Light Weight Bert-Based Pipeline for Chinese Slot Understanding

Benhao Huang, Yizhou Liu, Pengxiang Zhu[†]

***Abstract*—Slot understanding is a crucial part in the field of spoken language understanding. Prior works have employed pretrained models such as BERT and levereged lexicon information. In this paper, we follow the general pipeline of Part-Of-Speech (POS) Tagging and propose a light weight BERT-based module with a post-processing section that incorporates word-level features with hidden features from a pretrained language model. Compared with existing methods, our model preserves the structure of BERT and features model-level fusion. Experiments on the given dataset which has limited size and contains noisy data show that our model achieves decent results.**

***Index Terms*—Slot Understanding, BERT, Lexicon Interpretation**

I. Introduction

Language Understanding, a fundamental aspect of artificial intelligence, revolves around the comprehension and interpretation of human language by machines. In the dynamic landscape of technology, the quest to endow machines with the ability to grasp the subtleties of language has led to the emergence of Language Understanding as a pivotal discipline. From parsing syntactic structures to capturing semantic nuances, Language Understanding seeks to bridge the gap between human communication and machine intelligence, paving the way for advancements in natural language interfaces, information retrieval, and dialogue systems. As we explore the landscape of Language Understanding, we unravel its significance in augmenting the capabilities of artificial intelligence and facilitating seamless interactions between humans and machines. Regarding with specific tasks in language understanding, our research implements slot understanding by formulating (a, s, v) tuples which relates value with given act set and slot set.

II. Related Works

A. Pre-trained Language Model

Language model pre-training involves training a model on a large dataset for general language understanding. The model can later be finetuned for given tasks by exposing it to task-specific data.

Certain NLP researches have provided warm startups for pre-trained language models. Methods leveraging word embeddings, such as Word2Vec [1] and GloVe [2] act as common initializers for word vectors of deep learning models, effectively improving the performance on downstream tasks. To supplement the representation of diverse meanings of words in context, context-aware language models were proposed to generate complete context information, which facilitated significant performance improvement in sentiment analysis, text classification and object classification [3]. Nonetheless, such algorithms require apropos pre-training and only perform outstandingly in a fraction of NLP tasks.

Since then, *pretraining and finetuning* paradigm emerges. Generative pre-training demonstrated the potential of pre-training methods for diverse downstream tasks by groundbreakingly utilizing unidimensional transformers as the backbone [4]. BERT [5] leveraged bidimensional transformers and introduced a denoising autoencoding pre-training task, achieving another leap in natural language understanding. Mainstream pre-trained language models adopt self-supervised pre-training tactic, where labels are derived from unannotated data. Pre-trained language models have been proved to boost the performance in diverse natural language processing tasks [6], such as natural language inference, named entity recognition, and question answering [7], [8].

B. Chinese Sequence Labelling

Sequence labeling has proven to be effective in various natural language processing (NLP) tasks that depend on contextual information, including named entity recognition, part-of-speech tagging, and shallow parsing [9]. In this framework, the model takes a sequence of tokens $(w_1, ..., w_T)$ as input and generates a label for each token as output $(x_1, ..., x_T)$. The utilization of word embeddings enables sequence models to learn similar representations for words that are semantically or functionally alike. Recent advancements in sequential model frameworks have enhanced the model's capacity to deduce representations for words not encountered before and to share information regarding morpheme-level regularities [10]. Sequence labeling models derive advantages from the incorporation of long short-term memory (LSTM) units [11] as these units can capture long-term contextual dependencies in natural language. A modified version of the traditional architecture, bi-directional LSTM (BiLSTM) [12], has proven highly successful in language tasks. BiLSTM considers both left and right contexts of a word, thereby increasing the amount of pertinent information available to the network. Additionally, employing secondary learning objectives can enhance the number of salient features and access to relevant information. For instance, it is demonstrated that training a model to predict surrounding words concurrently incentivizes the discovery of useful features and associations unlikely to be revealed otherwise [13].

[†] All three authors are senior students from Shanghai Jiao Tong University and are of equal contribution.

