Final Project Report: Comparing Different Models for Using BERT Embeddings in Part-of-Speech Tagging

Meng Li and **Peilu Lin** and **Siyu Tao**
Department of Language Science and Technology
Saarland University
66123 Saarbrücken, Germany
{mengli, peilulin, siyutao}@coli.uni-saarland.de

Abstract

This paper reports the implementation and results of our final project for the lecture Neural Networks: Implementation and Application (Winter 2020/21) at Saarland University. Two models are proposed and implemented to utilize pretrained word embeddings from BERT in a part-of-speech tagging task. Results and evaluation of the models show the benefit of using pretrained models in easily achieving state-of-the-art results but bring to attention the need for thoughtful instead of gratuitously complex architectures to make best use of information from pretrained models.

1 Introduction

One of the first steps for many natural language processing tasks is that of linguistic sequence labeling, among others part-of-speech (POS) tagging, named entity recognition (NER), and text chunking. The sophistication and performance of downstream tasks depend in part upon their output labels like the POS tags. In recent years, advances in deep learning models have seen them replacing traditional linear statistical models like Hidden Markov Models (HMM) and Conditional Random Fields (CRF) for the sequence labeling task, thanks to their ability to learn complex features without task-specific feature annotations and other resources.

Such neural models standardly consist of three modules: an embedding module, a context encoder module, and an inference module. The embedding module maps words into distributed word representations, i.e., word embeddings; the context encoder module extracts features from the input embeddings; the inference module predicts labels as output. Contextualized word embeddings were first introduced to solve the difficulty static context-independent embedding approaches such as word2vec (Mikolov et al., 2013) have dealing with with polysemous words. ELMo (Peters et al., 2018), for example, generates representation for each word given the entire sentence as input. Similarly a pretrained language model, BERT (Devlin et al., 2019) is trained on a masked word prediction task using a bidirectional transformer, i.e., a transformer-encoder that learns representations conditioned on contexts on both sides of the masked word. They exploit large amounts of unlabelled data and has brought significant improvement to a broad range of NLP tasks downstream.

It is no exaggeration to say that fine-tuning pretrained language models such as BERT has quickly become ubiquitous in NLP research since their advent. These pretrained language models learn universal language representations from huge text corpora and provide better initialization and generalization, speeding up convergence on specific tasks.

This paper presents results from our modest experiments with fine-tuning BERT for a POS tagging task. We use a pretrained BERT model to provide embeddings for texts from the OntoNotes 4.0 corpus and experiment with two methods of using these embeddings. In the first, we use them as input to a linear layer which will then predict tags based on the BERT embeddings; in the second, we use the BERT encoder as the embedding layer of a Bidirectional Long Short-Term Memory (BiLSTM) network. We compare the results from these two methods, including also two configurations of the BiLSTM model and show that pretrained BERT embeddings achieve a high accuracy on the test set even when combined only with a linear layer and that BiLSTM in fact hurts model performance by a small margin.

2 Architecture

In this section, we describe our neural network architectures. We first introduce the BERT and

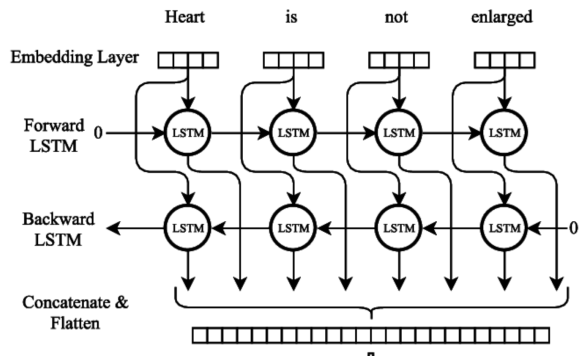


Figure 1: Bidirectional LSTM has forward and backward LSTMs (Image Source: Cornegruta et al. 2016)

LSTM components summarily before describing the BERT+Linear and BERT+LSTM architectures we experiment with.

2.1 BERT

BERT adopts the encoder from Transformers (Vaswani et al., 2017) to generate a language representation model. The model training introduces a masking mechanism: the entire sequence with 15% randomly masked words is run in BERT attention based encoder and the model predicts the masked words, given the context of non-masked words in the sequence. Note that we use the BERT tokenizer to tokenize our input texts, as we ideally need to format our input sequence in the same way in which the BERT model is trained.

2.2 BiLSTM

In sequence labeling tasks, Recurrent Neural Network (RNN)-based decoders are often used to capture long dependencies between predicted labels, conditioned on both the features of input and the previous predicted labels. LSTM is a variant of RNN capable of learning such long dependencies. Each LSTM cell has three gates controlling the flow of information: an input gate, a forget gate, and an output gate. The cell remembers values over arbitrary time intervals.

