BiLSTM-CRF for Chinese NER

Jiyuan Lu (jl11046@nyu.edu) Licheng Shen(ls4330@nyu.edu)

Summary

Named Entity Recognition (NER) is an important area in the study of NLP. We implemented a BiLSTM-CRF model for Chinese NER problem and then modified the code and hyperparameters to get a better validation F1-score and training speed. Our implementation is based on a former project of Licheng that does BiLSTM-CRF for Chinese NER. We modified the code for it to be compatible with parallel computing and GPUs that gained a great boost on speed. Then we tried to tune hyperparameters of the model like dropout rate and learning rate to gain a better F1-score.

Motivation

The motivation of this project is to improve the result from Licheng's project that does BiLSTM-CRF for Chinese NER. Due to lack of knowledge of deep learning, the result he got from his project was very poor and the training speed was extremely low. We would like to improve that model with what we learned from this course and try to find the best hyperparameter for the problem.

Background

NER

Named Entity Recognition (NER) is an important area in the study of NLP. It plays an important role in many different NLP applications like information retrieval, machine translation, etc. In recent years, it is trending to use DL-based NER models and achieve state-of-the art results. Compared to classical NER models, DL-based NER has the advantage that it detects features automatically from the corpus and saves a significant effort on designing features.

Compared to English NER, some characteristics of Chinese language make it more difficult for NER. The first reason is that there is no capitalization in Chinese. In English, the initials of a named entity are usually capitalized, which is a convenient feature for detecting named entities, but in Chinese there is no such feature that we can rely on. Another reason is that in English, words in a sentence are separated by whitespaces, but Chinese sentences are not. Thus, a single sentence in Chinese may have multiple ways to be segmented into words, which makes it more difficult to determine the boundaries of named entities.

Recurrent Neural Networks

Recurrent Neural Networks (RNN) are a kind of neural networks that takes a sequence of input and in each neuron, there are hidden states served as "memory" of prior inputs in the sequence. Unlike Feedforward Neural Networks or Convolutional Neural Networks, the input of an RNN can be a sequence with arbitrary length, which makes it suitable for NLP tasks since we are handling sentences of different sizes. Also, the feature that prior inputs in the sequence can influence the output of current input coincides well with NLP models since the property or meaning of a word in a sentence is usually determined by not only the word itself, but also the context before or after it.

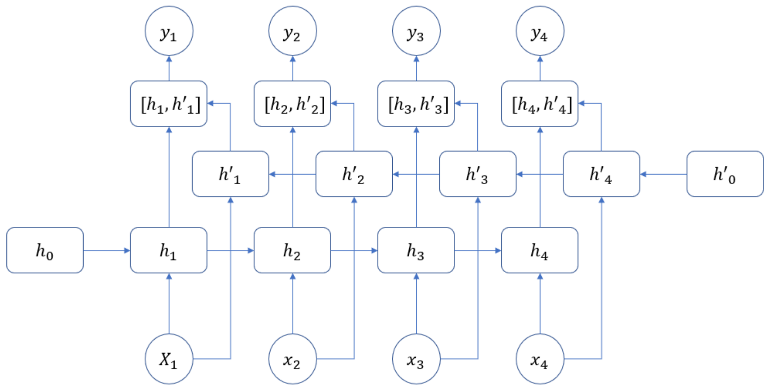

Figure 1. Basic structure of BiLSTM network

LSTM

Although in theory, RNN can capture long term dependencies, it turned out to be exceedingly difficult for it to do so. Multiplicative gradients can be exponentially decreasing or increasing with growth of the number of layers, which makes long term dependencies suffer severely from vanishing gradient problems.

Long Short-Term Memory Networks (LSTM) are introduced to tackle such problems. In LSTM, a cell is used to store long term memory in a hidden layer and several gates are implemented to control how inputs influence cell memory.

Note that hidden states and cell memory in LSTM implementations only reflect information from prior inputs, but in natural language, it is very common that words that come later also provide vital information for a current word. So, to capture such a feature, bidirectional LSTM is introduced. FIGURE 1.2 shows the basic structure of BiLSTM. It is a very simple trick that we use two LSTMs, the forward one gets the input sequence in normal order while the backward one gets the input in reverse order and then we concatenate them together as the hidden states output for x_t for our BiLSTM model.

CRF

The output of the BiLSTM network represents the emission score of each label for each input, but this assumes independence between two labels. In real languages, the transitions of labels are affected by rules like grammar. So a Conditional Random Field (CRF) is introduced at the end that takes the probability of transition between labels into account.

Implementation Details

Our experiment was run on Google Cloud Platform (GCP) with Nvidia K80 GPU. The model was implemented in PyTorch version 1.8. Our dataset comes from People’s Daily 2014 corpus and contains 50,000 characters and 5 features.

