Universität des Saarlandes
Department Informatik
Computer Linguistics

Span extraction based slot-filling using attention and RNNs

Seminar Paper

Dialogue processing for human-robot collaboration

Pavle Marković, Sangeet Sagar
Matr.Nr. 7007913, 7009050
{pama00002,sasa00001}@stud.uni-saarland.de

April 29, 2022

Abstract

In this work, we attempt to perform slot-filling using a recurrent neural network (RNN) model based on a multi-head attention mechanism. The model considers both user's utterance and slot type in determining the slot value. The attention mechanism aids in capturing key components in the dialogue related to the slot type. We compare our results to other recent state-of-the-art models based on the F1 score.

Contents

1 Introduction

Spoken language understanding (SLU) involves identifying two key components in a human-machine dialogue system to predict response to a human utterance: identifying the user's intent and identifying constituent parts from the user's utterance concerning the intent. In this work, we will focus on the latter. In the following utterance where a user wishes to perform booking at a restaurant: *There will be 5 adults and 1 child*, the user's intent is "restaurant reservation", and this intent is identified by the value *5 adults and 1 child* whose slot type is *people*. Therefore, we are attempting to perform slot-filling where the goal is to fill in appropriate values for a slot.

Slot filling is a key component of the spoken dialogue system (SDS). This can be treated as a token-level sequence labeling task where given an input sequence: $\mathcal{X} = (X_1, X_2, \cdots, X_N)$ of a specific slot type: *slot*, the goal is to label each token in the sequence with a value $\mathcal{Y}^{slot} = (Y_1^{slot}, Y_2^{slot}, \cdots, Y_N^{slot})$. Generative models like Hidden Markov Models [3] with the Viterbi algorithm are conventional approaches to predicting the most likely tags of a given sequence of words. More recent approaches to this use conditional random fields (CRFs), which happens to be a variant of the Markov network, performs this elegantly. With the advancement of neural-based methods like recurrent neural networks (RNNs), when integrated with more recent attention mechanisms [2] such tasks have achieved a drastic increase in performance [10].

In modular systems, slot filling (and intent detection) is either preceding component or a sub-component of the dialogue state tracking (DST) thus vital for properly modelling context. We approach this task as a value (span) extraction. In our work we use BERT [5] to encode slot and SpanBERT [8] to encode user's utterance and perform slot-filling using multi-headed attention-based RNN. Our approach to multi-head attention is inspired from [11]. Rather than overloading the decoder with the entire input sequence, the attention mechanism helps identify the most important constituent words in a dialogue that should concern a dialogue system when slot filling. Multi-head attention uses a self-attention layer multiple times in parallel, leading to a performance boost. This is the key motivation for our work. We use the Restaurant8K dataset [4] to perform our experiments and evaluate our results on. Our code is open-sourced and can be found on `https://github.com/sangeet2020/dialogue-processing`.

2 Model

2.1 Context Encoder

The key observation that we found in our literature review [4] was that proposed work did not take into account slot type for which value is to be found. In a production system, this information should be available from system's action (dialogue management would produce a dialogue act with slots for which system expects

values from a user). Our hypothesis is that such information could help system improve performance and extract corresponding values better.

Inspired by the paper we presented in the seminar [9], we decided to use BERT based encoders for our inputs (a user's utterance and a slot type). First, WordPiece tokenizer is used to tokenize both the slot type and the user's utterance. Next, the tokenized slot type is input into the BERT encoder to obtain summarized representation of the slot type (as in [9], output vector of [CLS] token is taken). On the other hand, since we modelled the task as a span extraction, it seemed appropriate to use SpanBERT [8] to encode the user's utterance (again as in [9], output token vectors are taken). After obtaining contextualised vector representations for our inputs, the multi-head attention is applied (described in the following subsection).

2.2 Multi-head attention

The simplistic self-attention mechanism computes a context-aware embedding vector for each input. However, this does not involve any learnable parameters. Multi-head attention mechanism uses three learnable weight parameters for query, key and value (different for every self-attention layer) that are multiplied with the input sequence of embeddings (in our case, these will be encoded context vectors). This allows attending to different parts in a sequence differently.

In our work, we use the multi-head attention as proposed in [11]. The input to our multi-head attention layer, i.e. query (Q) is the encoded slot-vector $\mathbf{Q}_V$ and for key (Q) and value (V) is encoded utterance vector $\mathbf{Q}_U$. It further performs scaled-dot product attention (A) using these inputs that give a h linear projections stacked over one another, where h is the number of heads.

$$A(Q, K, V) = \text{Softmax}\left(\frac{Q \cdot K^\top}{\sqrt{d_k}}\right) \tag{1}$$

where $\sqrt{d_k}$ is a scaling factor given by dimension of encoded context vector divided by number of heads. These are then concatenated and fed through linear layer to obtain the final output. These attention weights are combined with the encoded utterance vector to form the input to RNN discussed in next section.

2.3 RNN

The next step is to perform span extraction which was done through sequence tagging. The BIO format was used for this purpose, so each input token is tagged with B, I and O tags. B and I indicate beginning and inside of slot's value span, while O indicates outside of slot-span. In the Table 2 an example of the tagged user's utterance can be found. Lastly, additional P tag was used to tag PAD tokens.

