Exploring Improvements in Chinese Semantic Slot Filling: from Data and Model Perspectives

Rui Tao [*1,2] **Puyi Wang** [*1,2] **Shuwen Wu** [*1,2]

Abstract

Spoken Language Understanding (SLU) plays a vital role in supporting voice assistants and dialogue systems. It consists of two main tasks: Intent Detection and Semantic Slot Filling (SSF). Traditional deep learning pipelines for SSF use an encoder-decoder framework to perform semantic sequence labeling on sentences. In this paper, we explore several approaches to enhance the model's ability in SSF: introducing pre-trained model, augmenting data, filtering noises, leveraging dialogue history and fusing word-chunk information. Our final proposed approach outperforms the baseline by 9.16% and the pure BERT model by 2.23% in terms of accuracy on the test set.

1. Introduction

Spoken Language Understanding is a core component of intelligent human-computer interaction systems, enabling machines to comprehend and extract semantic information from spoken input generated from Automatic Speech Recognition (ASR). SLU plays a vital role in supporting voice assistants and dialogue systems.

SLU consists of two main tasks: Intent Detection and Semantic Slot Filling (SSF) (Kim et al., 2019). While intent detection is a standard classification problem in which only one label is predicted for each sentence, slot filling is often formulated as a sequence labeling task (Xu & Sarikaya, 2013), where a sequence of labels need to be assigned jointly. In this paper, we focus on the task of SSF for spoken input.

The essence of the Slot Filling task is breaking down a sentence and understanding its semantics. The process includes extracting key information as `value`, labeling them with appropriate `act` and `slot`, and finally generating the semantic triples `(act, slot, value)`.

[*] Equal contribution [1]Shanghai Jiao Tong University [2]School of Electronic Information and Electrical Engineering. Correspondence to: Rui Tao <taorui_sjtu@sjtu.edu.cn>, Puyi Wang <wangpuyi@sjtu.edu.cn>, Shuwen Wu <mike0510@sjtu.edu.cn>.

Building upon the traditional deep learning pipeline, we explored the following methods:

- Using pre-trained models as encoder cell (BERT, MacBERT, RoBERTa) rather than RNN. And using RNN as decoder rather than fully connected networks.
- Augmenting training data from various aspects, such as replacing `POI` name to expand training set and filtering low-quality noisy training data.
- Leverage history information in the tagging process to merge context information.
- Leveraging the semantics of word chunks (Using `Jieba` to generate word chunks).

The outcome results demonstrate many of these approaches lead to significant improvements in accuracy and F1-score. We ultimately propose a model that integrates all the effective aforementioned methods, which achieves an improvement in accuracy $\uparrow$ **9.16% over traditional deep learning pipeline, and $\uparrow$ 2.23% over pure BERT model baseline**.

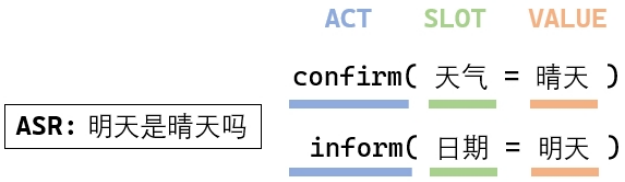

Figure 1. Example of semantic triplets.

2. Related Work

2.1. Semantic Slot Filling

Semantic Slot Filling (SSF) aims to extract structured semantic information from spoken input, and segment it into slots, which serve as the fundamental units of semantics. In this project, the task involves filling semantic triplets, as illustrated in Fig. 1.

Xu & Sarikaya (2013) pointed out that slot filling can be formulated as a sequence labeling task, where a sequence of labels need to be assigned jointly. Conditional random

field (CRF) (Lafferty et al., 2001) is a proven technique for slot filling. As opposed to other traditional methods for sequence labeling, such as maximum entropy markov models (MEMM) (McCallum et al., 2000), CRF directly models the global conditional distribution.

With the development of deep learning, neural network-based approaches for SSF have gradually replaced traditional statistical models. Typical deep learning methods include the use of RNN (Elman, 1990), as well as improved models like LSTM (Hochreiter & Schmidhuber, 1997) and Bi-LSTM (Graves & Schmidhuber, 2005), which are capable of modeling temporal dependencies in input sequences. Detailed principle and applications are discussed in the following section.

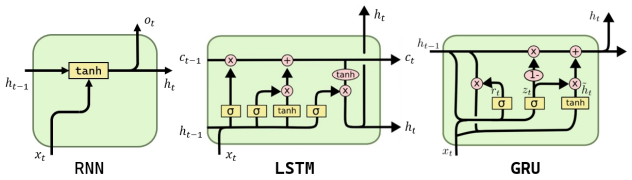

Figure 2. Comparison of structure of RNN, LSTM and its simplified version GRU.

