A HIERARCHICAL DECODING MODEL FOR SPOKEN LANGUAGE UNDERSTANDING FROM UNALIGNED DATA

Zijian Zhao, Su Zhu and Kai Yu

MoE Key Lab of Artificial Intelligence
SpeechLab, Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, China
{1248uu, paul2204, kai.yu}@sjtu.edu.cn

ABSTRACT

Spoken language understanding (SLU) systems can be trained on two types of labelled data: aligned or unaligned. Unaligned data do not require word by word annotation and is easier to be obtained. In the paper, we focus on spoken language understanding from unaligned data whose annotation is a set of act-slot-value triples. Previous works usually focus on improve slot-value pair prediction and estimate dialogue act types separately, which ignores the hierarchical structure of the act-slot-value triples. Here, we propose a novel hierarchical decoding model which dynamically parses act, slot and value in a structured way and employs pointer network to handle out-of-vocabulary (OOV) values. Experiments on DSTC2 dataset, a benchmark unaligned dataset, show that the proposed model not only outperforms previous state-of-the-art model, but also can be generalized effectively and efficiently to unseen act-slot type pairs and OOV values.

***Index Terms*—** Spoken language understanding, unaligned data, hierarchical decoding, pointer network

1. INTRODUCTION

The spoken language understanding (SLU) module is a key component of spoken dialogue system (SDS), parsing user's utterances into corresponding semantic forms. Typically, the SLU problem is regarded as a sequence tagging task which needs word-level annotations[1, 2, 3], e.g., the utterance *"Show me flights from Boston to New York"* can be parsed as *"Show me flights from [Boston:from_city] to [New York:to_city]"* [4]. Beyond this word aligned annotation, there is also sentence-level semantic annotation which is unaligned, e.g., the utterance *"I want a high priced restaurant"* has an act-slot-value triple annotation of *"inform(pricerange=expensive)"* and the utterance *"what type of food does it serves"* has an annotation of *"request(food)"*.

The unaligned SLU has some advantages against the aligned one. First, as a downstream module of Automatic Speech Recognition (ASR), SLU module based on statistical method often requires that training data should be labelled on the outputs from ASR, which can improve robustness to ASR errors. Therefore, it is hard and sometimes impossible to align the semantic annotations onto ASR outputs due to ASR errors (especially word insertion and deletion errors). Second, value aliases are also difficult to be handled in a word-aligned way which is very time-consuming. In this paper, we focus on SLU with the unaligned semantic annotation that a sentence is labelled as a set of act-slot-value triples [5].

There are numerous previous works for the unaligned SLU. Support vector machines (SVM) have been used for learning semantic tuple classifiers [6, 7]. Yazdani et al. propose a model to calculate the similarity between the input sentence and all possible semantic tuples [8]. It assumes all possible values have been known, which may be impractical in real applications and inefficient, e.g. there are a large number of songs in a music domain. Sentence and context representations are exploited in [9] and the OOV problem in slot values is addressed by utilizing a pointer network in [10]. However, they predict act type and slot-value pair separately ignoring relation between the act and slot-value pair.

In this paper, we propose a novel hierarchical decoding model for SLU from unaligned data. The model predicts act-slot-value triples hierarchically by following the triple structure. The hierarchical decoding can predict multiple act-slot-value triples completely and generalize to unseen act-slot type pairs. Pointer network [11] is employed to generate out-of-vocabulary (OOV) values with a context-aware attention mechanism. In the experiments, our method achieves state-of-the-art performance in the DSTC2 dataset, and shows great generalization capacity.

The rest of the paper is organized as follows. The next section introduces relations to prior works. In section 3, we describe the hierarchical decoding model in detail. Experiments and analyses are presented in section 4, followed by conclusions.

The corresponding author is Kai Yu. This work has been supported by the China NSFC projects (No. 61573241). Experiments have been carried out on the PI supercomputer at Shanghai Jiao Tong University.

978-1-5386-4658-8/18/$31.00 ©2019 IEEE

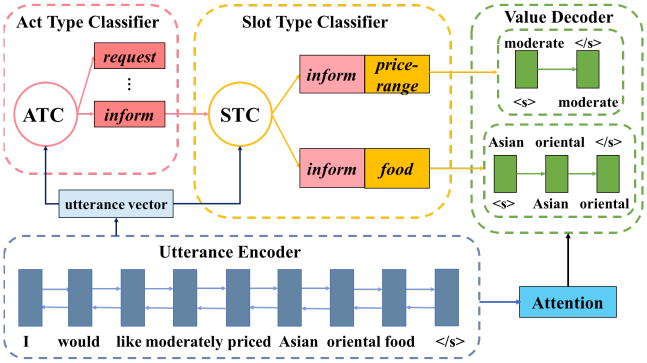

Fig. 1. A Hierarchical Decoding Model for Spoken Language Understanding from Unaligned Data. ATC denotes act type classifier and STC denotes slot type classifier.