Comparatively, Chinese sequence labelling is more difficult since it usually involves word segmentation. Word segmentation before sequence labelling is adopted, and methods are proposed to alleviate word boundary issues [14], [15]. To avoid negative impacts of incorrect word segmentations, Chinese sequence labelling is conducted on character-level [16], [17], [18]. Moreover, external lexicon can provide rich word boundary information and word semantic information, which have been proven effective in Chinese NER [19]. For instance, lexicon words are incorporated into character representations with BERT via SoftLexicon [20]; Zhao et al. [21] develop a lexicon-enhanced adaptive attention mechanism, leveraging external lexicons and achieving outstanding performance on rare or ambiguous words.

C. *Slot Filling in Language Understanding*

Slot filling is a crucial aspect of language understanding that involves identifying and extracting specific pieces of information, referred to as *slots*, provided text or spoken input. It aims to discern and categorize key details within a larger context, facilitating a more nuanced comprehension of user intent or content. A richer understanding of the overall semantic can be achieved via effectively capturing essential elements in designated slots. Hence, this nuanced approach enhances the accuracy and relevance of responses in natural language processing applications, making interactions more seamless and contextually aware.

In slot filling tasks, the input is a sentence L, while the output can be the sequence of slot ids S, corresponding to each word in the sentence. As slots are usually tailored into specific domains, artificial intelligence-based methods may face challenges due to errors in automatic speech recognition and limitations in modeling the diverse ways people express the same concept in natural language. Therefore statistical approaches, including generative methods, discriminative classification methods and knowledge-based methods, are employed [22], [23]. These methods aim to find the slot sequence with maximal *a posteriori* probability via the Bayes rule:

$$\hat{S} = \arg\max_S P(S|L) = \arg\max_S P(L|S)P(S)$$

The initial generative model posits an injection between model states and segments: each state corresponds to a single segment, and the arrangement of the segments mirrors the order of the states [24], [25]. Meanwhile, the Hidden Vector State model takes a different approach, where the representation of states encapsulates all structural information of the semantic tree. This modification transforms prior models into a Markov chain within a Hidden Markov Model (HMM) framework [26].

In parallel, discriminative models, such as conditional random field [27] (CRF), which models the conditional probability of slot sequences given the word sequence by extracting features from current and previous states, are leveraged and quickly outperform conventional generative models. Similar methods emerge: for instance, SVM-based semantic classifier treats classification as a black-box, requiring decent feature engineering [28]. Generally, a promising prospect is to integrate feature design and classification into deep learning for improved efficiency and adaptability [29], [30].

III. Method

A. *Problem Formulation and General Framework*

Given an input of ASR translation results of spoken language (denoted by concatenation of static characters $[c_1, \ldots, c_n]$), we need to formulate an output tuple (a, s, r), where a denotes the act of the sentence, s denotes the slot of the sentence and r denotes the value of the semantic label. The acts and slots of a given dataset falls in a fixed set of words, and we denote the action set as A and the slot set as S.

We follow the typical POS-tagging method to approach the problem. For each character c_i in the given sentence, we classify it into $2|A||S| + 2$ labels, including `B-(a,s)` for the beginning character of a value affiliated to act a and slot s, `I-(a,s)` for intermediate values, `O` for a character not belonging to any value and `<pad>` for padding indicator.

The framework of our proposed pipeline is shown in Figure 1. Given an input Chinese sentence, we first feed the sentence through a BERT backbone and obtain the final hidden state. To better leverage lexicon information, we first pad the sentences in batches and apply `jieba` cut to obtain the possible word segmentation of the sentence. It is then passed through a pretrained text embedding layer and further fed into a Lexicon adaptation module that extracts local features. By merging the two parts together, we can derive a feature that contains both character-level and word-level information. We will discuss the implementation details of the optimized lexicon adapter in the next section.

B. *Optimized Lexicon Adapter for Local Features*

Following [31], we apply lexicon Adapter to further extract word-level information that can help resolve the ambiguities of the final classification. As shown in Figure 1, the lexicon Adapter takes in two inputs, *i.e.* hidden features and word embeddings. Note that the segmentation results are already given for [31], which is not the case for this paper, and we will list our procedures as follows. Given a raw sentence $[c_1, \ldots, c_n]$ where c_i denotes a Chinese character, we first apply segmentation using `jieba` package with `cutall=True`. This yields a list of words $[w_1, \ldots, w_m]$ where $w_i = [c_j, c_{j+1}, \ldots, c_k]$ and $j, k \in \{1, \ldots, n\}$. By iterating over each word w_i, we pair the word w_i with each $c_j \in w_i$. Suppose W is the maximum number of words that a character is paired with, we pair all the other words with empty string as padding. For a batch of B sentences, we also pad the sentences to length L, eventually yielding a list of size $B \times L \times W$ for training.

