Fall 2020 NLP Final Project Report

Ruocheng Shan **Xuzheng Lu** **Ruikai Zhou** **Kai Wang**
shanruocheng@gwu.edu
xuzheng_lu@gwu.edu
rkzhou_1997@gwu.edu
wangkai314@gwu.edu

1 Introduction and Background

In previous study, we introduced the task of Named Entity Recognition (NER). We have studied the it's history of developing and methodologies. In this group project, we use two different data set and implement 4 different models to perform NER. In order to compare the performance, we use accuracy, precision, recall and F1-score as evaluation metrics. In this report, the implementation of models will be presented along with performance analysis.

2 Data

2.1 CoNLL-2000

The CoNLL-2000 shared task as dividing text into syntactically related non-overlapping groups of words, so-called text chunking. All chunks were automatically extracted from the parsed version of the Treebank, guided by the tree structure, the syntactic constituent labels, the part-of-speech tags and by knowledge about which tags can be heads of which constituents.(Tjong Kim Sang and Buchholz, 2000)

The CoNLL-2000 dataset has 8396 sentences for training and 2012 sentences for testing. There are 21 different tags for words.

2.2 ScienceExamCER

ScienceExamCER is a densely-labeled semantic classification corpus. Labels including taxonomic groups, meronym groups, verb/action groups, properties and values, and synonyms. (Smith et al., 2019)

The ScienceExamCER dataset has 15129 sentences for training and 4923 sentences for testing. There are 1050 different tags for words.

3 Models and Implementations

3.1 Hidden Markov Model

Given a sequence of observations, in NER task the tokens' sequence, and the states, the tags for each token, HMM uses Viterbi algorithm to find the most likely tag sequence in the state space of the possible tag distribution based on the state transition probabilities.

We use HMM as the baseline model.

3.2 Conditional Random Field

By introducing a custom feature function, it can not only express the dependence between observations, but also express the complex dependencies between the current observation and multiple states before and after.

Currently, plenty of CRF packages are readily available as open source for segmenting or labeling sequential data. The CRF model of NER consists of mainly data training and data testing. Figure 1 briefly shows the main structure of this CRF model we have used.

For the project, we used the sklearn_crfsuite (Korobov, 2017) package to implement CRF model. We also used cross validation for tunning two coefficient: c1 for L1 regularization and c2 for L2 regularization.

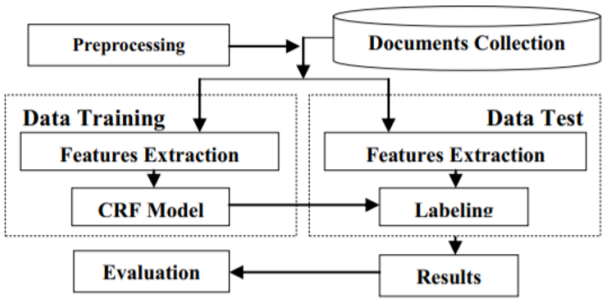

Figure 1: Conditional Random Field for NER (Pedregosa et al., 2011)

3.3 LSTM with CRF

In modern approaches, NER tasks are often performed by end-to-end deep learning models. An advanced method is to use the Embedding layer for extracting the distributed representations for inputs, the Bi-directional LSTM layer for context encoding, and the CRF layer for tag decoding. Figure 2 shows the structure of Bi-LSTM with CRF network.

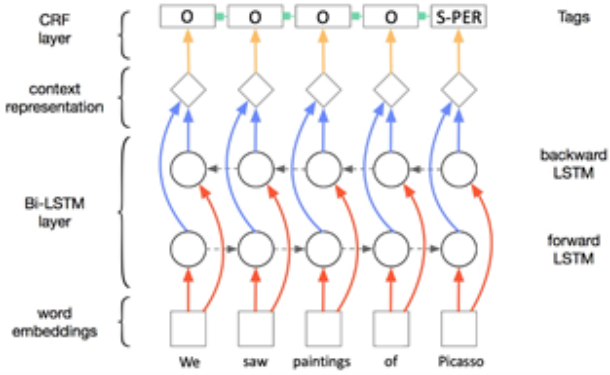

Figure 2: Bi-LSTM and CRF for NER

Word-embedding can be used to represent the words' meaning better and forces the model to pay attention to the local dependencies. In the embedding layer, words or phrases from the vocabulary will be mapped to embedding vectors. One of the useful nature of word embedding is that the distance between similar semantic words is smaller than words that are not semantic related. Therefore, the embedding layer could be used for dimensionality reduction and extracting the representations for data.(Mulla, 2020)

Bi-LSTM is used as a context encoder. As an advanced version of LSTM, it is good at obtaining contextual semantic information and short- and long-term dependencies. A Bi-LSTM consists of two separate LSTM hidden layers and processes sequence data in both forward and backward directions(Cui et al., 2018). When training a Bi-LSTM, the inputs will be processed in two ways, one from past to future and one from future to past. Then, the LSTM hidden layers could preserve information from both the past and future at any time point (Aggarwal, 2019).

