Exploring Slot Filling Method for Task Oriented Spoken Language Understanding

Project of CS3602 Natural Language Processing, 2023 Fall, SJTU

Xiangyuan Xue, Chenrun Wang, Zeyi Zheng

School of Electronic Information and Electrical Engineering

Abstract

Spoken Language Understanding (SLU) is one of the key modules in spoken dialogue systems, which interprets spoken language into semantic information. At the core of SLU is the concept of slot filling, which translates sentences into semantic structures. In this project, we focus on exploring the slot filling technique for task-oriented spoken language understanding, especially using sequence labeling algorithms founded on deep learning. We design an encoder-decoder framework for solving the slot filling problem, which consists of word embedding, context encoder, label decoder and entity refinement. We conduct a series of experiments to compare and analyze the performance of different deep learning architectures. We also conduct ablation studies to verify the effectiveness of rule-based processing tricks. Through these efforts, we expect to provide a better insight into different deep learning models and tricks for slot filling tasks.

1 Introduction

As speech-based interfaces become increasingly ubiquitous, enabling smooth and effective human-computer interaction relies significantly on the advancements in Spoken Language Understanding (SLU). In essence, SLU is a sub-field of computational linguistics that deals with equipping machines with the ability to comprehend and act upon spoken language. The prevailing relationship between SLU and dialogue systems underscores that an efficient SLU is cardinal for facilitating dialogue systems [19] to interpret user intentions appropriately, thereby executing relevant actions.

At the core of SLU is the concept of slot filling [8], which is vital for the interpretation of sentences into semantic structures. This process involves breaking down a sentence and extracting meaningful triples of action, slot and value, which represent the category of information and the actual piece of information relevant to the slot. Through this formulation, SLU systems can grasp the semantics of user utterances, transforming natural language into structured representations that are palpable for dialogue processing. In recent years, there have been significant advances in the field of SLU, with several remarkable approaches proposed. For instance, [14] employing bidirectional network architectures, [15] utilizing transformer structures, and [1] leveraging BERT networks have demonstrated impressive performance in both intent classification and slot filling tasks.

In this project, we focus on exploring the slot filling technique for task-oriented spoken language understanding, especially using sequence labeling algorithms [7] founded on deep learning. Our study pivots around several key contributions to the realm of SLU by enhancing the efficacy of the slot filling technique. We delve into the application of sequence labeling algorithms to improve the accuracy of slot filling tasks. Then we design an encoder-decoder framework for solving the slot filling problem. We also conduct a series of experiments on different deep learning architectures to dissect and analyze their performance rigorously. Through these efforts, we aim to unearth insights into the characteristics and efficacies of each model concerned with slot filling challenges. More-over, we conduct a detailed exploration into the effectiveness of rule-based processing, outlining the impacts of text correction and entity refinement on the overall SLU system. By performing ablations on these processes, we intend to discern their contributions and potential in fortifying the accuracy and reliability of slot filling methodologies.

2 Method

In this project, we are exploring the slot filling method, which views the spoken language understanding task as a sequence labeling problem [7]. Given a sentence from the speech recognizer, the slot filling method aims to label each word with a slot tag, which indicates the type of the word and whether it is the start of the entity. Specifically, we are using the BIO tagging scheme, where B indicates the beginning of an entity, I indicates the inside of an entity, and O indicates the outside of an entity. Then we can easily extract the semantics from the labeled sentence by merging the words with the same tag, with the form of action-slot-value triples.

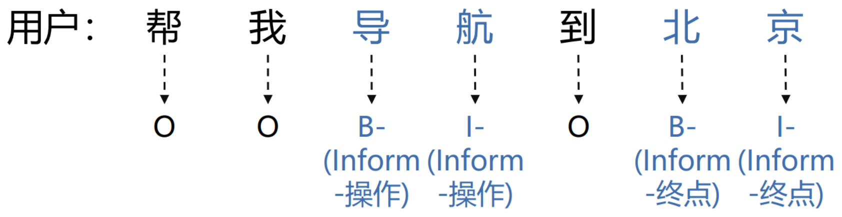

Figure 1: An example of sequence labeling for slot filling. Each word is labeled with a tag, which indicates the action and slot of the word and whether it is the start of the entity, where B indicates the beginning of an entity, I indicates the inside of an entity, and O indicates the outside of an entity.