In [31], the Lexicon Adapter module is embedded in the first and second hidden layer of the BERT backbone. In our proposed pipeline, to maintain the integrity of the BERT backbone and offer more flexibility, we move the Lexicon Adapter to the final hidden layer of the BERT backbone. To better leverage the information of the Lexicon Adapter, we add a single-directional LSTM layer to process the extracted

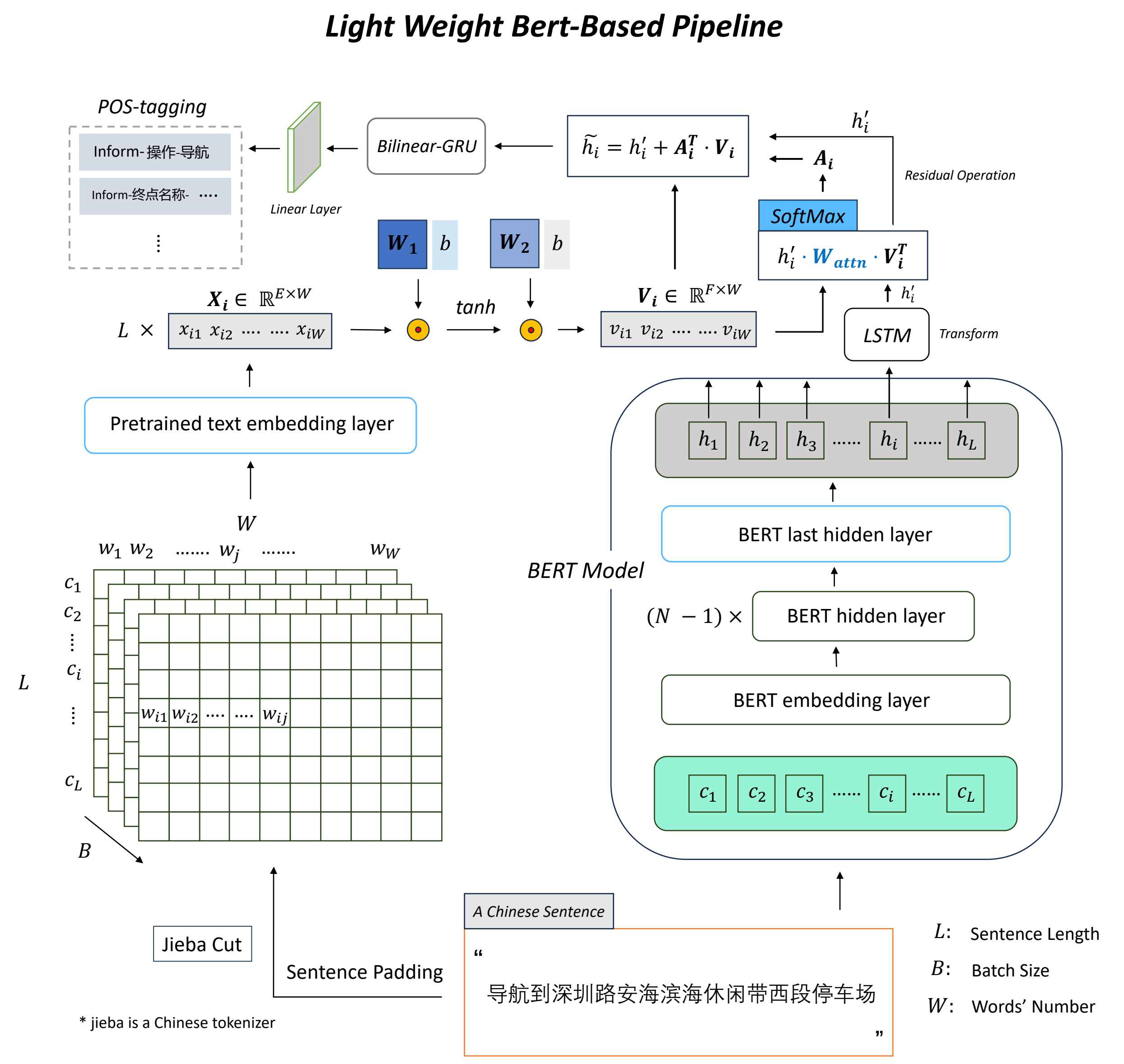
Light Weight Bert-Based Pipeline
POS-tagging
Inform-操作-导航
Inform-终点名称- ····
Linear Layer
Bilinear-GRU
$\tilde{h}_i = h_i' + A_i^T \cdot V_i$
h_i'
A_i
Residual Operation
SoftMax
$h_i' \cdot W_{attn} \cdot V_i^T$
W_1 b
W_2 b
tanh
$X_i \in \mathbb{R}^{E\times W}$
$L \times$ $x_{i1}\ x_{i2}$ ···· ···· x_{iW}
$V_i \in \mathbb{R}^{F\times W}$
$v_{i1}\ v_{i2}$ ···· ···· v_{iW}
LSTM
Transform
Pretrained text embedding layer
h_1 h_2 h_3 h_i h_L
BERT last hidden layer
BERT Model
$(N-1)\times$ BERT hidden layer
BERT embedding layer
c_1 c_2 c_3 c_i c_L
W
$w_1\ w_2$ ······· w_j ······· w_W
c_1 c_2 c_i c_L
L
$w_{i1}\ w_{i2}$ ···· ···· w_{ij}
B
Jieba Cut
Sentence Padding
* jieba is a Chinese tokenizer
A Chinese Sentence
“ 导航到深圳路安海滨海休闲带西段停车场 ”
L: Sentence Length
B: Batch Size
W: Words' Number

