Semantic Slot Prediction on low corpus data using finite user defined list

Bharatram Natarajan, Simma Dharani, Chirag Singh, Anish Nediyanchath and **Sreoshi Sengupta**
research.samsung.com
{bharatram.n, simma.d, c.singh, anish.n, s.sengupta}@samsung.com

Abstract

Semantic slot prediction is one of the important task for natural language understanding (NLU). They depend on the quality and quantity of the human crafted training data, which affects model generalization. With the advent of voice assistants exposing AI platforms to third party developers, training data quality and quantity matters for any machine learning algorithm to learn and generalize properly. AI platforms provides provision to add custom external plist defined by the developers for the training data. Hence we are exploring dataset, called LowCorpusSlotData, containing low corpus training data with larger number of slots and significant test data. We also use external plist for the above dataset to aid in slot identification. We experimented using state of the art architectures like Bi-directional Encoder Representations from Transformers (BERT) with variants and Bi-directional Encoder with Custom Decoder. To address the low corpus problem, we propose a pipeline approach where we extract candidate slot information using the external plist extractor module and feed as input along with utterance.

1 Introduction

In recent market, many voice assistants like Samsung Bixby, Amazon Alexa are providing AI platforms for developers outside to develop their custom applications. In these platforms, developers add the data for their custom made applications which is being used by the voice assistant for the training its NLU components. The main responsibility of NLU is to understand the user utterance and to determine the domain, intent and slots. Domain Prediction extracts the domain in which the utterance belongs to i.e., the application that will execute the user intention. Intent Detector is to extract the user intention or the action that we will execute inside the application. Slot Filler is to extract the semantic slots or objects of interest on which we execute the action. Example is name entity finding like name of person, location and date time. The data, developed by the developers, might be small for training slot-detection task when compared to intent detection or domain prediction task. Therefore, the current scope of this paper is limited solving the Slot Filler task. To explain Slot Filler task with an example, consider the utterance "Is the Barbeque Nation closed at this time?". Slot Filler task is to identify "Barbeque Nation" as BusinessName and "Closed" as OpenHourDescriptor from the utterance.

Lots of work is going on the field of slot prediction as independent task. Shin et al. (2018) proposes the architecture based on encoder-decoder attention model with aligned input where Bi-GRU as encoder and GRU decoder which learns jointly both slot filling and delexicalized sentence generation. Saha et al. (2018) proposes the variants of LSTM and GRU integrated with the CRF layer at end for the task of slot filling. Mesnil et al. (2014) recommends the usage of RNN for the slot prediction. Jaech et al. (2016) performs an experiment using a multi-task model with open vocabulary embeddings increases the generalizability by which the data required for the training the slot-filling is minimalised. Huang et al. (2015) proposes a various LSTM-CRF models for sentence tagging. Kurata et al. (2016) proposes the encoder-labeler LSTM which performs slot filling conditioned on the encoded sentence-level information which was generated by LSTM. Shi et al. (2016) recommends a recurrent support vector machine, which is a combination of recurrent neural network, and a structured support vector machine for the slot tagging. Liu et al. (2020) proposes a cross-domain slot filling using Bi-LSTM for handling the limitation of data and unseen slot types. Zhang et al. (2018) proposes a capsule based neural network model,

Proceedings of the 17th International Conference on Natural Language Processing, pages 329–333
Patna, India, December 18 - 21, 2020.

which accomplishes slot filling and intent detection via a dynamic routing-by-agreement schema. Firdaus et al. (2019) recommends a multi-task hierarchical approach using the CNN, RNN to get the contextual information and uses CRF for model label dependency. All the above approaches suffers from the lack of intent information to enhance slot task.

To circumvent the above limitation, exploration on slot prediction as joint task with intent gained momentum. Wang et al. (2020) proposes a new architecture for joint intent detection and slot filling based on pre-trained BERT (Chen et al., 2019), added the self-attention and slot gate with CRF which improvement in slot filling. Gangadharaiah and Narayanaswamy (2019) proposes attention information (calculating the attention of current encoder with respect to previous encoders), in addition to encoder during each decoder step for predicting joint intent and slot.