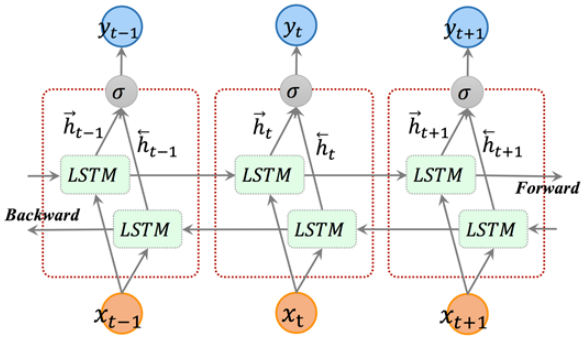

Figure 3: Bi-LSTM Network Structure (Cui et al., 2018)

Figure 3 shows the sturcture of a bidirectional long short term memory recurrent neural network.

CRF layer is added to the end of the model as a tag decoder. It could add constraints to the final predicted labels to ensure that the results are valid and robust. The CRF layer can automatically learn these constraints on the training data set during the end-to-end training process and gives the model the ability to view the whole picture when decoding.

We used Keras(Chollet et al., 2015) to implement the Embedding layer and LSTM layer. The CRF layer was implemented via an extension repository of Keras: keras-contrib.

3.4 BERT

BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary.(Devlin et al., 2018)

As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore it is considered bidirectional, though it would be more accurate to say that it's non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).(Horev, 2018)

We used simpletransformers(ThilinaRajapakse, 2020) library to implement BERT.

4 Results

4.1 Evaluation Metrics

For CoNLL-2000:

Model	Pre.	Rec.	F1.	Acc.
HMM	-	-	-	0.89
CRF	0.95	0.95	0.95	0.95
biLSTM+CRF	0.92	0.92	0.92	0.92
BERT	0.97	0.98	0.98	0.98

For ScienceExamCER:

Model	Pre.	Rec.l	F1.	Acc.
HMM	-	-	-	0.79
CRF	0.84	0.84	0.83	0.84
biLSTM+CRF	0.83	0.81	0.81	0.81
BERT	0.76	0.73	0.71	0.76

4.2 Analysis

For CoNLL-2000, the performance of BERT is the highest, however it is also the lowest for ScienceExamCER. Since BERT is a pre-trained model, and the words in CoNLL-2000 are very common, the performance is hence reasonable. One the other hand, the tokens in ScienceExamCER may not occur that often in common text, which leads to the low performance with BERT.

Some results for certain type of entities are really poor, because the number of training samples is too small. For example, the tag B-TheoryOfMatter only occurred twice in ScienceExamCER, and the scores are all zero for this one.

The overall accuracy of each model either exceeded or at least approached the baseline, thus we consider the these implementations successful.

5 Conclusion and Future Work

In this project, we implemented HMM, CRF, Bi-LSTM with CRF and BERT for named entity recognition. However, there are a lot more novel models or Neural Networks that are being considered as state-of-the-art models for NER.

For instance, Yamada et al. introduced the LUKE model which is the best model for CoNLL-2003 dataset. Researches are also using CNN to perform feature extraction in NER. Whats more, attention provides a solution to the bottleneck problem that in each decoding step, focus on a specific part of the source sentence.

The imbalance of samples is a good topic, as we have ran into such problem during this project.

References

Raghav Aggarwal. 2019. Bi-lstm.

François Chollet et al. 2015. Keras. https://keras.io.

Zhiyong Cui, Ruimin Ke, Ziyuan Pu, and Yinhai Wang. 2018. Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction. *arXiv:1801.02143*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *Google AI Language*.

Rani Horev. 2018. Bert explained: State of the art language model for nlp.

Mikhail Korobov. 2017. sklearn-crfsuite.

Zeeshan Mulla. 2020. Word embeddings in natural language processing — nlp.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Crf based name entity recognition (ner) in manipuri: A highly agglutinative indian language. *2nd National Conference on Emerging Trends and Applications in Computer Science*, pages 1–6.

Hannah Smith, Zeyu Zhang, John Culnan, and Peter Jansen. 2019. Scienceexamcer: A high-density fine-grained science-domain corpus for common entity recognition. *arXiv*.

ThilinaRajapakse. 2020. simpletransformers.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task chunking.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. Luke: Deep contextualized entity representations with entity-aware self-attention. *arXiv:2010.01057*.

A Appendices

Please find our project on github:

https://github.com/carlruikai/NLP-Final-Project