Fig. 1. General framework of our proposed pipeline. Given an input Chinese sentence, we first feed the sentence through a BERT backbone and obtain the final hidden state. To better leverage lexicon information, we first pad the sentences in batches and apply jieba cut to obtain the possible word segmentation of the sentence. It is then passed through a pretrained text embedding layer and further fed into a Lexicon adaptation module that extracts local features. By merging the processed hidden states and the lexicon information, we can derive a feature that contains both character-level and word-level information.

hidden features first. Following [31], we merge word level information with characters.

Given a sentence $\mathbf{s} = [c_1, \dots, c_n]$, suppose the character c_i has transformed hidden features h_i and paired words $[w_{i1}, \dots, w_{ij}]$. We first employ a pretrained text embedding layer [32] to transform w_{ij} to feature vectors $x_{ij} \in \mathbb{R}^E$ (E denotes the text embedding dimension). We then use two linear mappings combined with $\tanh$ activation function to align two representations, where $\mathbf{W}_1 \in \mathbb{R}^{F\times E}$ and $\mathbf{W}_2 \in \mathbb{R}^{F\times F}$, F denotes the feature.

$$v_{ij} = \mathbf{W}_2(\tanh(\mathbf{W}_1 \cdot x_{ij} + \mathbf{b}_1)) + \mathbf{b_2} \tag{1}$$

According to [31], character-to-word attention is introduced to pick out the most relevant words from all matched words. Suppose $V_i = [v_{i1}, \dots, v_{ij}]$, the *weight* for each paired word can be formulated as follows, where $\mathbf{W}_{\text{attn}}$ is the learnable weight matrix from bilinear attention.

$$A_i = \text{softmax}(h_i \mathbf{W}_{\text{attn}} V_i^T) \tag{2}$$

Hence, the weighted sum of all the paired words is then derived. We apply residual connection to the transformed word embeddings with the corresponding hidden features.

$$z_i = \sum_{i=1}^{W} A_{ij} v_{ij} \tag{3}$$

$$\tilde{h}_i = h_i + z_i \tag{4}$$

Inspired by [33], we apply a single bilinear GRU layer followed by a linear layer for POS-tagging classification as the final decoder of our proposed pipeline.

IV. Experiments

A. Data Augmentation for Noisy ASR Inputs

For a typical SLU task, the values are naturally the subset of the original sentences. It is worth noting that in this approach, if the ASR inputs is faulty, it is possible that the values are never present in the ASR inputs upon which the values are extracted using the POS-tagging pipeline. For these data, it is theoretically impossible to predict the correct value. Amongest the given `train` and `dev` datasets, 11.2% and 12.3% of which fit this condition. We deem those data as *invalid* for our method.