To solve this sequence labeling problem, we utilize the sequence-to-sequence paradigm [17], which is one of the most popular methods in deep learning nowadays. Generally, it consists of an encoder and a decoder. The encoder converts the context into a hidden representation, based on which the decoder generates the outputs. We divide our framework into three parts: word embedding, context encoder, and label decoder. In addition, rule-based pre-processing and post-processing can be applied to significantly improve the performance, such as text correction and entity refinement.

2.1 Word Embedding

Previous works have shown that one-hot encoding is not a good choice for word representation, since it cannot capture the semantic information of the words, while mapping the words into a continuous vector space can significantly improve the performance of the language models [13, 12]. In this project, we explore two word embedding methods. For the simple models based on RNNs and Transformers, we directly use the provided word embedding, which maps each word into a 768-dimensional vector. For the models with pretrained language models as backbones, we use the tokenizer from the pretrained language models to convert the sentence into word embeddings. Note that the mainstream pretrained language models tend to use subword units, which is a bit different from the setting in this project. Hence, we insert a blank space before each word to force the tokenizer treat each word as a single token, thus aligning the word embeddings with the slot tags. This may lead to a slight performance drop, but it is necessary for applying pretrained models.

2.2 Context Encoder

With the development of deep learning, the context encoder has evolved from MLPs [16] to CNNs [9], RNNs [5] and even Transformers [18], where RNNs and Transformers have been proved effective for sequence modeling. In this project, we explore multiple architectures for the context encoder to compare the performance and analyze the advantages and disadvantages of different architectures.

Recurrent Neural Network. Proposed in the 1980s, RNNs [5] are one of the most popular models for sequence modeling. The key idea of RNNs is to use the hidden state to capture the context information, which is updated at each time step. The hidden state at time step t is computed as

$$\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t), \tag{1}$$

where $\boldsymbol{x}_t$ is the input at time step t, $\boldsymbol{h}_{t-1}$ is the hidden state at time step $t-1$, and f is a non-linear function. A common choice of f is the hyperbolic tangent function, which is computed as

$$f(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t) = \tanh(\boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{W}_x \boldsymbol{x}_t + \boldsymbol{b}), \tag{2}$$

where $\boldsymbol{W}_h$, $\boldsymbol{W}_x$, and $\boldsymbol{b}$ are the parameters to be learned. The hidden state at the last time step is used as the context representation.

In this project, we also try to use Long Short-Term Memory (LSTM) [6] and Gated Recurrent Unit (GRU) [2] as the improved units of RNNs. LSTM applies multiple gates to control the information flow, maintaining a long-term memory. GRU modifies the design of the gates to simplify the structure of LSTM, bringing faster training speed while maintaining comparable performance.

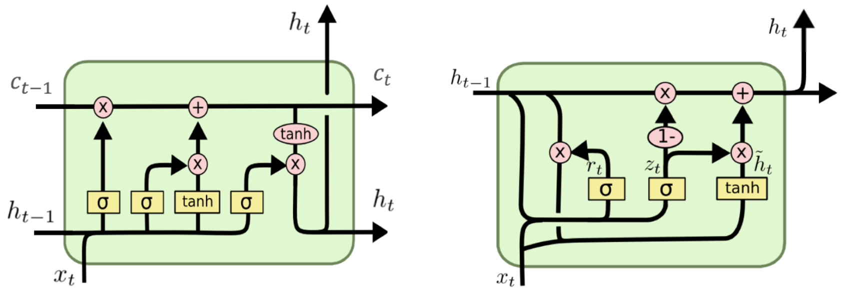

Figure 2: The structure of the LSTM and GRU cells. LSTM applies multiple gates to control the information flow, maintaining a long-term memory. GRU modifies the design of the gates to simplify the structure of LSTM, bringing faster training speed while maintaining comparable performance.

Transformer. In recent years, transformers [18] have emerged as a novel architecture for sequence modeling, achieving state-of-the-art results in multiple domains. Transformers show strong capabilities in capturing long-range dependencies and modeling complex relationships, which makes learning from tremendous amounts of data feasible. The core of transformers is the attention mechanism, which computes the weighted sum of the values based on the similarity between the query and the keys. The attention mechanism can be formulated as

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V}, \tag{3}$$

where $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ are the query, key and value matrices respectively, and d_k is the dimension of the key. The transformer encoder is composed of multiple layers of multi-head attention and feed-forward networks. The key idea of multi-head attention is to project the query, key and value into multiple subspaces and extract the dependencies from different representations, which enhances the representation ability of the model. In this project, we only use the transformer encoder, since the transformer decoder is not necessary for sequence labeling.

