Size matters?

Submitted by Nebojsa Pepic — Nicholas Jennings

Abstract

The state of the art NLP model BERT is responsible for great improvements in a variety of fields. BERT is a pre-trained model, which means that it should be able to perform fairly well on small datasets. At the same we assume that more data leads to better performance. The present report attempts to examine to what extent using larger amounts of data actually improves the performance of a non fine-tuned BERT Model.

1 Introduction

In past years there has been steady improvement in the field of natural language processing with neural models. The most recent major development has been BERT(devlin et al,2019)[2], which has greatly increased performance on a variety of NLP tasks. BERT is a model that is already pretrained to understand language and our goal is to find out how BERT performs with minimal fine tuning. To do this we look at the task of Part-of-Speech(POS) tagging by training on samples from the ontonotes 4.0 dataset[5] of various sizes and examining the results.

2 Methods

2.1 BERT

BERT(Bidirectional Encoder Representations from Transformers) is a model that learns contextual relations between words or sub-words in texts with the help of Transformer. Specifically Transformer contains an encoder to read input text. Compared to directional methods, the encoder reads an entire sequence of words at once instead of instead of reading the input sequentially, which allows model to learn the context of word based on all of its surroundings. Base BERT uses to two main training strategies: Masked LM and Next sentence prediction.[3]

Masked LM: 15 percent of words in sequence are replaced with a [MASK] token, before it is fed into BERT. The model then predicts the masked words based on the rest of the sentence. The loss function in BERT only regards the prediction for masked values and ignores non-masked words. This means that it converges slower compared to directional models, but gains increased contextual awareness.[3]

Next sentence prediction: During training the model receives pairs of sentences as the input, where in 50 percent of pairs the second sentence(sentence B) follows the first(sentence A) in the original document. In the other 50 percent of pairs sentence B is chosen at random. The input is modified by inserting a [CLS] classification token before sentence A and a separator token [SEP] is inserted after both sentences. The input then looks like this: [CLS] Sentence A [SEP] Sentence B [SEP]. Each token is given a sentence embedding indicating sentence A or B and position embedding indicating its position in the sequence.[3]

Both components are trained simultaneously in order to minimize their combined loss function. Our task is a named entity recognition task(NER), specifically classifying tokens in a sentence with POS-tags. To do so we adapt NER model[4] from HuggingFaces to suit our purposes. The BERT embedding we use is bert-base-cased[1], which works just like previously explained base model and takes case into account.

2.2 Data and preprocessing

As for the dataset, we used the Ontonotes 4.0 and smaller sample dataset for english language. The Ontonotes 4.0 dataset 69.882 sequences and 1.587.449 tokens. In the sample data we had 311 sequences and 6101 Tokens. The data was provided in .conll format and for our purposes we processed

Datasets	Sequences	Tokens
Train	279	5.491
Test	32	610

Table 1: Sample data

Datasets	Sequences	Tokens
Train	62.893	1.428.704
Test	6989	158.745

Table 2: Ontonotes 4.0

the data, extracting position of tokens, tokens and the corresponding part-of-speech tag for each token into a .csv dataframe. The data was then split into train and validation sets using a 90:10 ratio resulting in the numbers which can be seen in Table 1 and Table 2.

2.3 Training

We used a pre-trained BERT model from the Huggingfaces/transformers library on both of our preprocessed datasets using the following configuration:

- Task: pos-tagging
- Batch-size: 8
- Training-Epochs: 1
- Bert-model: bert-base-case

We did not do any further finetuning, as our goal was simply to show how the models performance is affected when it is trained on a larger dataset compared to a significantly smaller one. Therefore this method should not be regarded as an optimal approach to achieving high performance on pos-tagging. Fine-tuning the model itself on a smaller size of data might achieve similar results with potentially faster runtime and an overall more efficient model.

Modelstatus	Precision	Recall	F1	Accuracy
Pre-training	2.82	2.48	2.64	3.10
Trained	56.56	45.47	50.29	61.90

Table 3: Sample

Modelstatus	Precision	Recall	F1	Accuracy
Trained	97.52	97.62	97.57	98.23

Table 4: Ontonotes 4.0

3 Results

We evaluated the models performance on both datasets and provided pre-training and trained metrics. (Table 3 & Table 4) On the sample dataset after training and evaluation, we got an accuracy score of 61.90%. (Table 3) On the significantly larger ontonotes 4.0 dataset the model had an accuracy up to 98.23%.(Table 4) In Figure 1. we can see the validation loss after $\tilde{7}$500 evaluation steps. The curvature indicates a good and steady loss with few validation errors.

The results show that training and evaluation loss are very similar (Figure 1 & Figure 3), but the curvature is different. What is interesting about Figure 1. is that the training loss reduces really quick before the 1000 step mark, indicating a high learning rate of the model. This can be also seen in Figure 2. as the accuracy hits 97.48% after the 1000 step mark. These results further support the initial doubt that the amount of data (Ontonotes 4.0) is not necessarily needed to improve the performance on the given task (POS-tagging) by a great amount. However one should not disregard the improvement even though it is minor going further ahead of the step 1000 mark.

Figure 1: Plot 1. Validation Loss

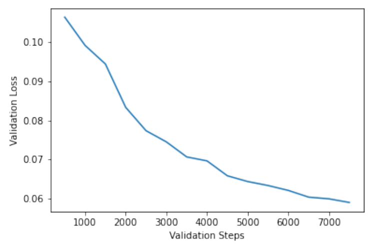


Figure 2: Plot 2. Evaluation Accuracy

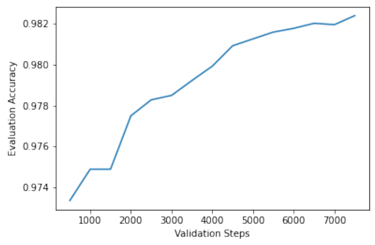
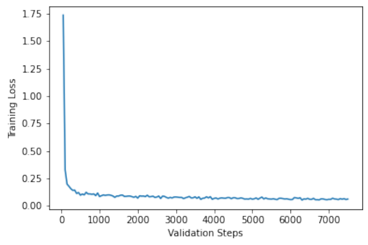


Figure 3: Plot 3. Training Loss



Upon seeing the results of training on the entire ontonotes dataset, we decided to test our model smaller subset(table 5). We found that training on this subset led to a significant boost in performance compared to the first sample and it even performs almost as well as when use full dataset(see table 6).

4 Conclusion

Looking at the results we could show that the size of training data matters to an extent. We improved our models performance by 37% with a greater trainingset. However we can see that we can achieve similar results with less data (Table 6.) and we might even use less data by providing a better tuning of our model itself as we were only training on a single epoch and did not take our data's distribution into consideration. Those are some possibilities we could have adapted, but as stated earlier, this was not the purpose of the present report. We succeeded in demonstrating that training size can matter, to some extent.

Datasets	Sequences	Tokens
Train	7195	110.356.3
Test	801	12.261.7

Table 5: large Sample data

Modelstatus	Precision	Recall	F1	Accuracy
sample	56.56	45.47	50.29	61.90
large sample	97.19	97.34	97.27	97.98
ontonotes	97.52	97.62	97.57	98.23

Table 6: large sample

References

[1] *bert-base-cased*. https://huggingface.co/bert-base-cased/.

[2] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[3] Rani Horev. *BERT Explained: State of the art language model for NLP*. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270/. 2018.

[4] *huggingface's NER example*. https://github.com/huggingface/transformers/blob/master/examples/token-classification/run_ner.py/.

[5] *OntoNotes Release 4.0*. https://catalog.ldc.upenn.edu/LDC2011T03. 2011.