Actual tagging process is performed using RNN mechanism, more specifically the LSTM [7]. In each time step, an LSTM cell receives the hidden state and cell state from the previous time step, the current time step token and the output of

the multi-head attention. The current token and the attention outputs are first concatenated, then put through a linear layer and Sigmoid activation function to make more compact input for the LSTM cell. Finally, the output of the LSTM cell is sent through a Layer Normalization [1] (following [9]), and the final result is a predicted BIO tag.

3 Dataset

We use the Restaurant8K dataset to train our slot-filling model and evaluate our results. It consists of 8198 user utterances and 5 unique slot types. The dataset consists of requests and queries related to the restaurant domain. A total of 2975 utterances lacked any labels, and most of them accompanied multiple labels. We perform our experiments and compare results both inclusive and exclusive of these non-labeled utterances to determine if the model learns to predict from non-labeled utterances. To simplify our experiments, we cloned each example utterance to its number of labels, i.e. an example with three labels were cloned thrice and mapped each to one label. We had 7635 examples in the train set and 3477 examples in the test set. Slot distribution can be seen in the Table 1

Slot type	No. of utt.
people	2164
date	1721
time	1972
first_name	887
last_name	891

Table 1: *Slot distribution on training data*

Utterance	There	will	be	5	adults	and	1	child.
BIO format	O	O	O	B	I	I	I	I
Slot type	people							
Slot Value	*5 adults and 1 child.*							

Table 2: *Training example with slot type and value*

4 Experiment

For our experiments, we used $BERT_{BASE}$ model (12 layers with 768 hidden units) for tokenization, slot-type encoding and as a underline model for SpanBERT. Multi-head attention is further applied where the query is slot vector, while keys

and values are utterance vectors. Our multi-head configuration employs 8 heads with 768 hidden units. We train a single-layered LSTM model with 512 input units and 100 hidden units and specify a dropout probability of 0.1. We use Adam optimizer with a learning rate of 10^{-5} and cross-entropy loss function for the model training for 10 epochs

Initially, we trained the model on only labeled utterances and later extended to all examples inclusive of non-labelled utterances. We report the mean precision, recall and F1 score for each model type in Section 5. We also report slot-specific F1 scores to analyze how well the model captures slot value information for different slots types.

5 Results and Discussion

The achieved results per slots are shown in the Table 3, while summarized results over all slots, compared to other models, are presented in the Table 4. For evaluation, we followed [4] and calculated F1 score (the weighted average of Precision and Recall).

First of all, it can be observed that time slot is the least challenging one due to small values variety. On the other hand, the most challenging slots are first_name and last_name, where the score for the last_name is considerably lower (35.33 for last_name compared to 67.36 for first_name). This might suggest that our model is more incline towards first_name when it spot a name value. Also, number of values per these two slots are at least twice smaller compared to other slots (see Table 1). Given that values for these two slots are the most diverse, it is reasonable to assume that more examples are required. It is encouraging that the score for people slot is somewhat decent (74.78) given diversity of possible values (from a single number (e.g. 5) to wordy sequence (e.g. 1 adult and 4 children). Lastly, except for the first_name slot, there is no significant difference between results for all utterances and only labelled ones used for training.

Unfortunately, our model underperformed compared to previously proposed models. One of the reasons for that might be the selected sequence tagging mechanism. Our model relies on the RNN architecture, while all others on the Conditional Random Fields (CRF). In addition to that, we employed multi-headed attention mechanism, thus, it might be that there is no a sufficient number of training examples to properly parametrize these two mechanisms.

Slot type	**All utts**	**Only labelled utts**
people	74.78	74.88
date	86.22	85.93
time	98.24	96.68
first_name	67.36	49.48
last_name	35.33	35.95

Table 3: *Slot specific F1 scores on test set with model trained on all utterances and when trained on only labeled utterances*

Model	**F1**
Span-ConveRT[4]	96
V-CNN-CRF [4]	94
Span-BERT [4]	93
Our model	83

Table 4: *Mean F1 scores comparison*

6 Conclusion and Future work

We have implemented a slot-filling model based on [9] and [4] papers. The main idea was to model the slot-filling task as a span extraction and to utilize available information about slot type for which value is to be provided. Although our result did not match results of previously proposed models, we believe there are space to improve our approach to possibly match those. For the future work, other sequence tagging mechanisms should be explored (e.g. CRF are used by other models). Furthermore, the Span-ConveRT model uses ConveRT model [6] for encoding input which is specifically trained for better representation of conversational utterances. It would be interesting to apply such model for our approach as well.

References

[1] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Leonard E. Baum and Ted Petrie. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics*, 37(6):1554 – 1563, 1966.

[4] Sam Coope, Tyler Farghly, Daniela Gerz, Ivan Vulic, and Matthew Henderson. Span-convert: Few-shot span extraction for dialog with pretrained conversational representations. *CoRR*, abs/2005.08866, 2020.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[6] Matthew Henderson, Iñigo Casanueva, Nikola Mrkvsi'c, Pei hao Su, Tsung-Hsien, and Ivan Vulic. Convert: Efficient and accurate conversational representations from transformers. *ArXiv*, abs/1911.03688, 2020.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. 9(8), 1997.

[8] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

[9] Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. Sumbt: Slot-utterance matching for universal and scalable belief tracking. *ArXiv*, abs/1907.07421, 2019.

[10] Bing Liu and Ian R. Lane. Attention-based recurrent neural network models for joint intent detection and slot filling. *CoRR*, abs/1609.01454, 2016.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.