distinct meanings for the time entities in the second turn.

In many spoken dialog systems, DM plays the central role in resolving such contextual ambiguity. It is usually formulated as the task of dialog state tracking [5] in the literature. The goal of the task is to exploit the comprehensive dialog context to determine the best summarization of the dialogue status, based on which a proper system response can be generated. We want to demonstrate in this work that many of the same contextual signals can be utilized at the SLU level as well. Due to its role as part of the initial processing components, an SLU error early in the pipeline such as domain or intent classification error, often leads to completely wrong system response to the user. To avoid such unrecoverable errors, it is therefore crucial to enable SLU models with sufficient contextual sensitivity.

Differing from the previous work on contextual SLU modeling based on standard linear models [6], another important focus of this paper is to combine it with deep learning technology that has been shown to generate superior results for myriads of tasks. While neural network based techniques have been extensively applied to most of the SLU problems in recent years [7, 8, 9, 10, 11, 12, 13], they have not been fully explored for contextual understanding; to our best knowledge, [14] is the only related work of contextual neural network modeling although being limited to using the context information only from classification tasks. In this paper, we propose a more extensive joint modeling framework that exploits contextual information from both classification and sequence labeling tasks as well as the DM. This work is a multi-turn contextual model-

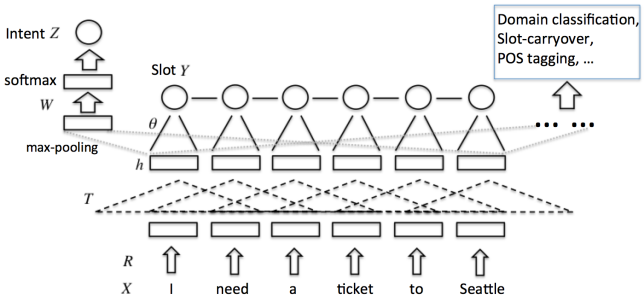

Figure 1: A CNN based multi-task learning framework for intent detection and slot filling. The same shared feature layer can be used for any classification or labeling tasks.

ing extension to the multi-task framework described in [15], as well as a multi-task extension to the recurrent neural network (RNN) based contextual domain classifier presented in [16].

The rest of the paper is organized as follows: we begin with a review of the non-contextual modeling framework for multi-task SLU based on convolutional neural networks (CNN). Next, we introduce how contextual modeling can be performed by adding recurrent and external connections. Finally, we describe the experimental setup and summarize our findings.

2. CNN Based Modeling for Single-turn Multi-task SLU

Before describing our RCNN based contextual SLU models, we first review its CNN based non-recurrent version.

2.1. CNN Based Feature Extraction

Recall that given the spoken language transcription, the conventional text featurization process extracts features according to a set of empirically designed feature templates, e.g., n-gram features. For deep learning, however, features are usually continuous and automatically learned from data – we employ CNN to derive such features from the input word sequence.

As shown in Figure 1, each word in the sentence is encoded as a fixed dimensional continuous vector, and the whole variable length query is represented as the concatenation of the word vectors. Note that CNN is particularly suitable for handling the variable length word sequence by the use of a shared convolutional feature transform T, which spans over a fixed-sized n-gram window. Thus, the hidden feature layer h with nonlinearities consists of each feature vector extracted from the shared convolutional window centered at each word position, and such a continuous space feature extraction mechanism is often believed to be able to capture more complex information than the n-gram counts. Further, the obtained features are fed into top layers for either classification or sequence labeling tasks. Note that multiple CNN layers can be stacked together to model longer range dependencies among words in the sequence.

2.2. Feature Sharing Among Multiple Tasks

The multi-task modeling framework based on CNN was first described in [17] for natural language processing. It was recently extended to simultaneously handling classification and labeling tasks for SLU in [15]. In this work, we follow the same multi-task paradigm – task-specific layers are added on top of the CNN feature layer h (Figure 1), with all tasks sharing the same feature vectors. For classification tasks, we can add

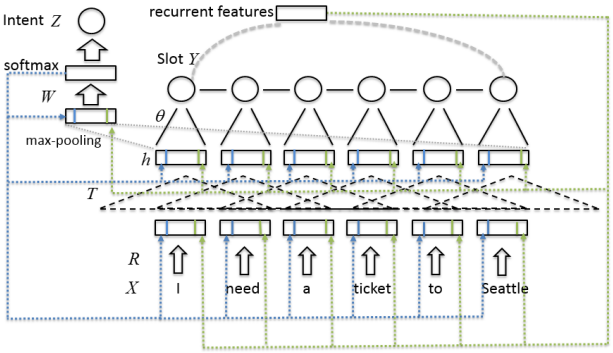

Figure 2: A RCNN based framework for contextual intent detection and slot filling. Feedback connections are added from intent and slot components back to the shared feature layers.

a max-pooling layer followed by a softmax output layer, upon which a distribution over the output classes can be derived. For sequence labeling tasks like slot filling, we deploy CRF models as the topmost layer. Note that this architecture differs from some of the previous RNN based slot taggers [10, 11, 12, 14] – the model is globally normalized in the same way as CRF, thus overcoming the notorious label bias problem [4].