In such case, we employ several techniques to facilitate training. We first remove all the invalid data in `train` dataset, as no training methods on these data will extract correct values. In addition, we manually create data with `asr` inputs replaced with manual scripts. By allowing the model to see the *ground truth* sentence along with the *transcribed* sentence, we attempt to improve the model's adaption ability while still preserving the correct values. Nevertheless, **no modifications are made to dataset** `dev`, and all the results reported in the following section have considered all the invalid cases.

During training, we also encountered cases of homophones, *i.e.* characters that are similar in pronunciation but different in the actual formation. Intuitively, we can employ a pretrained Chinese corrector model `pycorrector` [34] to first correct the mis-interpreted characters as a part of data preprocessing. However, empirically we have not observed a significant improvement in the final result (and the pretrained `pycorrector` model is relatively large), so we list it as a possible option that could work for more general datasets of a greater size.

B. Empirical Results

We conduct extensive experiment using the datasets given, namely `train` and `dev`. Note that `test` is unlabelled, so all the metrics are reported on tests on dataset `dev`, and the predicted result for `test` is also provided in the files submitted. We employ Adam optimizer and train for 100 epoches for all the experiments. For the BiLSTM baseline, we have employed the default hyper-parameters with learning rate 1×10^{-3}. For other BERT-based models, we set the learning rate to be 1×10^{-4} initially and decay the learning rate by $\gamma = 0.1$ after 50 epoches. We also set the weight decay to be 1×10^{-3}, dropout probability to be $p = 0.2$ and the fix rate to be 0.

The reason for setting the fix rate to 0 is illustrated in Figure 2, where we observe a consistent improvement in the model's performance as the fix rate decreases. Opting for a fix rate of 0 yields marginally superior performance compared to a fix rate of 0.2. However, the negligible difference between these two settings suggests that experimenting with a fix rate of 0.2 could be beneficial. This approach potentially offers a reduction in parameter size and a faster training process. Given the constraints of time, we plan to explore this possibility in our future research.

Additionally, we have conducted experiments to evaluate the impact of different configurations of the input decoder of

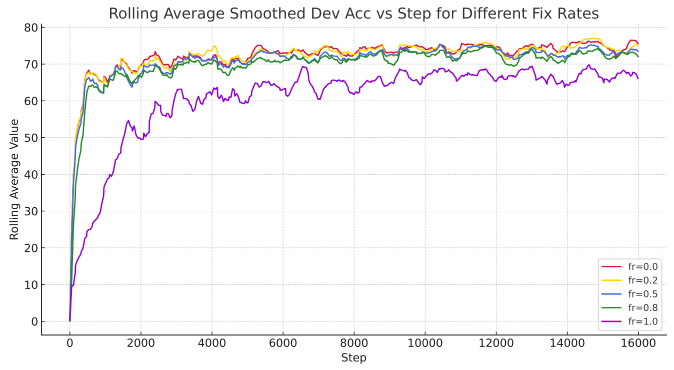

Fig. 2. Rolling Average Smoothed Accuracy on Development Set vs Step for Different Fix Rates.

the Lexicon Adapter (referred to afterwards as LA-decoder afterwards) and final decoder prior to the last linear layer, with the outcomes presented in Table I. These results indicate that varying the LA-decoder and decoder settings leads to subtle differences in the model's performance across several metrics. To maximize accuracy, we chose to configure the LA-decoder as an LSTM and the final layer decoder as a GRU, based on these findings.

TABLE I
Decoder Choice Experiment

LA decoder	decoder	Accuracy	Precision	Recall	F1
GRU	GRU	79.22	82.07	82.59	**82.33**
RNN	GRU	79.33	81.57	82.17	81.87
LSTM	GRU	**79.44**	81.55	82.06	81.81
GRU	LSTM	79.33	81.25	82.69	81.96
GRU	RNN	79.22	81.31	**83.00**	82.15
GRU	FNN	79.22	**82.10**	82.27	82.19

The empirical results for the baselines and our proposed model is shown in Table II. We can observe performance boost after employing the optimized Lexicon Adapter, attaining an accuracy of 79.44%. The curves for key metrics of the training is shown in Figure 3. From the figures, we can see that the model quickly converges to a relatively high accuracy, and subsequent training gradually refines the best accuracy. We deem that the fluctuation is due to the noise in dataset `dev` as well as the relatively small size of the dataset.