Inspired by performance of the above state of the art architectures in slot prediction work, we are exploring architectures namely BERT (Chen et al., 2019) and Encoder with Custom Decoder (Gangadharaiah and Narayanaswamy, 2019) on open source dataset with external list information.

We organize rest of the paper as follows. Section 2 describes the Proposed Approach. Section 3 describes the experimental setup including dataset, metrics used followed by results. Finally, we conclude and suggest future work and extensions.

2 Proposed Approach

2.1 Pre-Processor

All the external plist information are maintained in seperate files. Each file contains phrases asssociated to that plist. For example ”distanceunit” file contains phrases like ”kilometers”, ”km”, ”miles” and so on. These plists are generally extra information that the developer has defined to aid in slot identification for low corpus training data. We load all the plist information and store it in dictionary with key as phrase and value as list of plist(s) containing the phrase. Next, we gather all the subphrases from the given utterance and search in dictionary to find any matches. Then we filter the matches based on the following procedures

- We use ibo format to tag plist for the entire utterance.For example, consider the utterance ”show me pizza stores nearby”. The corresposnding tags will be ”o o b-businesscategory i-businesscategory o” if the business category contains ”pizza stores”.

- Filter the matches based on longest match for same plist. For example if the phrase ”Barbeque Nation” is matched by BusinessName as ”Barbeque” and ”Barbeque Nation”, then we choose ”Barbeque Nation” only as final match for BusinessName and tag it in ibo format.

- Keep the matches of all plist when subphrases within the phrase are matched. For example, consider the phrase ”punjabi thali”. ”punjabi thali” is present in ”BusinessName” and ”punjabi” in ”cuisinestyle”.We keep both the plist as candidate plist in IBO format as ”b-businessname__b-cuisinestyle i-businessname”.

- If more than one plist matches the same phrase, then we concatenate them by ”__”. For example if the phrase ”Barbeque Nation” matches ”BusinessName” and ”BusinessCategory”, then we concatenate the plist as ”BusinessName__BusinessCatory” in IBO format.

The above procedure is followed to get better candidate plist result for the utterance as this information influences slot prediction for each word in the utterance. Finally, we use this external plist sequence information in the model as explained in the next sections.

2.2 BERT Model

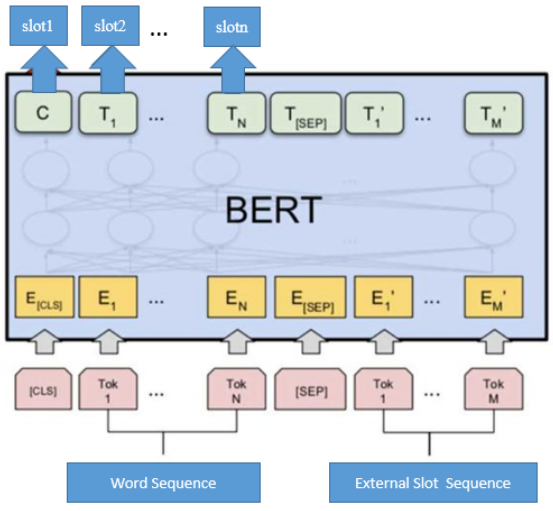

Figure 1: BERT Architecture with external plist sequence information and word sequence

Figure 1 explains BERT model architecture. BERT model is a multi-layer bi-directional Transformer encoder. The input to the model contains external plist sequence information or external slot sequence information, in addition to sentence. Since the plist can be unknown word to the vocabulary used for pre-trained BERT model, we added the list of unique plist used to the vocab list. The output of BERT provides learnt embeddings for both the word and its respective plist. To get the final embeddings for each word in sentence, we added a custom layer, which adds the embeddings of word and its plist. We pass these final embeddings to softmax layer to get the slots. We use pre-trained BERT model for the proposed experiment for the said method.