For the tasks in SLU, which mostly can be categorized as either classification or tagging, they can be easily combined together to benefit simultaneously from the powerful feature learning framework. Moreover, such combination can usually lead to smaller model sizes and simplified modeling process, also making it easier for us to exploit dependencies among different tasks for contextual modeling.

The training of such an architecture requires the same procedures at the top level as the corresponding linear model counterparts. However, the gradient with respect to each top level feature needs to be further propagated back to update the convolutional layers and the word vectors.

3. RCNN Based Modeling For Multi-turn Multi-task SLU

3.1. Adding Recurrent Connections for SLU Context

The contextual ambiguities discussed before necessitate development of contextual SLU models exploiting the conversation context from previous turns as well. To exploit the contextual features within SLU components, RNN is a natural framework to capture the temporal dependencies. The network activations of either the hidden layer (Elman-type [18]) or the output layer (Jordan-type [19]) can be retained in the internal memory as a summarization of the history and reused for future turns (Figure 2). Such feedback features can be appended to either the hidden layer or the input layer, or both, and the corresponding feature transforms expand accordingly. Training of such recurrent architectures generally requires unfolding the network and propagate the gradient back through several time stamps. It is worth pointing out that the RNN based slot taggers have demonstrated a lot of success in recent years [10, 11, 12], however, none of the previous work was designed for multi-turn multi-task modeling purposes we are considering here.

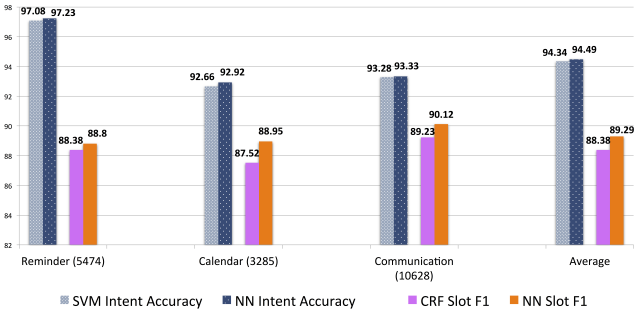

Figure 3: Non-contextual intent classification and slot filling results on all turns (including both first and follow-up turns), using SVM, CRF vs. Joint CNN.

3.2. Adding External Connections for Dialog Context

Apart from leveraging the signals within SLU components, we can further exploit the previous turn information maintained in DM components to enrich the semantic features of current turn. Note that DM maintains a representation of the dialog state by processing and summarizing the ASR and SLU outputs up to the current turn. Therefore, if the network activities of either hidden layer or input layer are connected to external information source beyond SLU components, e.g., DM, the contextual SLU interpretation can be further facilitated. In the framework described above, such external features can be simply concatenated to existing features with feature transforms augmented.

In our work, we are particularly interested in exploiting the user intents and slots (features within SLU), and dialog action (features beyond SLU) from previous turn query to improve the intent detection or slot filling of current turn; thus, our architecture is designed to this end (Figure 2). As we have pointed out, our architecture is not limited to the two tasks here; other SLU tasks also can be incorporated seamlessly, enabling a unified framework for more complex cross-turn, cross-task modeling.

4. Experimental Results

4.1. The Dataset

We evaluate our intent detection and slot filling performance on several domains using real datasets randomly extracted from the live query logs of Microsoft's Cortana personal assistant. Note that the data are in the form of sessions, and each session consists of the entire conversation between the user and the system until the user exits the application. Each domain contains 10K training queries, and the size of each test set is shown in the parenthesis of Figure 3. A similarly sized development set is used to fine-tune the training process.

4.2. Non-Contextual Modeling (SVM, CRF vs. Joint CNN)

For completeness, before evaluating the efficacy of our contextual modeling, we first present results on single-turn, non-contextual modeling, as illustrated in Figure 1. Our baseline approach consists of separate modeling for intent and slots, using internal SVM tool and CRF++ [20] respectively. For our CNN models, we use 30-dimensional word vectors, and 100-dimensional hidden feature layers with rectifier nonlinearity and feature dropout [21]. All CNN parameters including word vectors, softmax and CRF layers are randomly initialized, and further jointly trained by stochastic gradient descent; the learning rate is initially set as 0.05 and halves every time neither classification nor labeling results improve on the development set for