2.2. Recurrent Neural Network

Recurrent Neural Networks RNN (Elman, 1990) have become a cornerstone in the field of SLU due to their ability to process sequential data effectively. Traditional RNN are designed to handle time-series data by maintaining a hidden state that captures information from previous time steps. The hidden state h_t at time step t is computed as:

$$h_t = \sigma(W_h h_{t-1} + W_x x_t + b_h)$$

where σ is the activation function, W_h and W_x are weight matrices, x_t is the input at time step t, and b_h is the bias term. RNN has been proven effective in SLU tasks (Guo et al., 2014), especially in slot-filling (Mesnil et al., 2015). However, RNN faces significant challenges in gradient vanishing problem, which hampers their ability to learn long-range dependencies in sequences.

Long Short-Term Memory LSTM (Hochreiter & Schmidhuber, 1997) networks were introduced to address the above limitations, by incorporating a memory cells and gating mechanisms. Key update equations for LSTM unit are:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad \text{(Forget gate)}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad \text{(Input gate)}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad \text{(Output gate)}$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad \text{(Candidate cell state)}$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad \text{(Cell state)}$$
$$h_t = o_t \odot \tanh(C_t) \quad \text{(Hidden state)}$$

where f_t, i_t, and o_t are the forget, input, and output gates, respectively. C_t is the cell state; and $\odot$ denotes element-wise multiplication. This memory-forgetting mechanisms makes LSTM particularly well-suited for tasks in SLU (Yao et al., 2014), where understanding text information among long-range is common. However, LSTM can only capture information from the previous text, and unable to integrate subsequent context. The introduction of Bi-LSTM (Graves & Schmidhuber, 2005) addresses this issue by incorporating a bidirectional recurrent network, allowing each part of the sequence to access information from the entire context. Bi-LSTM has been proven effective in several NLP tasks, such as sentence type classification (Chen et al., 2017).

Gated Recurrent Units GRU (Cho et al., 2014) is a simpler alternative to LSTM, which combines the forget and input gates into a single update gate. Comparison between these structures is shown in Fig 2.1. Key update equations are:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \quad \text{(Update gate)}$$
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \quad \text{(Reset gate)}$$
$$\tilde{h}_t = \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h)$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \quad \text{(Hidden state)}$$

where z_t and r_t are the update and reset gates, respectively. Studies have shown that GRU can achieve comparable results to LSTM, often with faster training times and fewer parameters (Chung et al., 2014).

2.3. Pretrained Model–BERT

Pretrained Model Language model pre-training involves training a model on large datasets for general language understanding. The model can later be fine-tuned for given tasks by exposing it to task-specific data. Methods that leverage word embeddings, such as Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), effectively improving the performance on downstream tasks. Nonetheless, such algorithms require apropos pretraining and only perform outstandingly in a fraction of NLP tasks. Since then, the pretraining and finetuning paradigm has emerged. Generative pre-training demonstrated the potential of pre-training methods for diverse downstream tasks by groundbreakingly utilizing uni-dimensional transformers as the backbone (Radford & Narasimhan, 2018).

Bidirectional Encoder Representations from Transformers BERT (Devlin et al., 2019) represents a significant advancement in NLP. In contrast to previous models that processed text in a unidirectional manner, it's designed to capture context from both directions, by training on a bidirectional transformers (Vaswani et al., 2017) structure.

This bidirectional approach allows BERT to better understand the relationships between words in a sentence, leading to improved performance across a wide range of NLP tasks.

BERT for Chinese Despite BERT's high effectiveness for English, its application to Chinese presents unique challenges due to the structural differences between the two languages. In contrast to English where words are typically separated by spaces, Chinese is a logographic language with no clear word boundaries. This characteristic makes tokenization more complex in Chinese.

This paper involves 3 version of BERT adapted specifically for Chinese. **BERT-Chinese** (Sun et al., 2021) is a version of BERT specifically trained on Chinese text. Bert-Chinese leverages a WordPiece tokenization method designed to effectively handle the intricacies of Chinese characters and word segmentation. **Mac-BERT** (Cui et al., 2020) is an improved version of BERT specifically tailored for Chinese text, which modifies the pre-training tasks and tokenization process. **Ro-BERTa** (Liu et al., 2019) is a robustly optimized version of BERT. RoBERTa eliminated the Next Sentence Prediction task and was trained on much larger datasets, longer sequences and greater number of steps.

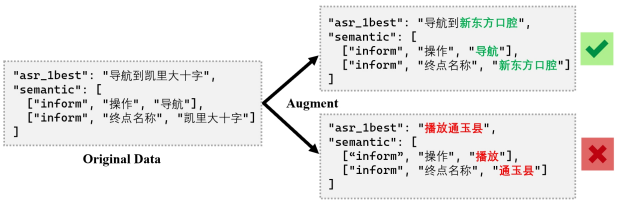

Figure 3. A data augmentation example, replacing the `location` part only. Otherwise, for example, if we randomly replace all the slot-value pairs simultaneously, `action=navigate` may be replaced with `action=Play` and `location=Kaili Dashi Crossroad` changes to `location=Tongyu County`, which results in a sentence with no sense. Such semantically incorrect examples would be detrimental to model training.