2. RELATION TO PRIOR WORK

The work most relevant to us is [10], which also uses a pointer network [11] to handle the OOV problem of values. They focus on filling value for a slot which is a sub-task of SLU, while we aim to hierarchically generate a set of act-slot-value triples which is a complete target of the unaligned SLU. Moreover, the relation between act and slot-value pair is ignored in [10]. We apply a context-aware attention mechanism within the pointer network by incorporating the predicted act and slot. A comparison of results on the DSTC2 dataset also shows that we can get better performance.

3. HIERARCHICAL DECODING MODEL

In this section, the details of our model are given. The task of SLU from unaligned data is to predict a set of act-slot-value triples given an input utterance. To represent rich semantics, the triples are of three types: single act like *"thankyou()"* and *"bye()"*, act-slot pair like *"request(food)"*, and act-slot-value triple like *"inform(pricerange=expensive)"*, which are given in domain ontology. Not all act types are followed by a slot and value. Thus, we predict the act-slot-value triples by following the triple structure. The overall model consists of four modules, as shown in Figure 1:

- a shared utterance encoder;
- an act type classifier with the utterance as input to predict act types;
- a slot type classifier with the utterance and an act type as inputs to predict slot types;
- a value decoder with the utterance and an act-slot type pair as inputs to generate the value sequence.

In training state, the modules can be trained together at the same time as multi-task learning. However, in testing stage, the modules decode recursively to generate the triples. The details of hierarchical decoding are given in Algorithm 1.

Algorithm 1: Hierarchical Decoding Algorithm

```
Input: utterance, ontology, model
Output: a set of act-slot-value triples
Initialize a empty list;
pred_acts = act_classifier(utterance);
for act in pred_acts do
    if the act doesn't need a slot then
        add the single act into the list;
    else
        pred_slots = slot_classifier(utterance, act);
        for slot in pred_slots do
            if the slot doesn't need a value then
                add the act-slot pair into the list;
            else
                value = value_decoder(utterance, act,
                  slot);
                add the act-slot-value triple to the list;
            end
        end
    end
end
Return the list containing triples.
```

3.1. Shared Utterance Encoder

A bidirectional LSTM (BLSTM) [12, 13] model is exploited to encode the utterance. Let $\boldsymbol{e}_w$ denote the word embedding of each word w, and $\oplus$ denote the vector concatenation operation. The encoder reads the utterance $\boldsymbol{w} = (w_1, w_2, \cdots, w_T)$ and generates T hidden states of BLSTM:

$$\boldsymbol{h}_i = \overleftarrow{\boldsymbol{h}_i} \oplus \overrightarrow{\boldsymbol{h}_i}; \quad \overleftarrow{\boldsymbol{h}_i} = f_l(\overleftarrow{\boldsymbol{h}_{i+1}}, \boldsymbol{e}_{w_i}); \quad \overrightarrow{\boldsymbol{h}_i} = f_r(\overrightarrow{\boldsymbol{h}_{i-1}}, \boldsymbol{e}_{w_i})$$

where $\overleftarrow{\boldsymbol{h}_i}$ is the hidden vector of the backward pass in BLSTM and $\overrightarrow{\boldsymbol{h}_i}$ is the hidden vector of the forward pass in BLSTM at time i, f_l and f_r are LSTM units [14] of the backward and forward pass respectively. The final representation of the utterance (*utterance vector*) is defined as:

$$\tilde{\boldsymbol{h}} = \overleftarrow{\boldsymbol{h}_1} \oplus \overrightarrow{\boldsymbol{h}_T}$$

The utterance vector $\tilde{\boldsymbol{h}}$ will be used for the following act and slot type classifications, and hidden vectors $\{\boldsymbol{h}_1, \cdots, \boldsymbol{h}_T\}$ will be used for the value sequence generation with pointer network [11, 15, 16].