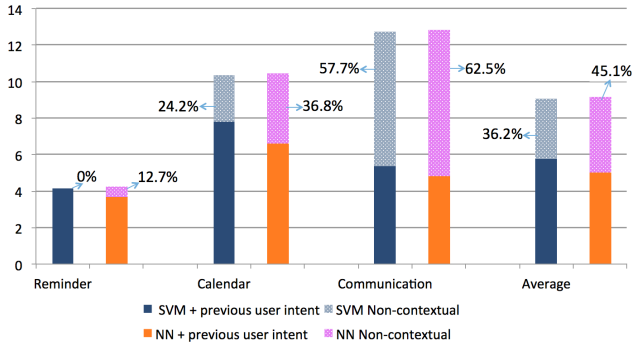

Figure 4: For both SVM and Joint CNN, intent error rates on follow-up turns with and without adding previous user intent features are shown for each domain, and the corresponding relative intent error rate reductions denoted in percentage.

5 consecutive iterations. 5-grams are used for feature extraction for both the CNN models and the baseline approaches.

As presented in Figure 3, the CNN based joint modeling consistently outperforms individual SVM and CRF results on each domain, particularly by a large margin for slot filling.

4.3. Contextual Modeling Using RCNN

Toward integrating the contextual knowledge, we enhance the non-contextual CNN based models by adding recurrent connections from the preceding intent and slot outputs to the current turn, as illustrated in Figure 2. In our experiments, we have found the type of recurrence from previous output layers (Jordan-type) consistently yields better results than from previous hidden layers. Therefore, the numbers reported in this section are all based on the Jordan-type recurrent networks.

Note that for previous user intent, the recurrent feature is the output distribution from the softmax layer in the previous turn. For previous user slot, the recurrent feature is derived as follows. Since each word in a user query is probabilistically labeled to a slot, for a given word, we first compute its marginal probability distribution over all slots from CRF models. Thus, for each slot, we can obtain its probability of not being labeled for the given word; we further multiply such probability over all the words in a query, giving the probability that a slot is not labeled for any word in the query (i.e., a slot does not occur in the query), by approximately assuming the slot independence among query words. Thus, the corresponding slot distribution, which indicates the probabilities that each slot occurs in a query respectively, can be derived and further used as recurrent features for the following turn.

Moreover, we only report results on follow-up turn queries for evaluating the contextual models. While first turn queries have empty context which can also be used as a feature, we have found in our experiments that the impact of contextual modeling is generally small for first turn queries – large benefits are mostly observed on follow-up turns.

As shown in Table 1, intent classification benefits significantly from the addition of recurrent features, although the error rate reduction comes almost entirely from the previous turn user intent. For comparison, we also evaluate the same previous user intent features as additional features for SVM. As illustrated in Figure 4, the intent error rate reduction in SVM is much smaller than that in RCNN based framework using the same contextual features, which clearly demonstrates the superior feature learning capability of our recurrent architecture.

For slot filling, the improvements are apparently more diffi-

Table 1: *Intent error rate and slot filling F1 comparisons of adding recurrent connections from previous intent and slot outputs (evaluated on follow-up turns only).*

Domain	Task	NN non-contextual	NN + previous user intent	NN + previous user slot	NN + both
Reminder	Intent error	4.27	3.73	3.97	3.73
	Slot F1	84.58	84.65	85.83	85.84
Calendar	Intent error	10.49	6.64	9.90	6.58
	Slot F1	85.85	85.40	85.44	85.42
Communication	Intent error	12.90	4.90	11.18	4.89
	Slot F1	85.64	86.21	86.73	86.73
Average	Intent error	9.22	5.09	8.35	5.07
	Slot F1	85.36	85.42	86.00	86.00

Table 2: *Intent error rate and slot filling F1 score comparisons of adding external connections from dialog system response (evaluated on a subset of follow-up turns that have non-null previous system action features).*

Domain	Task	NN non-contextual	NN + previous system action	NN + previous user intent, slot + previous system action
Reminder	Intent error	4.04	3.36	3.36
	Slot F1	86.81	87.69	88.01
Calendar	Intent error	10.08	7.75	6.20
	Slot F1	86.55	87.39	87.30
Communication	Intent error	14.02	7.23	6.50
	Slot F1	84.66	87.00	88.49
Average	Intent error	9.38	6.11	5.35
	Slot F1	86.01	87.36	87.93

cult to accomplish. On communication and reminder domains, we are able to achieve more than 1% absolute gains in F1 score with the proposed technique (no gains using CRF). On calendar domain, however, is not able to benefit from contextual modeling using either CRF or CNN. Differing from intent classification, the improvement in slot filling mainly comes from the features extracted from the previous slot output. For example, in the query session below,

Turn 1: text mike,
Turn 2: mike thanks for the call.