2.4. Data Augmentation

Data augmentation refers to techniques that artificially expand the size and diversity of a dataset by generating new training samples from existing ones. In the context of Spoken Language Understanding (SLU), data augmentation is particularly valuable due to the challenges of limited labeled data and the presence of noise introduced by Automatic Speech Recognition (ASR) systems. Research has shown that data augmentation can significantly improve the robustness and generalization capabilities of SLU models, especially when combined with pre-trained language models like BERT. (Wei & Zou, 2019).

3. Exploration

In this section, we study several methods to improve model performance. Broadly speaking, explorations are from data perspectives and model perspectives.

```
{
    "utt_id": 6,
    "manual_transcript": "你是乖孩子吗(unknown)",
    "asr_1best": "孩子吗不是",
    "semantic": []
},
```

Figure 4. Noisy data example.

At the data level, we investigate various data augmentation techniques, data cleaning methods, and the integration of dialogue history. Ultimately, we demonstrate that **randomly replacing `POI` names to generate new training data is an extremely effective method**.

At the model level, we introduce the pre-trained BERT model as the encoder, which leads to significant improvements. We further explore different BERT models, varying fine-tuning degrees, and decoder cell choices on performance. Our findings show that **fine-tuning $80\% - 100\%$ of the BERT parameters yields excellent results for our SSF task, while the choice of decoder is not critical**, a simple FNN is sufficient to achieve satisfactory performance.

3.1. Data Augmentation

Motivation. To enhance the generalization capability of the model, we employed data augmentation to generate more data from a limited dataset, thereby increasing the quantity and diversity of training samples and improving the model's robustness.

Specifically, we adopted synonym replacement as our data augmentation strategy. Synonym replacement is a commonly used NLP data augmentation technique that involves substituting certain words in the text with their synonyms or similar terms to generate additional training samples. This approach not only enriches the diversity of the data but also helps the model gain a deeper understanding of the relationships and semantics between words. In SLU model training, improving semantic understanding is particularly crucial. By performing synonym or similar-word replacement on sentences, this method effectively enhances the model's generalization capability, enabling it to perform better across different scenarios.

Methodology. We **only** replace those slot values categorized in `location`, with names sourced from the `poi_name.txt`. The process is shown in Fig 3. In this process, we change the `location` part only, regardless of other slot classes. Because replacing `location` doesn't alter the overall semantics of the sentence. In contrast, **replacing all *slot-value* pairs at same time might generate a semantically incorrect sentence or makes no sense**, as shown in Fig 3. This assumption is further validated in the following experiment, where replacing too much slots at

same time brings no positive effect.

We firstly identify slot-value pairs that categorized as `locations` within the `semantic` items. Then randomly select a replacement from the `poi_name.txt` and simultaneously replace the corresponding word in the `asr_1best` field. These modified results are added to the new dataset as new utterances. In our experiment, we performed 5 replacements for each utterance, thereby increased the number of training samples from 5119 to 30714.

Results Analysis. Performance of data augmentation are shown in the second part of Tab 1, which demonstrates a significant improvement when replacing `POI`, and little improvement when replacing all slot. The two key reasons for the improvement when only replacing `location`-related slot, are: (a) Expanding the dataset enhances the model's generalization capability. (b) Replacing `location` values typically **does not alter the semantics of the sentence (i.e. it does not produce incorrect semantics)**.

Furthermore, we attempted to replace all `slot-value` pairs (`request type`, `route preference` etc.), but as shown in part-`AugAll` in Tab 1, this approach did not yield favorable outcomes. One possible reason is that, such replacement may cause semantically incorrect samples, as shown in Fig 3. Another reason could be that, our training is based on an in-car voice system, where the data primarily revolves around "finding a location". Therefore replacing `location`-related values is likely to generate a data distribution more consistent with original dataset.

3.2. Noisy Data Filtering

Motivation. We identify that some data in the training set is useless or serves as distracting content. For example, Fig 4 displays an example of an utterance considered as noisy data. Therefore, we attempt to clean the data to eliminate the impact of low-quality data, as well as to remove erroneous and missing values. Data cleaning involves removing noise and errors from the dataset, improving its quality and reliability, and thus providing a more accurate and trustworthy foundation for subsequent model training.

For example, we believe those marked as *(unknown)* parts in the `manual_transcript` are improperly recognized instances during the annotation process. Such data appears to offer no benefit to the training process and may even introduce noise. As a result, we primarily remove these samples and retained only the data that we consider to have a higher level of reliability.