Experiment Design Flow

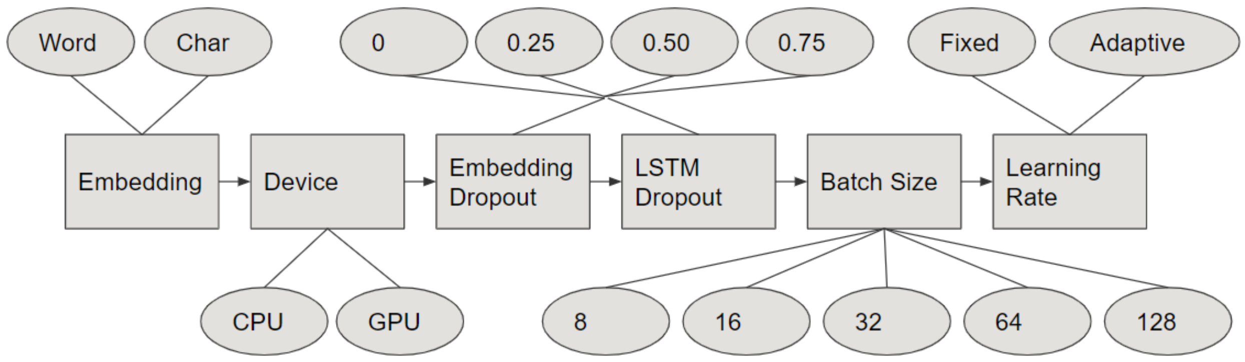

Figure 2. Experiment Design Flow

We have 6 different hyperparameters to experiment with.

First we decide whether our BiLSTM-CRF model uses a word embedding or a character embedding as input.

Second we decide whether to train our model on CPU or GPU.

Third and fourth we vary the dropout rate applied to the embedding layer and the LSTM output. We consider dropout rates of 0, 0.25, 0.5, 0.75 and their combinations.

Fifth we vary the batch size starting from 8 and increasing exponentially to 128.

Finally we decide whether to use a fixed base learning rate of 0.01 or to use an adaptive learning rate that scales logarithmically with the batch size.

Experiment Results

Experiment Result for Word Models

As can be seen from Figure 3, dropout on the LSTM output improves validation f1-score for word models. However, dropout on embedding has a negative effect for word models. Applying dropout to both layers does not further help improve performance and results in a performance just slightly better than applying dropout on the embedding layer.

Moreover, we notice that with a 0.75 dropout rate on the embedding layer, the model almost cannot learn anything.

This suggests that for word embedding models, one should apply dropout on the LSTM output only.

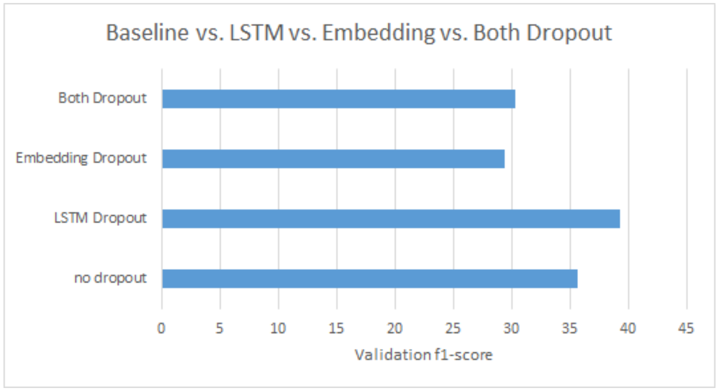

Figure 3. Highest validation f1-score for each scenario

Experiment Result for Char Models

As can be seen from Figure 4, dropout on embedding improves character models. However, dropout on the LSTM output has a negative effect for character models. Applying dropout to both layers does not further help improve performance and results in a performance slightly worse than applying dropout on the embedding layer.

Moreover, we notice that with a 0.5 dropout rate on the embedding layer, the model achieves the best validation f1-score across all experiments at 46.23%.

This suggests that for character embedding models, one should apply dropout after the embedding layer only.