The non-contextual slot model has a tendency to tag "mike" in Turn 2 again as a `contact_name` while it should have been tagged as part of the `text_message` entity. The contextual model however, takes into account the fact that a `contact_name` has already been tagged at the first turn and decides it is more likely to have a `text_message` slot rather than another `contact_name` slot at Turn 2.

It is important to point out that the results on contextual modeling largely depend on the scenarios the dialog system is supporting and the corresponding level of contextual ambiguity. As multi-turn scenarios become more complicated, the contextual ambiguity will increase inevitably, leaving larger room for the proposed modeling framework to improve SLU further.

4.4. Contextual Modeling Using Dialog Response

Consider that in a typical dialog, the user first issues a query and the system generates a corresponding response; such system response to the user query plays a huge role in what the user intends to say at the follow-up turns. While it is generally maintained within the DM to conduct state tracking, it has not been fully explored for SLU. In our work, we add this feature to SLU as external connections to our RCNN framework. To illustrate, consider the Turn 1 in the example query session above; after the system processes the user query "text mike", it generates a dialog action label `SayWhat`, based on which the language generation component will prompt the user for the body of the text message. We send this label to Turn 2, which is likely to facilitate the contextual model to reduce the ambiguity of tagging the second-turn query as `text_message` and detecting the user intent as `send_text`.

In our architecture, we concatenate such label as an external feature to both hidden and input layers, and expand the feature transforms accordingly. Due to some logging issues, some part of our data does not contain the system action label; therefore, we only evaluate the resulting performance on a subset of all follow-up turns, in which the labels of previous turn system actions are available. As shown in Table 2, system action features consistently achieve substantial gains for both intent and slot modeling. Further, we find the gains are additive to those from previous user intent and slot features, suggesting the complementarity between SLU and DM components.

5. Conclusions

We have presented a neural network based joint modeling framework for multi-turn multi-task SLU which effectively encodes contextual information as recurrent features. We demonstrate our non-contextual joint model to outperform the conventional methods based on separate modeling using SVM and CRF, and our contextual modeling strategy to produce significant further gains for both intent detection and slot filling. In particular, we find adding previous user intent features provides most gain on intent classification, while most slot filling gain comes from leveraging previous user slot outputs. Moreover, adding dialog system response as external features provides consistent further gains, being complementary with recurrent SLU features.

6. References

[1] D. Jurafsky and J. H. Martin, *Speech and language processing*. Prentice Hall, 2008.

[2] T. Gonzalez, J. Diaz-Herrera, and A. Tucker, *Computing Handbook: Computer Science and Software Engineering*. Chapman & Hall/CRC, 2014.

[3] V. Vapnik, *The nature of statistical learning theory*. Springer Science & Business Media, 2000.

[4] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," *Proc. of ICML*, 2001.

[5] M. Henderson, B. Thomson, and J. Williams, "The second dialog state tracking challenge," in *Proc. of SIGdial*, 2014.

[6] A. Bhargava, A. Celikyilmaz, D. Hakkani-Tur, and R. Sarikaya, "Easy contextual intent prediction and slot detection," in *Proc. ICASSP*, 2013.

[7] R. Sarikaya, G. E. Hinton, and B. Ramabhadran, "Deep belief nets for natural language call-routing," in *Proc. ICASSP*, 2011.

[8] G. Tur, L. Deng, D. Hakkani-Tur, and X. He, "Towards deeper understanding: deep convex networks for semantic utterance classification," in *Proc. ICASSP*, 2012.

[9] A. Deoras and R. Sarikaya, "Deep belief network based semantic taggers for spoken language understanding," in *Proc. INTERSPEECH*, 2013.

[10] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, "Recurrent neural networks for language understanding," in *Proc. INTERSPEECH*, 2013.

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

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

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

[14] Y. Shi, K. Yao, H. Chen, Y.-C. Pan, M.-Y. Hwang, and B. Peng, "Contextual spoken language understanding using recurrent neural networks," in *Proc. ICASSP*, 2015.

[15] P. Xu and R. Sarikaya, "Convolutional neural network based triangular crf for joint intent detection and slot filling," in *Proc. ASRU*, 2013.

[16] P. Xu and R. Sarikaya, "Contextual domain classification in spoken language understanding systems using recurrent neural network," in *Proc. ICASSP*, 2014.

[17] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. of ICML*, 2008.

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

[19] M. I. Jordan, "Serial order: A parallel distributed processing approach," *Advances in psychology*, vol. 121, pp. 471–495, 1997.

[20] T. Kudo, "CRF++: Yet another CRF toolkit," *http://crfpp.googlecode.com/svn/trunk/doc/index.html*, 2009.

[21] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv*, 2012.