One problem with LSTM is that it takes information only from the one direction for context. A Bidirectional LSTM (BiLSTM) is a solution that consists of two LSTMs: one taking the input in a forward direction, and the other in a backwards direction. BiLSTMs effectively increase the amount of information available to the network, making more the context available in the model. Figure 1 shows the architecture of a BiLSTM.

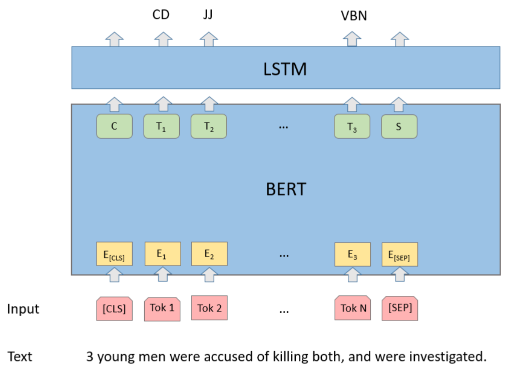


Figure 2: BERT + BiLSTM

2.3 BERT + Linear

The architecture of our **BERT + Linear** model is straightforward, with much of the complexity contained within BERT module. The BERT encoder is used as the embedding layer and we simply add a linear layer on top of these embeddings. For regularization, we apply a dropout layer between the BERT embeddings and another between the linear layer and the outputs.

2.4 BERT + BiLSTM

Similarly, our main modification of the BiLSTM model is simply using the BERT model as the embeddings layer. The input dimension and embedding dimensions are therefore taken over by the BERT model and the output embeddings are then input into the BiLSTM layers. Figure 2 shows our **BERT + BiLSTM** model architecture.

Within each BiLSTM layer, a forward LSTM processes the input sequence in the forward direction, while a backward LSTM processes the same sequence in the backward direction. Each layer of the BiLSTM also takes in the hidden and cell states from the previous layer and passes its own hidden and cell states to the next layer. The forward and backward hidden states of the final BiLSTM layer are concatenated and passed through a linear layer which is used to predict the outputs. Again, we apply a dropout layer between every two layers, including between the LSTM layers, for regularization.

3 Experiments

3.1 Data

We work with data from OntoNotes Release 4.0 (Weischedel et al., 2011), which we preprocess to keep only word tokens and their corresponding

POS tags. We split the corpus into training, validation, and testing sets, with 55904 sequences for training and 6988 sequences each for validation and testing.

3.2 Training

For the BERT + Linear model, the only hyperparameter is the dropout rate, which we set to 0.25. In total, this model has 108,348,722 trainable parameters. We use an Adam optimizer (Kingma and Ba, 2014) and set the learning rate to `5e-5`, one of the values suggested in Devlin et al. (2019) for fine-tuning, as we do not want drastic changes in our pretrained model.

We train two different configurations of our BERT + BiLSTM model, one with a single BiLSTM layer and another with two layers. All other hyperparameters are identical for the two instances. As is with the BERT + Linear model, the dropout rate is still set to 0.25 and the learning rate to `5e-5`. Other hyperparameters include a hidden dimension of 128. Our code implementation allows for the model to be instantiated as a regular LSTM too, although we only experiment with BiLSTMs in this paper. We additionally note that the configuration with a single BiLSTM layer has one fewer dropout layer otherwise placed between the BiLSTM layers.

For each of the models, we train with a relatively small batch size of 32, given that the BERT model is already quite large. We use Weights & Biases (`wandb`, Biewald 2020) for experiment tracking and visualizations.

3.3 Results and Discussion

BERT +	precision	recall	f1-score
Linear	**0.9603**	**0.9596**	**0.9598**
1-layer BiLSTM	0.9601	0.9591	0.9594
2-layer BiLSTM	0.9556	0.9546	0.9548

Table 1: Micro-averaged evaluation results

Results from our three models (**BERT + Linear**, **BERT + 1-layer BiLSTM** and **BERT + 2-layer BiLSTM**) are reported in terms of precision, recall and f1-score, as evaluated on the test set. We trained each model for 10 epochs. Each epoch took around 6.5 minutes to train with a NVIDIA Tesla P100 GPU on Google Colaboratory. The micro-averaged results for each model are reported in Table 1.

Across the board, The results are above 95% and all within 1 percentage point of each other. The high accuracy of all the models using BERT embeddings is in line with our expectation and previous work and shows that fine-tuning pretrained models can easily produce state-of-the-art results.