TABLE II
Empirical Results

Method	Accuracy	Precision	Recall	F1
BiLSTM-Baseline	71.28	81.61	73.62	77.41
BERT-Baseline	77.99	81.49	81.23	81.36
RoBERT-Baseline	77.99	82.19	81.33	81.76
MacBERT-Baseline	78.32	**82.25**	80.71	81.47
Ours	**79.44**	81.55	**82.06**	**81.81**

V. Conclusion

In this paper, we integrate a pretrained BERT model with a post processing module to complete the task of slot language

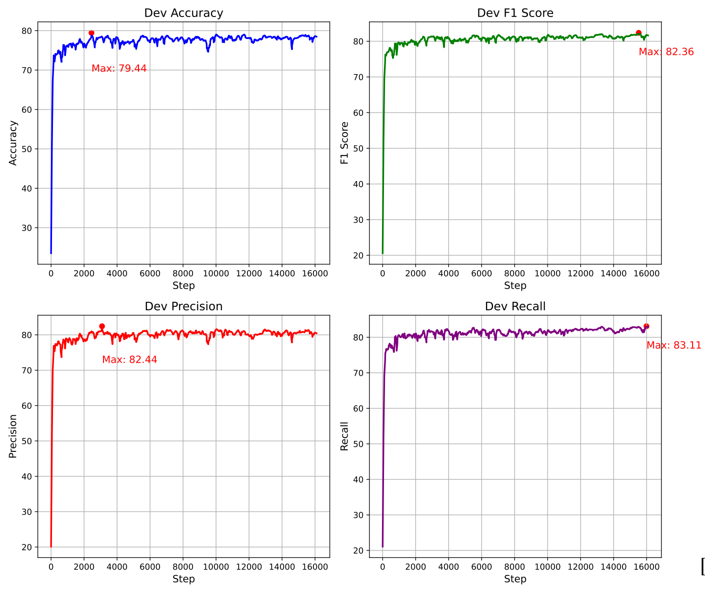

Fig. 3. Key metrics of the model. From the figures, we can see that the model quickly converges to a relatively high accuracy.

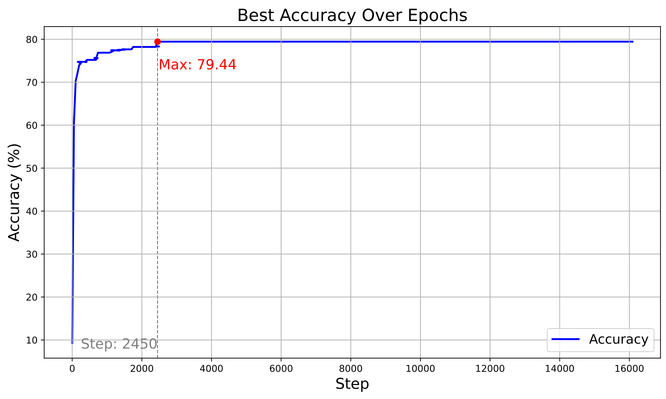

Fig. 4. Best Accuracy on Development Dataset During Training Procedure

understanding. By leveraging the Lexicon Adapter, we intend to merge word-level and character-level information to facilitate the training. We find that this achieves better performance on the given test dataset. However, there is still room for improvement, as there is currently no apporach for predicting the result input if the value itself is not in `asr`. This could be resovled by introducing generative structure in later works.

We thank Prof. Lin and the TA Jushi Kai for their contributions throughout the course.

REFERENCES

[1] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

[2] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[3] H. Wang, J. Li, H. Wu, E. Hovy, and Y. Sun, "Pre-trained language models and their applications," *Engineering*, vol. 25, pp. 51–65, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2095809922006324

[4] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[6] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," *Advances in neural information processing systems*, vol. 28, 2015.

[7] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," *arXiv preprint arXiv:1508.05326*, 2015.

[8] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[9] S. Gooding and E. Kochmar, "Complex word identification as a sequence labelling task," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 1148–1153. [Online]. Available: https://aclanthology.org/P19-1109

[10] M. Rei, G. K. Crichton, and S. Pyysalo, "Attending to characters in neural sequence labeling models," *arXiv preprint arXiv:1611.04361*, 2016.