2.3 Bi-Directional Encoder with Custom Decoder

Gangadharaiah and Narayanaswamy (2019) proposed novel architecture using encoder and decoder model. Encoder module is made of Bi-directional LSTM encoder and Decoder module is made of LSTM module with attention information of input encoder during decoding stage at each step. We pass the output through dense layer. We have implemented the code from scratch based on the author description of the paper. During decoder implementation, we use attention information of encoder as additional input along with each encoder hidden information as input and calculate its importance using the modified equation as shown in Equation 6.

$$f_t = \sigma_g(W_f * x_t + U_f * h_{t-1} + V_f * a_t + b_f) \quad (1)$$

$$i_t = \sigma_g(W_i * x_t + U_i * h_{t-1} + V_i * a_t + b_i) \quad (2)$$

$$o_t = \sigma_g(W_o * x_t + U_o * h_{t-1} + V_o * a_t + b_o) \quad (3)$$

$$e_t = \sigma_g(W_c * x_t + U_c * h_{t-1} + V_c * a_t + b_c) \quad (4)$$

$$c_t = f_t * c_{t-1} + i_t * e_t \quad (5)$$

$$h_t = o_t * \sigma_c(c_t) \quad (6)$$

Where a_t represent attention information of hidden encoder h_t at time t and x_t represent input hidden encoder information at time t. We modified the input to the model to take both utterance and external plist sequence information. We converted utterance, containing list of words, to list of word indices based on dictionary (built by taking unique words from training data, sort the unique words and adding "unkword" in the end of sorted dictionary). If word is not present in dictionary, we assign "unkword" index. To maintain uniform length while training we pad sentence to max_length. We also construct weight matrix by assigning 300 dimensional vector to each row index, representing word in sorted dictionary. We obtained this 300 dimensional vector using glove embedding. If word is not in glove embedding, we assign "unkword" embedding which is randomly initialized 300 dimensional vector. We pass through Embedding module with utterance matrix and weight matrix to get 3-dimensional word embedding matrix.

We converted external plist sequence information as external slot embedding where each plist is assigned randomly initialized vector pdim (plist_dimension). As each word can have more than one plist assigned, we create matrix of plist embedding vector for each word. Hence, matrix for the external plist sequence information will be 4-dimensional (batch_size, max_sequence_words, distinct_plist_count, plist_dimension). We resize to 3D flattening the features of plist embedding from (distinct_plist_count, plist_dimension) to (distinct_plist_count * plist_dimension). We then concatenate word embedding matrix with external slot embedding matrix and pass as input to Encoder Module as shown in Figure 2.

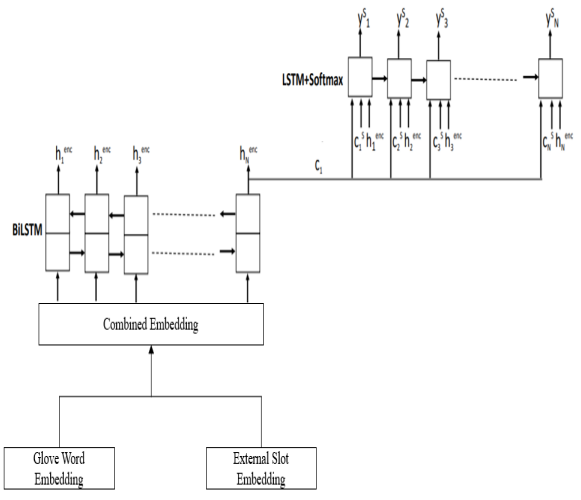

Figure 2: Bi-directional Encoder with Custom Decoder. Here we concatenate the word-embedding matrix with external slot embedding matrix and we send the combined embedding to Encoder module.

3 Experiments

Data	Train	Test	Slot
Business Search	301	1001	38

Table 1: BusinessSearch Data details.

We evaluated above models in BusinessSearch[1] Data as shown in Table 1. "BusinessSearch" data deals with details, search of Business names like dominos, reliance store and Business categories like grocery, gym with the help of external plist. It contains 301 training data and 1001 test data. It contains plist files like name, category for external use by the models.

3.1 Training Details

We share training details for BERT and Bi-directional encoder with Custom Decoder in below sections.

3.2 BERT

We used pre-trained 12 layer un-cased BERT model as initial start point for training model. We added plist names into vocab so that pre-processor does not tokenize the pist info and use it as it is. We use "Tensorflow" platform with optimizer as "Adam", loss as "categorical crossentropy", batch size as 64 and learning rate as 0.001.