3.2. Act and Slot Type Classifiers

Act type prediction is defined as a multi-label classification problem here. A normal solution is to train a binary classifier for each label. We apply a feed forward network with two layers to calculate an existence score for each possible label:

$$\boldsymbol{r} = \mathrm{ReLU}(\boldsymbol{W}_u\boldsymbol{u} + \boldsymbol{b}_u)$$
$$\boldsymbol{p} = \sigma(\boldsymbol{W}_r\boldsymbol{r} + \boldsymbol{b}_r)$$

where $\boldsymbol{u}$ is the input vector, $\boldsymbol{W}_u, \boldsymbol{W}_r$ are weight matrices and $\boldsymbol{b}_u, \boldsymbol{b}_r$ are biases. σ is the sigmoid function to normalize output scores. In the training stage, Binary Cross Entropy (BCE) loss function[1] is used. In the testing stage, classes with score higher than 0.5 are predicted. For act type prediction, the input vector $\boldsymbol{u}$ is just the utterance vector $\tilde{\boldsymbol{h}}$.

Slot type prediction is formatted in a similar way, while not only the utterance vector but also the corresponding act type are fed to the slot classifier. An embedding layer is also defined to encode each act type into a continuous vector. Let a denote an act type and $\boldsymbol{e}_a$ denotes its embedding, then the input vector for the slot type classifier is:

$$\boldsymbol{u} = \tilde{\boldsymbol{h}} \oplus \boldsymbol{e}_a$$

A notable point is that we define embedding modules for act and slot types as word embedding to encode each type into a continuous representation. This allows us to utilize the predicted results from former modules in the latter, e.g., usage of act types in slot types prediction.

3.3. Value Decoder with Pointer Network

To predict value of the corresponding act-slot type pair, we utilize a sequence-to-sequence model with attention [17] and pointer network [11] to generate word sequence of the value. Since the encoder has been introduced above, we describe the details of the decoder below.

A LSTM model is used to decode the value sequence $\boldsymbol{v} = (v_1, v_2, \cdots, v_N)$. We define v_N as *"</s>"* which means the end of a sequence. The LSTM proceed as $\boldsymbol{s}_i = f(\boldsymbol{s}_{i-1}, \boldsymbol{e}_{v_i})$, where $\boldsymbol{s}_i$ is the hidden vector at time i and f is the LSTM units. In order to incorporate the context information of corresponding act and slot, we define that:

$$\tilde{\boldsymbol{s}}_i = \boldsymbol{W_s}(\boldsymbol{s}_i \oplus \boldsymbol{e}_a \oplus \boldsymbol{e}_s) + \boldsymbol{b_s}$$

where $\boldsymbol{e}_a$ and $\boldsymbol{e}_s$ are embeddings of corresponding act type a and slot type s respectively, $\boldsymbol{W}_s$ is a weight matrix and $\boldsymbol{b}_s$ is a bias vector. $\tilde{\boldsymbol{s}}_i$ is used in the attention mechanism to calculate context vector $\boldsymbol{c}_i$ as follows:

$$\boldsymbol{c}_i = \sum_{j=1}^{T} \alpha_{ij}\boldsymbol{h}_j; \quad \alpha_{ij} = \frac{\exp(\boldsymbol{h}_j^T \tilde{\boldsymbol{s}}_i)}{\sum_{k=1}^{T} \exp(\boldsymbol{h}_k^T \tilde{\boldsymbol{s}}_i)}$$

The encoded information of predicted act and slot in $\tilde{\boldsymbol{s}}_i$ could help the attention mechanism to focus semantically.

Finally $\tilde{\boldsymbol{s}}_i$ and $\boldsymbol{c}_i$ are concatenated to be an input of the output layer which calculates the probability distribution $\boldsymbol{P}_i^{gen}$ over the basic vocabulary as in [17].

To handle the OOV problem in value generation, we enhance the basic Seq2Seq model with pointer network [11] which can generate a probability distribution $\boldsymbol{P}_i^{ptr}$ over the words of the input utterance according to the attention weights α_{ij}. Therefore, the final distribution over the extended vocabulary (the basic vocabulary and words in the input utterance) is calculated as:

$$\boldsymbol{P}_i = p_i * \boldsymbol{P}_i^{gen} + (1 - p_i) * \boldsymbol{P}_i^{ptr}$$
$$p_i = \sigma(\boldsymbol{w_p}(\boldsymbol{e}_{v_i} \oplus \tilde{\boldsymbol{s}}_i \oplus \boldsymbol{c}_i) + b_p)$$

where p_i is a balance score, $\boldsymbol{w}_p$ is a weight vector and b_p is a scalar bias.