[11] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[12] L. S.-T. Memory, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 2010.

[13] M. Rei, "Semi-supervised multitask learning for sequence labeling," *arXiv preprint arXiv:1704.07156*, 2017.

[14] J. Yang, Z. Teng, M. Zhang, and Y. Zhang, "Combining discrete and neural features for sequence labeling," in *Computational Linguistics and Intelligent Text Processing: 17th International Conference, CICLing 2016, Konya, Turkey, April 3–9, 2016, Revised Selected Papers, Part I 17*. Springer, 2018, pp. 140–154.

[15] H. He and X. Sun, "A unified model for cross-domain and semi-supervised named entity recognition in chinese social media," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.

[16] Y. Zhang and J. Yang, "Chinese ner using lattice lstm," *arXiv preprint arXiv:1805.02023*, 2018.

[17] P. Cao, Y. Chen, K. Liu, J. Zhao, and S. Liu, "Adversarial transfer learning for chinese named entity recognition with self-attention mechanism," in *Proceedings of the 2018 conference on empirical methods in natural language processing*, 2018, pp. 182–192.

[18] M. Shen, W. Li, H. Choe, C. Chu, D. Kawahara, and S. Kurohashi, "Consistent word segmentation, part-of-speech tagging and dependency labelling annotation for chinese language," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 298–308.

[19] T. Gui, R. Ma, Q. Zhang, L. Zhao, Y.-G. Jiang, and X. Huang, "Cnn-based chinese ner with lexicon rethinking." in *ijcai*, vol. 2019, 2019.

[20] R. Ma, M. Peng, Q. Zhang, and X. Huang, "Simplify the usage of lexicon in chinese ner," *arXiv preprint arXiv:1908.05969*, 2019.

[21] X. Zhao, M. Yang, Q. Qu, and Y. Sun, "Improving neural chinese word segmentation with lexicon-enhanced adaptive attention," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1953–1956.

[22] Y.-Y. Wang, L. Deng, and A. Acero, "Semantic frame-based spoken language understanding," *Spoken language understanding: systems for extracting semantic information from speech*, pp. 41–91, 2011.

[23] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu *et al.*, "Using recurrent neural networks for slot filling in spoken language understanding," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 530–539, 2014.

[24] R. Pieraccini, E. Tzoukermann, Z. Gorelov, J.-L. Gauvain, E. Levin, C.-H. Lee, and J. G. Wilpon, "A speech understanding system based on statistical representation of semantics," in *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1. IEEE, 1992, pp. 193–196.

[25] S. Miller, R. Bobrow, R. Ingria, and R. Schwartz, "Hidden understanding models of natural language," in *32nd Annual Meeting of the Association for Computational Linguistics*, 1994, pp. 25–32.

[26] Y. He and S. Young, "A data-driven spoken language understanding system," in *2003 IEEE Workshop on Automatic Speech Recognition and Understanding (IEEE Cat. No. 03EX721)*. IEEE, 2003, pp. 583–588.

[27] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[28] M. Henderson, M. Gašić, B. Thomson, P. Tsiakoulis, K. Yu, and S. Young, "Discriminative spoken language understanding using word confusion networks," in *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2012, pp. 176–181.

[29] L. Deng, G. Tur, X. He, and D. Hakkani-Tur, "Use of kernel deep convex networks and end-to-end learning for spoken language understanding," in *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2012, pp. 210–215.

[30] Y. Bengio *et al.*, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[31] W. Liu, X. Fu, Y. Zhang, and W. Xiao, "Lexicon enhanced chinese sequence labeling using bert adapter," *arXiv preprint arXiv:2105.07148*, 2021.

[32] M. Xu, "Text2vec: Text to vector toolkit," https://github.com/shibing624/text2vec, 2023.

[33] L. Horne, M. Matti, P. Pourjafar, and Z. Wang, "GRUBERT: A GRU-based method to fuse BERT hidden layers for Twitter sentiment analysis," in *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop*, B. Shmueli and Y. J. Huang, Eds. Suzhou, China: Association for Computational Linguistics, Dec. 2020, pp. 130–138. [Online]. Available: https://aclanthology.org/2020.aacl-srw.19

[34] M. Xu, "Pycorrector: Text error correction tool," https://github.com/shibing624/pycorrector, 2023.