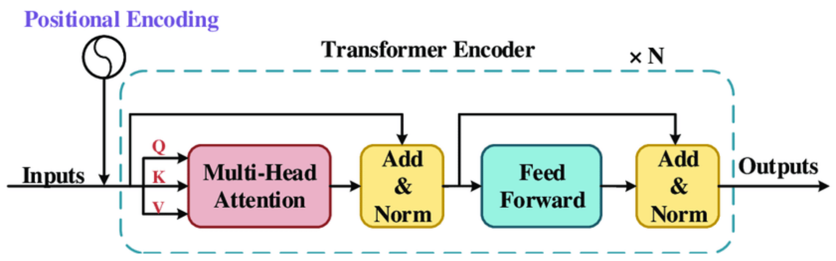

Figure 3: The structure of the transformer encoder. The transformer encoder is composed of multiple layers of multi-head attention and feed-forward networks. Positional embedding is applied to encode the positional information of the inputs.

Note that the transformer encoder is not a recurrent architecture, which cannot capture accurate temporal information. To solve this problem, positional embedding is applied to encode the positional information of the inputs. We follow the original paper [18] to use sine and cosine functions with different frequencies as the positional embedding in this project.

Bidirectional Encoder Representations from Transformers. BERT [4] is a novel language representation model, which is pretrained on a large-scale corpus and fine-tuned for downstream tasks. BERT is based on the transformer encoder, which is composed of multiple layers of multi-head attention and feed-forward networks. The key idea of BERT is to mask some words in the sentence and predict the masked words based on the context. In addition, BERT also uses the next sentence prediction task to capture the relationship between sentences. BERT has achieved state-of-the-art results in multiple tasks, such as question answering, natural language inference, and named entity recognition. In this project, we explore using BERT as the context encoder for sequence labeling to exploit the power of pretrained language models.

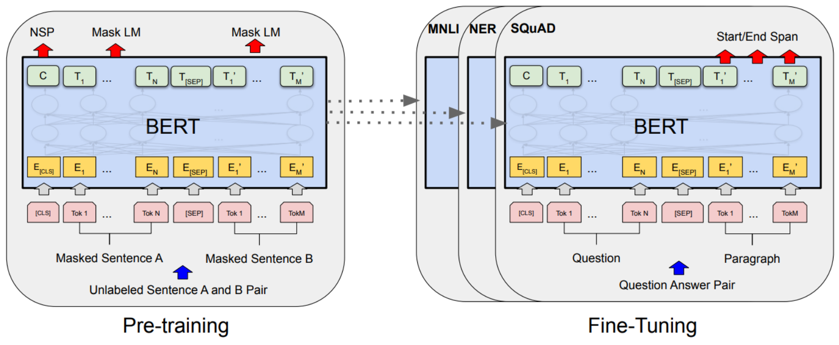

Figure 4: The overview of BERT. BERT is based on the transformer encoder, which is composed of multiple layers of multi-head attention and feed-forward networks. As a mainstream backbone, BERT is pretrained on a large-scale corpus and fine-tuned for downstream tasks.

Note that the original BERT model is pretrained mainly on the English corpus, which may not be suitable for Chinese tasks. In this project, we utilize a variant of BERT pretrained on the Chinese corpus [3], which is promised to be more effective for Chinese tasks. To apply BERT as an encoder, we extract the hidden states of the last layer as the context representation.

2.3 Label Decoder

Besides a context encoder, we also require a decoder to predict the slot tags based on the context representation. In this project, we explore three different architectures as the label decoder.

Multi-Layer Perceptron. Since the number of slot tags is relatively limited, we believe that MLPs [16] are a simple and effective choice for the label decoder. The context representation at each position is fed into a fully-connected network to predict the corresponding slot tag. Here we use hyperbolic tangent as the activation function to preserve the negative information. A softmax layer is applied to normalize the outputs into a probability distribution over all the slot tags. Cross-entropy loss is used as the training objective, which is computed as

$$\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij} \log \hat{y}_{ij}, \tag{4}$$

where $\boldsymbol{y}$ is the label, $\hat{\boldsymbol{y}}$ is the prediction, N and K represent the number of samples and tags.