However, we observe that the **BERT + Linear** model performs the best across all three metrics and the addition of BiLSTM layers proves to in fact hurt model performance, even though only by less than a percentage. The results from the **BERT + Linear** and **BERT + 1-layer BiLSTM** are even almost identical. We recall that BERT word embeddings are already contextualized and conditioned on both sides of the context and hypothesize that the additional contextualization by BiLSTM provides little benefit for the POS tagging task if at all. In fact, the additional layers may have functioned to obscure information from the BERT embeddings instead, even though the performances converge with more epochs of training. See also Figure 3 for plots of training losses and accuracies of the models.

4 Conclusion

In conclusion, this paper presented results from experimenting with fine-tuning pretrained BERT model for POS tagging. We implemented a BERT + Linear and a BERT + BiLSTM model and the results from training showed that all models performed similarly but the simplest BERT + Linear model in fact performed the best though by a small margin. The results show that pretrained models such as BERT can provide powerful improvement to NLP tasks even with the simplest architecture. At the same time, the decrease in performance brought about by the additional BiLSTM layers is also a reminder that more is not necessarily better and that, when using pretrained models, it is important to design model architecture in a way that best facilitates information from pretrained models instead of obscuring it.

Note on Reproducibility. All our implemented code and instructions for reproducing our results are made available in the GitHub repository at

`https://github.com/siyutao/nnia-project`

References

Lukas Biewald. 2020. Experiment tracking with weights and biases. Software available from wandb.com.

Savelie Cornegruta, Robert Bakewell, Samuel Withey, and Giovanni Montana. 2016. Modelling radiological language with bidirectional long short-term memory networks. In *Proceedings of the Seventh International Workshop on Health Text Mining and Information Analysis*, pages 17–27, Auxtin, TX. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

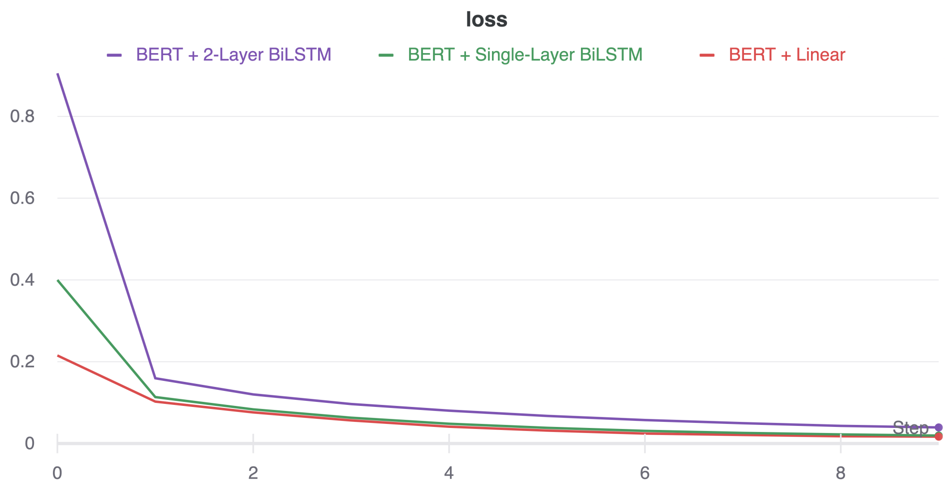
Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

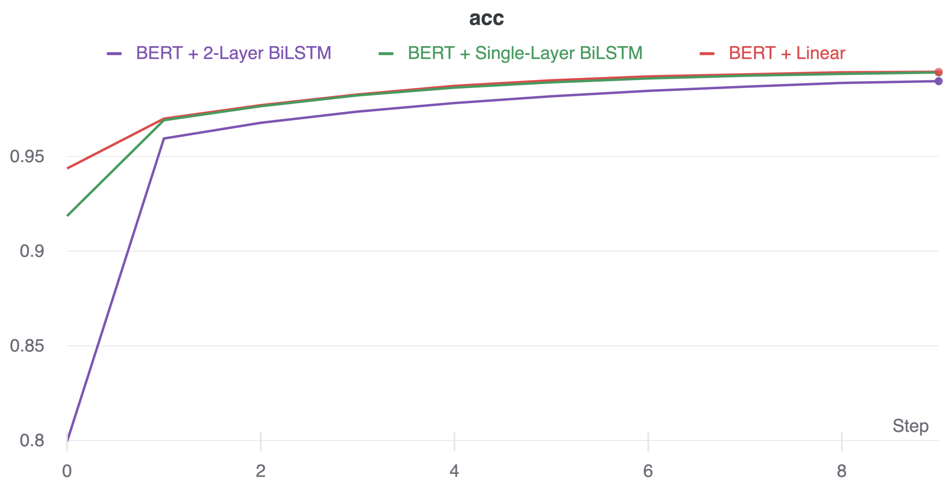
Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. 2011. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*.



(a) loss



(b) accuracy

Figure 3: Training loss and accuracy from the models by step/epoch.