Results Analysis. Result is shown in the third part of Tab 1. However, the performance after data filtering does not improve, or even shows a decline in some metrics. This could be attributed to the following reasons:

Model	Opereation	Accuracy	Precision	Recall	F1
LSTM	–	71.06	81.40	73.93	77.49
RNN	–	71.28	76.32	73.93	75.11
GRU	–	71.84	79.93	75.60	77.71
Augmenting Data by Replacing Slot					
LSTM	+ AugPoi	**78.77**	**83.14**	**81.23**	**82.17**
RNN	+ AugPoi	**77.88**	**80.46**	**81.13**	**80.79**
GRU	+ AugPoi	**78.32**	**81.94**	**81.86**	**81.90**
LSTM	+ AugAll	70.50	81.70	73.10	77.16
RNN	+ AugAll	70.50	76.69	73.41	75.01
GRU	+ AugAll	71.17	80.71	73.72	77.06
Handling Noise Data					
LSTM	+ Filter	70.28	78.00	73.93	75.91
RNN	+ Filter	69.72	73.81	72.89	73.35
GRU	+ Filter	70.06	81.50	73.51	77.30
Considering Dialogue Context					
LSTM	+ History	69.39	80.28	70.91	75.30
RNN	+ History	68.04	73.93	70.07	71.95
GRU	+ History	70.50	77.92	71.01	74.30
Fusing Word Chunk Embedding					
LSTM	+ Fused	**72.51**	**82.25**	**74.87**	**78.38**
RNN	+ Fused	**71.51**	**80.23**	**74.04**	**77.01**
GRU	+ Fused	**72.74**	**79.93**	**75.18**	**77.49**

Table 1. Preliminary study on several data processing methods and new pipeline involving word-chunk information. (First part) Baseline. (Second part) Including two data augmentation methods: `AugPoi` only augments the `POI` value, and `AugAll` augments all slot categories. (Third part) Handling noise data by filtering. (Forth part) Concatenating dialogue histories in order to leverage context information. (Fifth part) Fusing additional word-chunk embedding into original embedding. The word-Chunk is generated by `Jieba`. Each part's performance is verified on traditional pipeline varying 3 different model.

1. The data cleaning in this study primarily relies on `(unknown)` as the criterion, without further analyzing the contextual information or other features of these samples. This could result in some helpful samples being misclassified as noise and removed.

2. Although the cleaned data contain noise or errors, such noise might, in certain scenarios, provide the model with opportunities to train for robustness. Completely removing this noise could lead to the model lacking adaptability to similar noisy situations in real-world scenarios.

3. While noise data may superficially affect the accuracy of model training, it can as well contribute positively to the model's robustness. In real-world applications, inputs are rarely perfectly normalized or noise-free. For example, accents in speech recognition, environmental noise, or spelling errors in text. If noise data is entirely removed during model training, the model

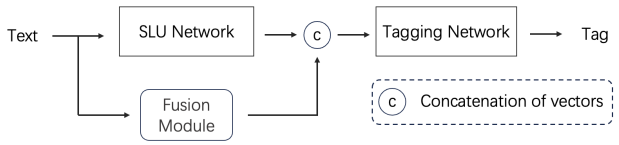

Figure 5. New pipeline in traditional framework. Word-chunk is obtained by `Jieba`, and is then transferred to embedding by sentence model. Finally the word-embed and chunk-word-embed is concatenated.

may perform well only under ideal data distributions but lack the ability to handle noisy inputs in practical scenarios.

3.3. Adding History

Motivation. In spoken language understanding (SLU) tasks, leveraging dialogue history is essential because user utterances are often concise and context-dependent. Certain current utterances can only manifest semantics in a context. For example, consider the multi-turn user utterances: `["I want to book a flight","Departing from Shanghai","Tomorrow morning","To New York"]`. In this dialogue, the full meaning of `To New York` cannot be understood without the context of the previous dialogue. By incorporating dialogue history, the user's complete intent—`to book a flight departing from Shanghai tomorrow morning to New York`—can be accurately inferred. Dialogue history provides rich contextual information, enabling the model to capture implicit semantic relationships and accurately parse user intents and key slot information. Such a comprehensive understanding of context may improve model accuracy and enhance its adaptability to complex dialogue scenarios, ultimately optimizing the user experience.

Methodology. When the `utt_id` of the input utterance is n, we concatenate the `asr_1best` of the current sentence with those of the previous $n-1$ turns, separating them with `<pad>` tokens, while keeping the `semantic` unchanged. In the resulting `pred` we retain only the items whose slot values exist in the `asr_1best`. This approach enables the use of dialogue history to perform semantic triple parsing for the current sentence.

Results Analysis. Results are shown in the forth part in Tab 1, which indicate that incorporating dialogue history does not lead to performance improvement. This is likely due to 90% of the samples in the given training set consist of only a single utterance, with only a small portion containing dialogue history. Moreover, based on the collected data, many of the samples with context were miscollected (e.g. the system incorrectly assumed the passenger intended to have a dialogue). As a result, many consecutive dialogues consist of distracting information, causing the concatenation to introduce more noise than benefits, ultimately leading to suboptimal results.