Recurrent Neural Network. Although MLPs have enough capacity to learn the mapping from the context representation to the slot tags, they fail to model the dependencies between the outputs, which is of vital importance for sequence labeling. Hence, we explore adding RNNs [5] before the MLPs to improve the performance. We utilize bidirectional RNNs to fully exploit the context information. Similarly, a softmax layer is applied to form a probability distribution over all the slot tags. Cross-entropy loss is used as the training objective as well.

Conditional Random Field. Purposed early in 2001, Conditional Random Field (CRF) [10] is a powerful discriminative model for sequence labeling. CRF models the conditional probability of the outputs given the inputs as a log-linear model, which is formulated as

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp\left[\sum_{t=1}^{T}\sum_{k=1}^{K} \lambda_k f_k(\boldsymbol{y}_{t-1}, \boldsymbol{y}_t, \boldsymbol{x}, t)\right], \tag{5}$$

where $\boldsymbol{x}$ is the input sequence, $\boldsymbol{y}$ is the output sequence, T is the length of the sequence, K is the number of features, f_k is the k-th feature function, and λ_k is the weight of the k-th feature. $Z(\boldsymbol{x})$ is the normalization term, which is computed as

$$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y}'} \exp\left[\sum_{t=1}^{T}\sum_{k=1}^{K} \lambda_k f_k(\boldsymbol{y}'_{t-1}, \boldsymbol{y}'_t, \boldsymbol{x}, t)\right], \tag{6}$$

where $\boldsymbol{y}'$ is a possible output sequence. In this project, we use a CRF layer as the label decoder to model the dependencies between the outputs, expecting to improve the performance of the model. The CRF layer is trained with the negative log-likelihood loss, which is computed as

$$\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\log p(\boldsymbol{y}|\boldsymbol{x}). \tag{7}$$

which is different from the cross-entropy loss used for MLPs and RNNs.

2.4 Entity Refinement

Admittedly, deep learning models are powerful in modeling complex relationships, but they are not perfect. Since all the input texts are from the speech recognizer, the lack of language knowledge tends to bring many errors such as homophones and typos, which largely limits the accuracy of slot filling. Hence, we explore two tricks to solve this tough problem.

Entity Refinement. Another way to eliminate the errors is entity refinement, which is commonly used in spoken language understanding tasks. The key idea of entity refinement is to use the slot tags to extract the entities from the sentence, and then use the entities to correct the slot tags. In this project, we make use of the provided lexicon to refine the entities. Specifically, we extract the entities from the slot tags and check whether they are in the lexicon. If not, we try to find the most similar entity in the lexicon to replace it. The similarity is measured by the Levenshtein distance [11], which means the minimum number of edits (insertions, deletions or substitutions) required to change one word into the other. This trick is expected to eliminate the errors caused by typos and homophones effectively. We will conduct experiments to prove that in Section 3.

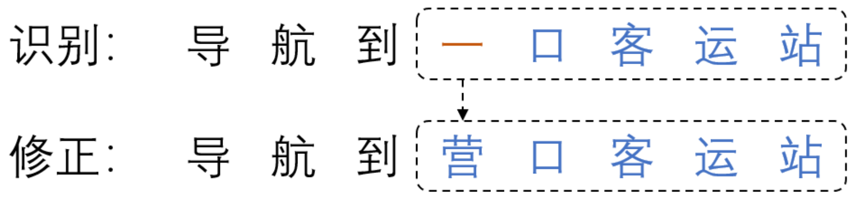

Figure 5: An example of entity refinement. The entities are extracted from the slot tags and checked whether they are in the lexicon. If not, the most similar entity in the lexicon is used to replace it. The similarity is measured by the Levenshtein distance, i.e. the minimum edit distance.

Text Correction. Since the manual transcripts are provided in the training data, we can use them instead of the results of automatic speech recognition. Note that such trick cannot be applied in the test data, where the manual transcripts are not available in practice. This brings a new problem of domain adaptation [20], i.e. the training set has a different distribution from the test set. Luckily, the problem is not that severe in this problem setting. We will conduct experiments to show the effectiveness of this trick in Section 3.