3.3 Bi-directional Encoder with Custom Decoder

We use 300 dimension Glove Embedding vector for each word to construct training matrix of word index to vector. We experimented pdim with 16, 32, and 64 randomly initialized vector for each plist. We use LSTM hidden units as 128 in Encoder and Decoder (Custom LSTM with Attention information) hidden dimension as 128. Attention information does not change the hidden dimension information. Dense layer hidden dimension is distinct slots size with "Softmax" activation. We use "Keras" platform with optimizer as "Adam", loss as "categorical crossentropy", batch size as 64 and learning rate as 0.001.

4 Results and Analysis

Table 2 shows the comparison of the different architectures on "BusinessSearch" data with external plist information. From the table, we are able to infer that Bi-LSTM Encoder with Custom Decoder is able to beat BERT model where we feed external plist information as second sequence. This is attributed to the fact that concatenation of plist features along with word embedding is able to perform better than BERT model where we feed external plist information as second sequence. In addition, when we represent plist embedding dimension with 64 we are able to get better accuracy than 16 and 32. This shows that Bi-directional LSTM is able to differentiate better between different slots when plist representation is higher. In addition, we understand the importance of role of external plist as its absence lead to poor generalization of the model.

Architecture	Sentence Level Accuracy
Bi-LSTM Encoder with Decoder and 16 external slot embedding	84.79
Bi-LSTM Encoder with Decoder and 32 external slot embedding	86.83
Bi-LSTM Encoder with Decoder and 64 external slot embedding	90.49
BERT with external slot sequence as 2nd sequence	77

Table 2: Comparison of state of the art models.

5 Conclusion

This work demonstrated the use of external slot information along with sentence. We showed the performance of DNN models on low corpus data with external plist and showed there is an improvement of 13.49%, by Bi-directional Encoder with custom Decoder when compared to state of the art BERT model. We believe the pipeline approach to such user-developed dataset will aid in better model generalization for semantic slot prediction. Future scope of the paper includes exploration of non-sequential models for sequence labelling task. Also we are planning to extend the work to predict domain, intent along with slot prediction.

[1] https://github.com/MultiIntentData/LowCorpusSlotData

References

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

Mauajama Firdaus, Ankit Kumar, Asif Ekbal, and Pushpak Bhattacharyya. 2019. A multi-task hierarchical approach for intent detection and slot filling. *Knowledge-Based Systems*, 183:104846.

Rashmi Gangadharaiah and Balakrishnan Narayanaswamy. 2019. Joint multiple intent detection and slot labeling for goal-oriented dialog. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 564–569.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Aaron Jaech, Larry Heck, and Mari Ostendorf. 2016. Domain adaptation of recurrent neural networks for natural language understanding. *arXiv preprint arXiv:1604.00117*.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. Leveraging sentence-level information with encoder lstm for semantic slot filling. *arXiv preprint arXiv:1601.01530*.

Zihan Liu, Genta Indra Winata, Peng Xu, and Pascale Fung. 2020. Coach: A coarse-to-fine approach for cross-domain slot filling. *arXiv preprint arXiv:2004.11727*.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2014. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Tulika Saha, Sriparna Saha, and Pushpak Bhattacharyya. 2018. Exploring deep learning architectures coupled with crf based prediction for slot-filling. In *International Conference on Neural Information Processing*, pages 214–225. Springer.

Yangyang Shi, Kaisheng Yao, Hu Chen, Dong Yu, Yi-Cheng Pan, and Mei-Yuh Hwang. 2016. Recurrent support vector machines for slot tagging in spoken language understanding. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 393–399.

Youhyun Shin, Kang Min Yoo, and Sang-goo Lee. 2018. Slot filling with delexicalized sentence generation. In *INTERSPEECH*, pages 2082–2086.

Congrui Wang, Zhen Huang, and Minghao Hu. 2020. Sasgbc: Improving sequence labeling performance for joint learning of slot filling and intent detection. In *Proceedings of 2020 the 6th International Conference on Computing and Data Engineering*, pages 29–33.

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip S Yu. 2018. Joint slot filling and intent detection via capsule neural networks. *arXiv preprint arXiv:1812.09471*.