3.4. Fusing Word Chunk information

Motivation. Jieba is an excellent Chinese word segmentation tool that efficiently provides segmentation information. In SLU tagging tasks, each character needs to be mapped to one of 74 labels to determine its `act` type, `slot` type, and `BIO` type. The introduction of segmentation information can provide the model with additional contextual and structured semantic information. Specifically, in `BIO` type classification, segmentation information helps the model more accurately capture word boundaries and semantic features. Based on this, we attempt to incorporate Jieba's segmentation information into the SLU tagging task, aiming to enrich input features, optimize the model's classification capability for `BIO` types, and ultimately improve overall performance. This approach theoretically enhances the model's understanding of sentence structures, supporting more complex semantic parsing tasks.

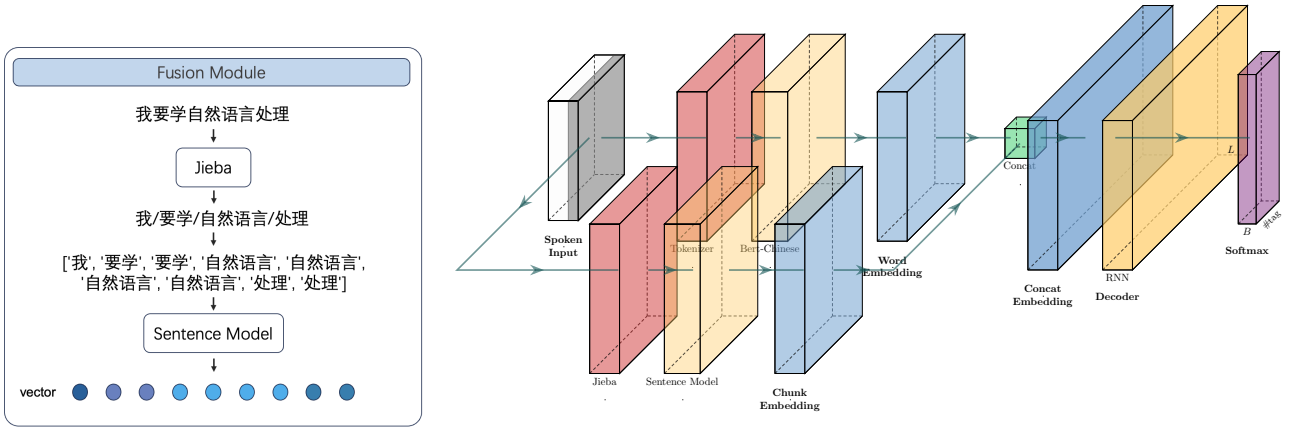

Figure 6. Integrated procedure of BERT-based model pipeline. (Left) First obtain word-chunk by `Jieba`, then compute word-chunk embedding by sentence model, whose input unit is chunk, differs from BERT. (Right) Concatenating chunk-level embedding and word-level embedding to be the final embedding. Then pass through decoder and softmax layer to output.

Pipeline. We first use `Jieba` to segment the input sentence and repeat each segmented token n times, where n is the length of each token. Next, we use a SentenceModel to encode each segmented token, generating its corresponding vector representation. If the input shape is (B, L), the resulting vector representation will have a shape of (B, L, D), where B is the batch size, L is the length of the longest sentence in the batch (with shorter sentences padded with empty strings), and D is the length of the encoded vectors. An example is exhibited in the left part of Fig 6.

As shown in the Fig 5, the vectors generated by Jieba segmentation are concatenated with the vectors produced by the SLU Network (which are the vecotr outputs of nn.Embedding in the baseline). These concatenated vectors are then passed through the Tagging Network, which assigns a label to each character in the sentence.

Results Analysis. Results are shown in the fifth part of Tab 1, which demonstrate that incorporating Jieba's segmentation information into the SLU tagging task leads to performance improvements across all metrics. For example, with the LSTM model, the accuracy increases from 71.06 to 72.51, and the F1 score improves from 77.49 to 78.38. This indicates that the introduction of segmentation information effectively enhances the model's ability to capture `BIO` type boundaries and semantic features. Similar improvements are observed with RNN and GRU models, suggesting that segmentation information has broad applicability across different model architectures. These results validate that segmentation information enriches input features, enabling the model to understand sentence structures and contextual relationships better, thereby improving semantic parsing capabilities and providing higher-quality outputs for subsequent tasks.

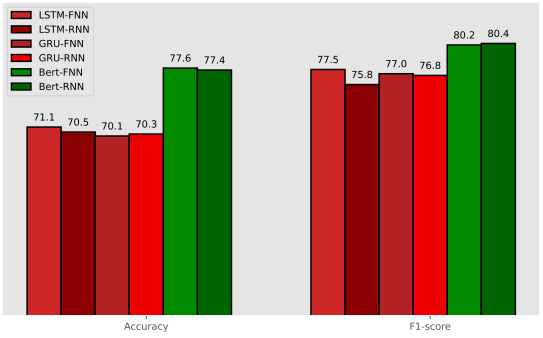

Figure 7. Comparison of choice of encoder and decoder. BERT significantly outperforms other traditional recurrent network. Decoder type is not important, FNN is sufficient to understand the feature.