3 Experiments

3.1 Experiment Setup

Data Processing. For the sake of reproducibility, all the training data are from the provided dataset, including a training set of 5119 examples, a development set of 895 examples and an unlabelled test set of 3 examples. We search the tags in the texts to form the labelled sequences, which correspond to the single words in the texts. The vocabulary is also built from the dataset for word embedding.

Implementation details. According to Section 2, we implement multiple models to respectively test their performance in the slot filling task, including RNN-MLP, Transformer-MLP, RNN-CRF, BERT-MLP, BERT-RNN-MLP, BERT-CRF, and BERT-RNN-CRF, which are named in the encoder-decoder form. Note that RNN-MLP is the baseline model and we add no tricks. For the other models, we enable entity refinement by default, while text correction is not used for fair comparison. For the BERT-based models, we utilize a variant of BERT called pretrained on the Chinese corpus [3]. The pretrained weights can be automatically downloaded from the website.

All the models are trained for 100 epochs with a batch size of 32, which is proved enough for convergence, as is shown in Figure 6. We use Adam optimizer for RNN-based models and AdamW optimizer for Transformer-based models. The learning rate is set to 0.001 for training, 0.002 for warmup and 0.00002 for finetuning. A common trick is used here for training BERT-based models. The pretrained BERT model is frozen at first and we use a large learning rate to train the output layer. After several epochs, we unfreeze the BERT model and finetune the whole model with a small learning rate. This trick is proved to be effective in reducing the training time and has been widely used in previous works [4]. All the models have small to medium number of parameters, which can be trained on a single NVIDIA GeForce RTX 3090 GPU within 15 minutes.

Evaluation metrics. To evaluate the performance of the models, we conduct inference on the development set to compute the loss, accuracy, precision, recall and F1-score, where accuracy requires all the tags in a sentence to be correctly predicted, while precision and recall are relatively loose metrics. Both Accuracy and F1-score are effective metrics, but the loss is a little tricky since MLP and CRF decoder utilize different loss functions. Hence, it sometimes cannot reflect the actual performance.

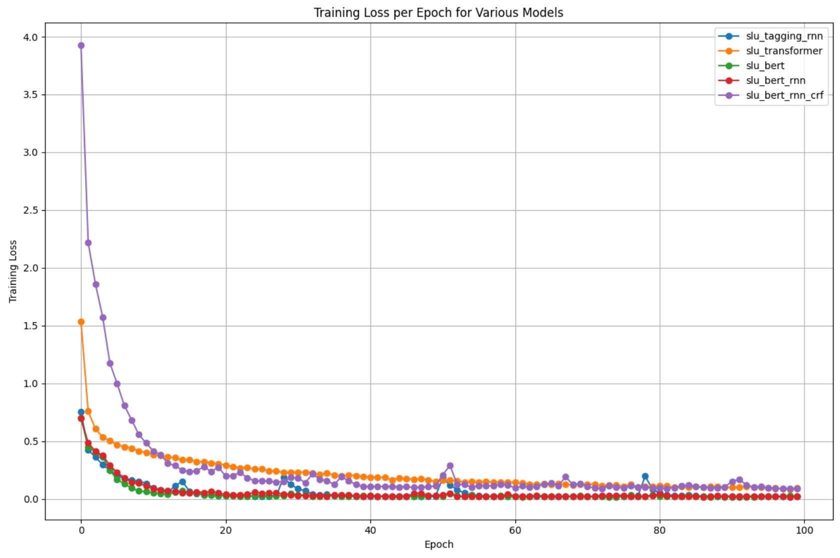

Figure 6: The loss curve of the models on the training set. All of the curves converge quickly and smoothly, which indicates that the models are well trained within 100 epochs. For the BERT-based models, we first warmup the output layer for several epochs and then finetune the whole model.

3.2 Model Performance

Following the experiment setup, we train the models and evaluate their performance on the development set. The results are shown in Table 1. We can see that the baseline achieves an accuracy of 71.40%, which is actually not bad considering the simplicity of the network architecture. By using transformer encoder, the accuracy can be improved to 74.41%. This result owes to the powerful representation ability of transformers, but it does not show a significant advantage due to the small size of the dataset. Using CRF as the decoder also provides a slight improvement, as it can capture the dependencies between tags in the output layer.