4. EXPERIMENTS

In our experiments, we use the dataset provided for the second Dialog State Tracking Challenge (DSTC2) [18]. It encompasses $11677, 3934, 9890$ pairs of utterance and the corresponding set of act-slot-value triples for training, development and testing respectively. Each utterance is annotated with semantics including multiple act-slot-value triples. Both the manual transcription and 10-best hypotheses are provided for each utterance. We use the manual transcription and top hypothesis (1-best) as inputs throughout our experiments.

The dimension of embeddings is 100 and the number of hidden units is 128. Dropout rate is 0.5 and batch size is 20. Maximum norm for gradient clipping is set to 5 and Adam optimizer is used with an initial learning rate of 0.001. All training consists of 50 epochs with early stopping on the development set. We report F1-score of act-slot-value triples by the offical scoring script from http://camdial.org/ mh521/dstc/.

Glove[2] word vectors are used to initialize all the embedding modules. For act and slot type embedding modules, we compose the embedding of these abstract concept words, for example, the embedding of *"pricerange"* is the average of the embeddings of *"price"* and *"range"*. We also tie the act embedding and the topmost weight matrix of the act type classifier [19], same for the slot embedding.

4.1. Overall Performance

First of all, we conduct experiments on the top hypothesis and compare the results with prior arts to evaluate the overall SLU performance of our model. [9, 10] are neural network based methods which have been mentioned before, and [20] is a statistical method which uses decision trees based binary classifiers to predict the presence of each slot-value pair and dialog act. The results are shown in Table 1. From the table, we can see that our model achieves the best F1 score and outperforms the prior works significantly.

[1] Binary cross entropy loss suits multi-labels classification very well: $L = -\sum_i [y_i * log(p_i) + (1 - y_i) * log(1 - p_i)]$, where y_i is the target value (0 or 1) of i-th label and p_i is the predicted probability of i-th label.

[2] `http://nlp.stanford.edu/data/glove.6B.zip`

Model	F1-score(%)
SLU2 [20]	82.1
CNN+LSTM_w4 [9]	83.6
S2S-Attn-Ptr-Net [10]	85.8
Our model	**86.9**

Table 1. Comparison with published results on DSTC2.

4.2. Generalization Capacity

In this section, we would like to evaluate the generalization capacity of our hierarchical decoding (HD) model. We adopt two baselines for comparison. One baseline [7] treats act-slot-value triple as a single label and train binary SVM classifiers to predict the existence of each label. The method is named as semantic tuple (ST) classifications. We replace the SVM with our BLSTM encoder for consistency. The other baseline [8] proposes a model to calculate the similarity between the embeddings of input utterance and all possible semantic items (act-slot-value triples) by zero-shot (ZS) learning. We also apply a BLSTM encoder to get the utterance vector. We randomly select $5\%, 10\%, 20\%$ of the training data to create specific datasets with less act-slot-value triples. The testing set is the same as before. Both manual transcriptions and top hypotheses (1best) are used in the experiments. The results are shown in Table 2.

Data Type	Data Size	ST	ZS	HD
manual	5%	70.0	80.4	**91.7**
	10%	92.4	91.0	**93.6**
	20%	95.1	94.2	**96.1**
	100%	**98.3**	98.1	**98.3**
1best	5%	50.6	73.1	**80.1**
	10%	80.0	79.6	**80.5**
	20%	83.2	81.9	**83.3**
	100%	**87.2**	86.1	86.9

Table 2. SLU performance (F1) with varying training size.

As we can see, the performance of our model will not degrade heavily as the data becomes less and less. Especially, it achieves a much better F1 score than the baselines when only 5% data remains. The results show that our model has a good capacity of generalization.

To better explore the reason why our model achieves much better performance than the baselines with only 5% data, we split the labels in the testing set into two categories according to whether the act-slot-value triple is seen in training set. Subsequently we report the F1 scores of our model and the baselines on these two categories in Table 3. We find that the decomposition of act-slot-value triples and hierarchical decoding of our method can generalize to unseen labels and improve the performance on seen labels simultaneously.

Data Type	Label Type	ST	ZS	HD
manual	seen	71.1	81.2	**92.2**
	unseen	0.0	10.7	**72.6**
1best	seen	51.1	74.2	**80.8**
	unseen	0.0	5.2	**45.3**

Table 3. SLU performance (F1) with 5% training data on two categories of labels (act-slot-value triples).

4.3. Analysis

The decomposition of act-slot structure allows us to predict unseen act-slot type pairs. For example, our model can predict *"confirm(area)"* even if the pair does not exist in training set. Since it can learn to compose the semantics of *"confirm(area)"* from *"confirm(food)"* and *"inform(area)"*.