3.5. BERT as Encoder and RNN as Decoder

Motivation and Method. As discussed in the related work, BERT, as a transformer-based pre-trained model, has demonstrated excellent performance in handling semantic tasks. We experiment by introducing BERT as the encoder, replacing the traditional RNN in the deep learning pipeline for computing word embeddings. After BERT, we continue to use FNN as the decoder, and finally, a softmax layer is applied to obtain the probability distribution of all possible labels for each word.

Furthermore, we reconsider the role of the decoder layer in processing the embeddings. If we replace the FNN with an RNN, it might be able to learn the contextual semantic relationships, leading to better tag predictions.

Results Analysis. To investigate this, we applied two different decoder architectures to Bert-Chinese and retrained the models without incorporating any additional data augmentation methods. The results are shown in Fig 3.4, which demonstrate two insights: (a) Using BERT as encoder extremely outperforms traditional recurrent network. (b) The decoder cell type has little contribution to the performance, FNN is sufficient to achieve satisfactory performance.

3.6. Fine-tuning BERT

Motivation. Since BERT is a large model, fine-tuning it requires substantial computational resources. Moreover, the corpus used for fine-tuning in this paper is much smaller than the pre-trained corpus, which may cause the model to learn noise or specific patterns from our data, potentially leading to overfitting. Based on these two points, we conducted a preliminary study on fine-tuning BERT with varying degrees of parameter locked.

Results. The results are shown in Fig 3.6, which indicate that fine-tuning from 80%, to even 100% BERT parameters on our small corpus is effective, and that's what we adopted.

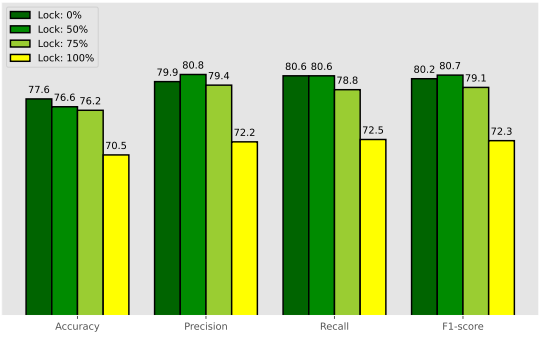

Figure 8. Fine-tuning BERT varying different locked parameters radio.

3.7. Revisit Data Augmentation

Motivation. Having thoroughly investigated the power of BERT and the correct fine-tuning methods, naturally, we revisit the data augmentation techniques we previously explored, applying them to the new BERT-based pipeline.

Results Analysis. We conducted the data augmentation on 3 different BERT model (BERT-Chinese, MacBERT, RoBERTa) as mentioned in 2.3. Results are shown in Fig 3.7, which indicates that this technique is still highly effective, leading to an average $\uparrow$ 1.9% in accuracy and $\uparrow$ 1.8% in F1-score.

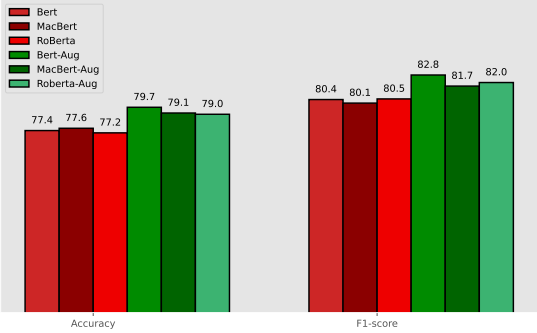

Figure 9. Performance of data augmentation on several BERT-based model. Replacing `location` slot leads to improvement in every BERT model.

3.8. Leveraging Word-Chunk Information in BERT

Motivation. We previously proposed how to fuse word-chunk embedding information within the traditional DL pipeline, as shown in Fig. 5, achieving performance improvements. In this section, we extent this pipeline to the scenario where BERT is used as the encoder and RNN as the decoder.

Method. Similarly, as shown in Fig. 6, based on the original pipeline, the spoken input sentence is processed through both the BERT tokenizer and `Jieba`, obtaining single-word level segmentation and chunk-word level segmentation, respectively. The single words are passed through the BERT model, with the final hidden layer being taken as the word embeddings. Since BERT's input requires individual Chinese characters, chunk-level words are processed through a new Sentence model to obtain chunk-level embeddings. Finally, the two embeddings are concatenated and passed through the decoder RNN and softmax layer to obtain the tagging prediction distribution for each Chinese character position.