Compared to the slightly improved baseline, the introduction of BERT brings a significant improvement in performance. By simply replacing the encoder with BERT, the accuracy reaches 80.34%, which indicates the strong modeling ability of pretrianed language models. By introducing prior knowledge in the specific language, the model can be quickly adapted to the slot filling task. Including extra RNN and CRF layers can further improve the performance, where BERT-BiLSTM-CRF achieves the sota performance with an accuracy of 81.68%.

We claim that the results are reasonable and reliable, while achieving a satisfactory performance on the slot filling task. The results show that by composing powerful deep neural networks such as BERT, RNN and CRF, we can increasingly improve the performance of the model and finally build a robust system for SLU tasks.

Table 1: The performance of the models on the development set. Multiple metrics are used, including loss, accuracy, precision, recall and F1-score. The BERT-BiLSTM-CRF model achieves the sota performance with an accuracy of 81.68% and an F1-score of 0.8517. The BERT-CRF model also achieves satisfactory performance with an accuracy of 81.23% and an F1-score of 0.8505.

Model	**Development Metrics**				
	Loss ↓	**Accuracy ↑**	**Precision ↑**	**Recall ↑**	**F1-Score ↑**
BiRNN-MLP	0.5293	70.73%	0.7487	0.7487	0.7487
BiLSTM-MLP	1.3285	71.40%	0.7991	0.7424	0.7697
BiGRU-MLP	1.0457	71.17%	0.7993	0.7435	0.7704
Transformer-MLP	0.6029	74.41%	0.8418	0.7769	0.8080
BiRNN-CRF	3.2606	74.41%	0.8121	0.7706	0.7908
BiLSTM-CRF	5.0666	73.97%	0.8561	0.7633	0.8071
BiGRU-CRF	3.9188	74.19%	0.8466	0.7769	0.8102
BERT-MLP	0.6988	80.34%	0.8591	0.8332	0.8460
BERT-BiRNN-MLP	0.6010	80.11%	0.8564	0.8332	0.8446
BERT-BiLSTM-MLP	0.6409	80.11%	0.8616	0.8311	0.8461
BERT-BiGRU-MLP	0.5840	80.34%	0.8687	0.8279	0.8478
BERT-CRF	4.0192	81.23%	0.8619	0.8394	0.8505
BERT-BiRNN-CRF	2.1814	80.45%	0.8633	0.8363	0.8496
BERT-BiLSTM-CRF	**1.5122**	**81.68%**	**0.8622**	**0.8415**	**0.8517**
BERT-BiGRU-CRF	1.5779	80.00%	0.8674	0.8321	0.8494

3.3 Ablation Studies

We also carry out two ablation studies to prove the effectiveness of the proposed rule-based preprocessing and post-processing methods. We select one model for each type of encoder-decoder architecture, including Transformer-MLP, BiLSTM-CRF, BERT-MLP, BERT-BiLSTM-MLP, BERT-CRF and BERT-BiLSTM-CRF. We test their performance on the development set with and without the specific tricks to prove their effectiveness.

Entity Refinement. The results are shown in Table 2. We can see that by introducing entity refinement, all the models gain an obvious improvement in accuracy. This is because the sentence provided by the ASR system is not accurate enough and contains many errors, but entity refinement empowers the model to correct the errors and thus improves the performance. However, the loss remains almost unchanged, which is reasonable since the trick does not change the loss function. Such a phenomenon also reveals that the loss is not directly related to the performance of the model.

Text Correction. The results are shown in Table 3. We can see that by introducing text correction, all the models gain a significant improvement in accuracy, since the manual transcripts completely eliminate the errors in the ASR system and enable the model to learn on the clean data. The loss also decreases to a large extent due to the same reason. Although the accuracy reaches 96.54% with the BERT-CRF model, we should note that it is not a fair comparison because we have no access to the manual transcripts on the test set. However, such results reveal the potential of improving the ASR

Table 2: The performance of the models with and without entity refinement. All the models gain an obvious improvement in accuracy, but the loss remains almost unchanged.