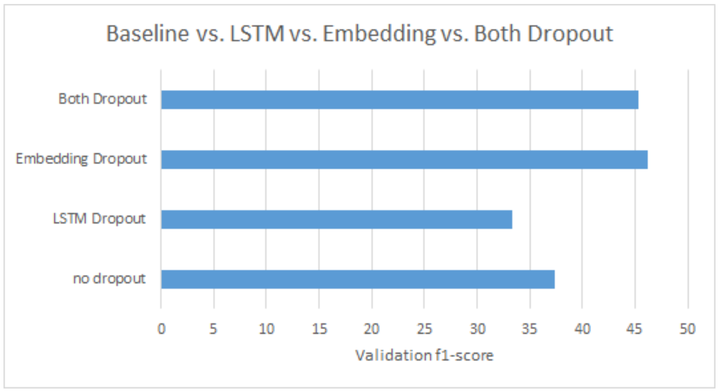

Figure 4. Highest validation f1-score for each scenario

Training Time Speed Up

As can be seen from Figure 5, parallelizing the model on GPU significantly improves the training speed. The speed up increases logarithmically with the batch size, achieving 4.72 for batch size of 128. The reason the speed up increases logarithmically rather than linearly might be that the CRF component in the model has to be executed sequentially.

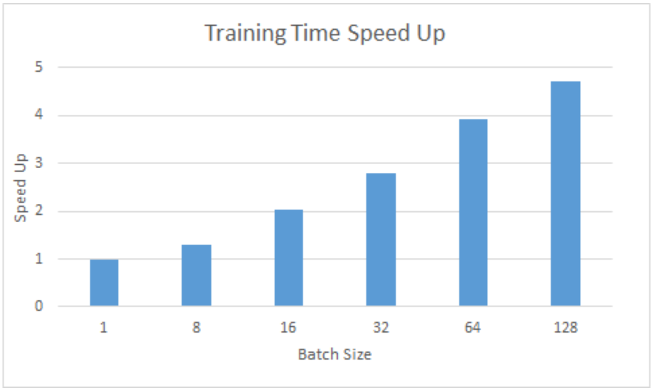

Figure 5. Training Time Speed Up

Time-To-F1-Score Speed Up & Adaptive Learning Rate

Figure 6 shows the speedup of time-to-f1-score (TTFS) metric on word models for different batch sizes and different learning rate schedules. The f1-score target is set to be 35%, which is the best validation f1-score we got on the baseline word model that has no dropout with batch size 1. For the fixed learning rate, we notice that there are some speedups for batch sizes 8, 16, and 32. However, when the batch size becomes 64 or higher, the model could not converge to 35% f1-score within 100 epochs. This is because we are making fewer updates with the increasing batch size. Our solution is to use the adaptive learning rate instead, which scales the learning rate logarithmically with the batch size. With the adaptive learning rate, the model can achieve a TTFS speedup of 12.55 with a batch size of 16. And the model is able to converge for batch sizes greater than or equal to 64.

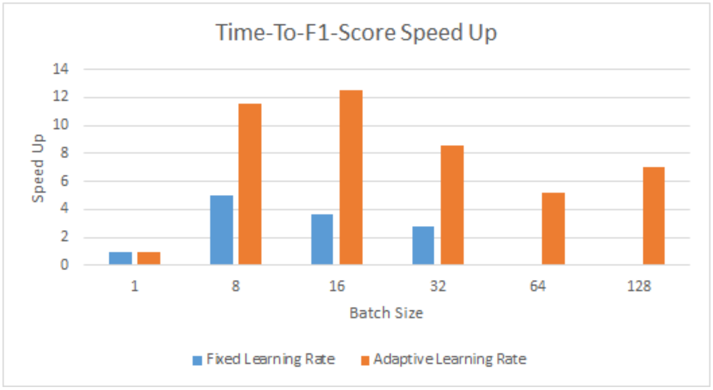

Figure 6. Time-To-F1-Score Speed Up

Conclusion

Training on GPU leads to considerable speed up and faster Time-To-F1-Score with adaptive learning rate for BiLSTM CRF model on Chinese NER tasks.

Dropout on embedding benefits char models but has a negative impact on word models.

Dropout in LSTM output benefits word models but has a negative impact on char models.

References

1. ADVANCED: MAKING DYNAMIC DECISIONS AND THE BI-LSTM CRF
https://pytorch.org/tutorials/beginner/nlp/advanced_tutorial.html

2. Deep Learning for NLP Best Practices https://ruder.io/deep-learning-nlp-best-practices/

3. J. Li, A. Sun, J. Han and C. Li, "A Survey on Deep Learning for Named Entity Recognition," in IEEE Transactions on Knowledge and Data Engineering, doi: 10.1109/TKDE.2020.2981314.
https://arxiv.org/pdf/1812.09449.pdf

4. Huang, Zhiheng, et al. “Bidirectional LSTM-CRF Models for Sequence Tagging.” *ArXiv.org*, 9 Aug. 2015, arxiv.org/abs/1508.01991.
https://arxiv.org/pdf/1508.01991.pdf

5. Deep Learning for NLP: An Overview of Recent Trends
https://medium.com/dair-ai/deep-learning-for-nlp-an-overview-of-recent-trends-d0d8f40a776d