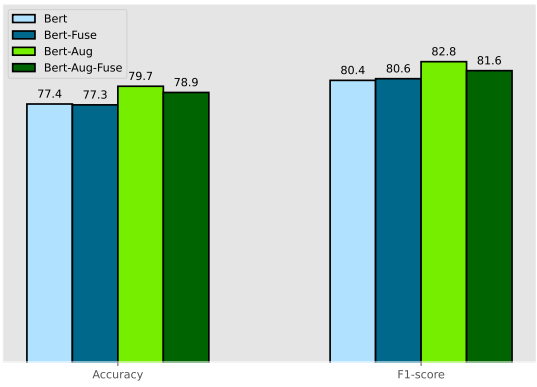

Figure 10. Performance of fusing word-chunk embedding as our proposed pipeline. It's worth noting that slight improvement in F1-score is observed when data augmentation is not applied. But the performance comes to a slight decline when adding data augmentation (but still better than without data augmentation).

Results and Analysis. The results are shown in Fig 3.8. To our surprise, the improvement in performance is marginal or even negative. The result shows that, when data augmentation is not applied, fusing word-chunk embedding leads to a slight improvement in F1. But when data augmentation is added, although the final accuracy and F1 is increasing, but it's worse than without fusing additional information.

In the framework of traditional DL pipeline without BERT, fusing additional word-chunk embedding can lead to a higher performance. We conclude this as: RNN have limited capacity to learn long semantic relationships from context, and the additional word-chunk semantics help RNN better capture the contextual meaning. However, when encoder is set to BERT, which can efficiently capture long-range semantics by self-attention, therefore the impact of integrating word-chunk semantics is limited.

4. Experiments

After the exploration in the previous section, we have investigated various methods to improve the SSF task. In summary, the three most effective methods are: (a) Using the pre-trained BERT model as the encoder. (b) Replacing the `location` slot to augment the training data. (c) Fusing word-chunk embedding information. In this section, we gradually integrate these methods, ultimately forming our final proposed model pipeline.

Dataset. 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. We report the results on the development set after training the models on the training set.

Parameters. All the models are trained for 100 epochs with a batch size of 32, which is proved enough for convergence. We use Adam optimizer with a learning rate of 1e-3 for RNN-based models and 1×10^{-4} for Bert-based models.

The experiments are conducted on a machine with NVIDIA GeForce RTX3090 GPU with 24 GB memory. The machine is equipped with an Intel(R) Core(TM) i9-10920X CPU, with a base clock frequency of 3.50GHz. This CPU features 12 cores and 24 threads.

Results. The results are shown in Fig 4. The grey bars represent two traditional DL baseline with recurrent network to generate embedding. The blue bar represents our

first method that engaging pre-trainging model as encoder to generate better word embedding. which leads to a huge improvement. Then the two green bar represent more modification on the BERT-based model basis. The darkgreen bar represents augmenting data by replacing `location` with random selected place-name in corpus. **This method is the most effective improvement we've made. It outperforms the baseline by 9.16% and the pure BERT model by 2.23% in terms of accuracy on the test set**. The lightgreen bar represents further fusing of additional word-chunk embedding information, which is based on a new pipeline we proposed in Fig 6. However, this final fused model did not perform as well as we had anticipated. It slightly underperformed compared to the results achieved using only data augmentation. Nevertheless, it still outperforms the baseline by $\mathbf{8.38}\%$ and the pure BERT model by $\mathbf{1.45}\%$ in terms of accuracy on the test set.

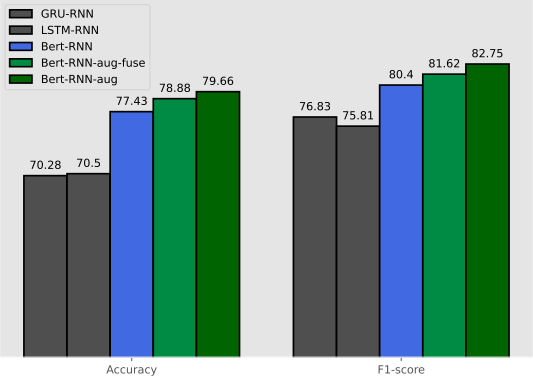

Figure 11. Results of our final proposed model and pipeline, which integrates several methods that studied in Exploration section. The GRU and LSTM turns to be baseline.

5. Conclusion and Future Work

This paper addresses the SSF problem in SLU from data and model perspectives. In the `Exploration` section, we investigated the effectiveness of various methods, including data augmentation, data cleaning, leveraging dialogue context, using pre-trained models for better word embedding, and employing `Jieba` and sentence models to obtain chunk-level embedding. In the final `Experiment` section, we gradually combined the three most effective methods, achieving improved accuracy and F1-score.

There are still several limitations and directions for future work. For instance, (a) Data cleaning process could be more thorough, with offline corrections applied to ASR errors. (b) The fusion of word-chunk embeddings did not yield the expected results, likely due to improper network structure design, which leaves room for further improvements.

Acknowledgments. We express our gratitude to advisor and teaching assistants for their guidance throughout the course, as well as their time to read and evaluate our paper.