Model	**Loss** $\downarrow$			**Accuracy** $\uparrow$		
	w/o	**w/**	$\boldsymbol{\Delta}$	**w/o**	**w/**	$\boldsymbol{\Delta}$
Transformer-MLP	0.5690	0.6029	+0.0339	72.29%	74.41%	**+2.12%**
BiLSTM-CRF	5.0666	5.0666	+0.0000	71.96%	73.97%	**+2.01%**
BERT-MLP	0.6988	0.6988	+0.0000	78.32%	80.34%	**+2.02%**
BERT-BiLSTM-MLP	0.4096	0.6409	+0.2313	77.99%	80.11%	**+2.12%**
BERT-CRF	3.9067	4.0192	+0.1125	79.33%	81.23%	**+1.90%**
BERT-BiLSTM-CRF	2.0721	1.5122	**−0.5599**	78.55%	81.68%	**+3.13%**

system to further enhance the performance of the model. Another promising direction is to design a robust correction model to automatically correct the errors in the ASR system.

Table 3: The performance of the models with and without text correction. All the models gain a significant improvement in accuracy. The loss also decreases to a large extent.

Model	**Loss** $\downarrow$			**Accuracy** $\uparrow$		
	w/o	**w/**	$\boldsymbol{\Delta}$	**w/o**	**w/**	$\boldsymbol{\Delta}$
Transformer-MLP	0.6029	0.1945	**−0.4084**	74.41%	93.97%	**+19.56%**
BiLSTM-CRF	5.0666	1.4406	**−3.6260**	71.96%	94.53%	**+22.57%**
BERT-MLP	0.6988	0.1742	**−0.5246**	78.32%	95.75%	**+17.43%**
BERT-BiLSTM-MLP	0.6409	0.1382	**−0.5027**	80.11%	95.64%	**+15.53%**
BERT-CRF	4.0192	0.8253	**−3.1939**	81.23%	96.54%	**+15.31%**
BERT-BiLSTM-CRF	1.5122	0.9113	**−0.6009**	81.68%	95.98%	**+14.30%**

4 Conclusion

In this project, we explore the slot filling technique for task-oriented spoken language understanding based on sequence labeling algorithms. First, we introduce the background of SLU tasks and the slot filling technique. We investigate the mainstream deep learning models for sequence labeling, including RNN, Transformer, BERT and CRF. Then we design an encoder-decoder framework for solving the slot filling problem, which consists of word embedding, context encoder, label decoder and entity refinement. We conduct a series of experiments on different deep learning architectures to dissect and analyze their performance rigorously. We find that both Transformer and CRF can benefit the baseline model. We also find that using pretrained language models can significantly improve the performance, where the BERT-BiLSTM-CRF model achieves the best performance in the slot filling task with an accuracy of 81.68% and an F1-score of 0.8517 on the development set, while the BERT-CRF model also achieves satisfactory performance. In addition, we conduct two ablation studies to verify the effectiveness of entity refinement and text correction respectively. We find that both of the tricks can significantly improve the performance, but text correction brings unfair comparison to some extent, raising the accuracy to striking 96.54%. However, we admit the value of text correction and point out that it is a promising direction for future work.

References

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

[2] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[3] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, and Ziqing Yang. Pre-training with whole word masking for chinese bert. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3504–3514, 2021.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] Kazuya Kawakami. *Supervised sequence labelling with recurrent neural networks*. PhD thesis, Technical University of Munich, 2008.

[8] Kunho Kim, Rahul Jha, Kyle Williams, Alex Marin, and Imed Zitouni. Slot tagging for task oriented spoken language understanding in human-to-human conversation scenarios. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 757–767, 2019.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[10] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of international conference on machine learning*, 2001.

[11] Vladimir I Levenshtein et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.

[12] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[14] Peiqing Niu, Zhongfu Chen, Meina Song, et al. A novel bi-directional interrelated model for joint intent detection and slot filling. *arXiv preprint arXiv:1907.00390*, 2019.

[15] Libo Qin, Tailu Liu, Wanxiang Che, Bingbing Kang, Sendong Zhao, and Ting Liu. A co-interactive transformer for joint slot filling and intent detection. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8193–8197. IEEE, 2021.

[16] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[17] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[19] Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*, 2016.

[20] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.

A Member Contribution

Xiangyuan Xue implements the BERT-based models, checks the correctness of the code and writes the majority of the report. Chenrun Wang implements the Transformer-based models, finetunes the model hyperparameters and writes a part of the report. Zeyi Zheng implements the CRF-based models, runs the majority of the experiments and writes a part of the report.