For non-enumerable slot types like *"food"* and *"name"* which may have a huge set of possible values, we can not define all the possible values in advance. The utilization of pointer network allows us to generate OOV values. In our experiments, most OOV values can be generated by recognizing the similar context around the values with pointer network.

Given the predicted act and slot, the attention mechanism of the value decoder would focus on corresponding words. This enables the decoder to generate values accurately. Figure 2 shows an example that how attention weights are distributed on the input utterance given different act-slot pairs. We can see that, "*inform-slot*" focuses on "thai" and "*deny-slot*" concentrates on "chinese" extremely.

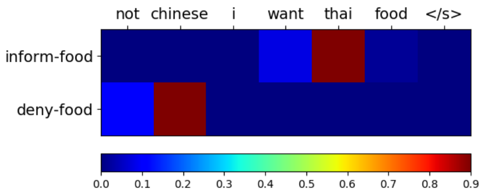

Fig. 2. Attention weights on input utterance of the value decoder with different act-slot pairs.

5. CONCLUSION

In this paper, we propose a novel hierarchical decoding model for SLU from unaligned data. The model exploits the structure of act-slot-value triples and can completely predict multiple triples. The decomposition of act-slot structure makes it possible to predict unseen act-slot type pairs. The utilization of pointer network in value decoder allows us to generate out-of-vocabulary (OOV) slot values. Finally, the experiment results show that our model possesses impressive performance and generalization capacity. In future, we would like to improve embeddings of act and slot types, and apply our method in domain adaptation problem of SLU.

6. REFERENCES

[1] Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio, "Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding.," in *Interspeech*, 2013, pp. 3771–3775.

[2] Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu, "Recurrent neural networks for language understanding.," in *INTERSPEECH*, 2013, pp. 2524–2528.

[3] Su Zhu and Kai Yu, "Encoder-decoder with focus-mechanism for sequence labelling based spoken language understanding," in *IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP)*, 2017, pp. 5675–5679.

[4] Roberto Pieraccini, Evelyne Tzoukermann, Zakhar Gorelov, J-L Gauvain, Esther Levin, C-H Lee, and Jay G Wilpon, "A speech understanding system based on statistical representation of semantics," in *icassp*. IEEE, 1992, pp. 193–196.

[5] Steve Young, "Cued standard dialogue acts," *Report, Cambridge University Engineering Department, 14th October*, 2007.

[6] François Mairesse, Milica Gasic, Filip Jurcicek, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young, "Spoken language understanding from unaligned data using discriminative classification models," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. IEEE, 2009, pp. 4749–4752.

[7] Matthew Henderson, Milica Gašić, Blaise Thomson, Pirros Tsiakoulis, Kai Yu, and Steve Young, "Discriminative spoken language understanding using word confusion networks," in *Spoken Language Technology Workshop (SLT), 2012 IEEE*. IEEE, 2012, pp. 176–181.

[8] Majid Yazdani and James Henderson, "A model of zero-shot learning of spoken language understanding," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 244–249.

[9] Lina M Rojas Barahona, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, Stefan Ultes, Tsung-Hsien Wen, and Steve Young, "Exploiting sentence and context representations in deep neural models for spoken language understanding," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 258–267.

[10] Lin Zhao and Zhe Feng, "Improving slot filling in spoken language understanding with joint pointer and attention," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2018, vol. 2, pp. 426–431.

[11] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly, "Pointer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2692–2700.

[12] Mike Schuster and Kuldip K Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[13] Kazuya Kawakami, *Supervised Sequence Labelling with Recurrent Neural Networks*, Ph.D. thesis, PhD thesis. Ph. D. thesis, Technical University of Munich, 2008.

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

[15] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio, "Pointing the unknown words," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, August 2016, pp. 140–149.

[16] Abigail See, Peter J Liu, and Christopher D Manning, "Get to the point: Summarization with pointer-generator networks," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, vol. 1, pp. 1073–1083.

[17] Minh-Thang Luong, Hieu Pham, and Christopher D Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[18] Matthew Henderson, Blaise Thomson, and Jason D Williams, "The second dialog state tracking challenge," in *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2014, pp. 263–272.

[19] Ofir Press and Lior Wolf, "Using the output embedding to improve language models," *arXiv preprint arXiv:1608.05859*, 2016.

[20] Jason D Williams, "Web-style ranking and slu combination for dialog state tracking," in *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2014, pp. 282–291.