References

Chen, T., Xu, R., He, Y., and Wang, X. Improving sentiment analysis via sentence type classification using bilstm-crf and cnn. *Expert Systems with Applications*, 72:221–230, 2017. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2016.10.065. URL https://www.sciencedirect.com/science/article/pii/S0957417416305929.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches, 2014. URL https://arxiv.org/abs/1409.1259.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014. URL https://arxiv.org/abs/1412.3555.

Cui, Y., Che, W., Liu, T., Qin, B., Wang, S., and Hu, G. Revisiting pre-trained models for Chinese natural language processing. In Cohn, T., He, Y., and Liu, Y. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 657–668, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.58. URL https://aclanthology.org/2020.findings-emnlp.58.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Elman, J. L. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. doi: https://doi.org/10.1207/s15516709cog1402_1. URL https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1402_1.

Graves, A. and Schmidhuber, J. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pp. 2047–2052 vol. 4, 2005. doi: 10.1109/IJCNN.2005.1556215.

Guo, D., Tur, G., Yih, W.-t., and Zweig, G. Joint semantic utterance classification and slot filling with recursive neural networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pp. 554–559, 2014. doi: 10.1109/SLT.2014.7078634.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

Kim, K., Jha, R., Williams, K., Marin, A., and Zitouni, I. 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)*, pp. 757–767, 2019.

Lafferty, J. D., McCallum, A., and Pereira, F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, 2001. URL https://api.semanticscholar.org/CorpusID:219683473.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

McCallum, A., Freitag, D., and Pereira, F. C. Maximum entropy markov models for information extraction and segmentation. In *International Conference on Machine Learning*, 2000. URL https://api.semanticscholar.org/CorpusID:775373.

Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., He, X., Heck, L., Tur, G., Yu, D., and Zweig, G. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3): 530–539, 2015. doi: 10.1109/TASLP.2014.2383614.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26, 10 2013. URL https://proceedings.neurips.cc/paperfiles/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

Pennington, J., Socher, R., and Manning, C. GloVe: Global vectors for word representation. In Moschitti, A., Pang, B., and Daelemans, W. (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

Radford, A. and Narasimhan, K. Improving language understanding by generative pre-training. 2018. URL https://api.semanticscholar.org/CorpusID:49313245.

Sun, Z., Li, X., Sun, X., Meng, Y., Ao, X., He, Q., Wu, F., and Li, J. ChineseBERT: Chinese pretraining enhanced by glyph and Pinyin information. In Zong, C., Xia, F., Li, W., and Navigli, R. (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 2065–2075, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.161. URL https://aclanthology.org/2021.acl-long.161.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wei, J. and Zou, K. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 6382–6388, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1670. URL https://aclanthology.org/D19-1670.

Xu, P. and Sarikaya, R. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 78–83, 2013. doi: 10.1109/ASRU.2013.6707709.

Yao, K., Peng, B., Zhang, Y., Yu, D., Zweig, G., and Shi, Y. Spoken language understanding using long short-term memory neural networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pp. 189–194, 2014. doi: 10.1109/SLT.2014.7078572.

A. Experiment Results

In the main body of the paper, we present the majority of experimental results, including a comparison of before and after data processing, on both the baseline and BERT models. To maintain control over the variables and highlight the point, each chart presented in the main body selects only a specific set of data on accuracy and F1. The complete experimental data is provided in appendix. The code is available at https://github.com/CalciumArgon/CS3602_Natural_Language_Processing

Encoder	Decoder	Data-Augmentation	Fuse Word-chunk	Accuracy	Precision	Recall	F1-score
LSTM	FNN	✗	✗	71.06	81.40	73.93	77.49
LSTM	RNN	✗	✗	70.50	78.61	73.20	75.81
GRU	FNN	✗	✗	70.06	81.01	73.41	77.02
GRU	RNN	✗	✗	70.28	80.71	73.31	76.83
GRU	RNN	✓	✗	78.32	82.25	80.71	81.47
LSTM	FNN	✗	✓	72.51	82.25	74.87	78.38
MacBert	GRU	✗	✗	77.65	79.73	80.40	80.06
MacBert	RNN	✗	✗	76.09	79.79	79.87	79.83
RoBerta	RNN	✗	✗	77.21	80.84	80.08	80.46
Bert	FNN	✗	✗	77.65	79.86	80.60	80.23
Bert	RNN	✗	✗	77.43	80.82	79.98	80.40
MacBert	GRU	✓	✗	79.11	81.95	81.44	81.69
MacBert	RNN	✓	✗	78.88	82.77	81.65	82.20
RoBerta	RNN	✓	✗	78.99	82.42	81.65	82.03
Bert	RNN	✗	✓	77.32	81.25	79.98	80.61
Bert	RNN	✓	✗	**79.66**	**83.46**	**82.06**	**82.75**
Bert	RNN	✓	✓	78.88	82.22	81.02	81.62