Fake News Detection

Varshil Gandhi

August 2019

1 Introduction

LIAR-PLUS is a benchmark dataset for fake news detection, released recently. This dataset has evidence sentences extracted automatically from the full-text verdict report written by journalists in Politifact. It consists of 12,836 short statements taken from POLITIFACT and labeled by humans for truthfulness, subject, context/venue, speaker, state, party, and prior history. For truthfulness, the LIAR dataset has six labels: pants-fire, false, barely-true, half-true, mostly-true, and true. These six label sets are relatively balanced in size.

The task was to achieve the highest possible accuracy for the above described task.

2 Model Design

The model design in NLP classification based problems can be broken down into two parts.

- Encoder Model design
- Decoder or Output model design

2.1 Encoder Model Design

The model uses BERT (Bidirectional Encoder Representations from Transformers) as the Encoder model for the Liar dataset inputs. Refer to the image to get a clear understanding of how the input has been fed. BERT

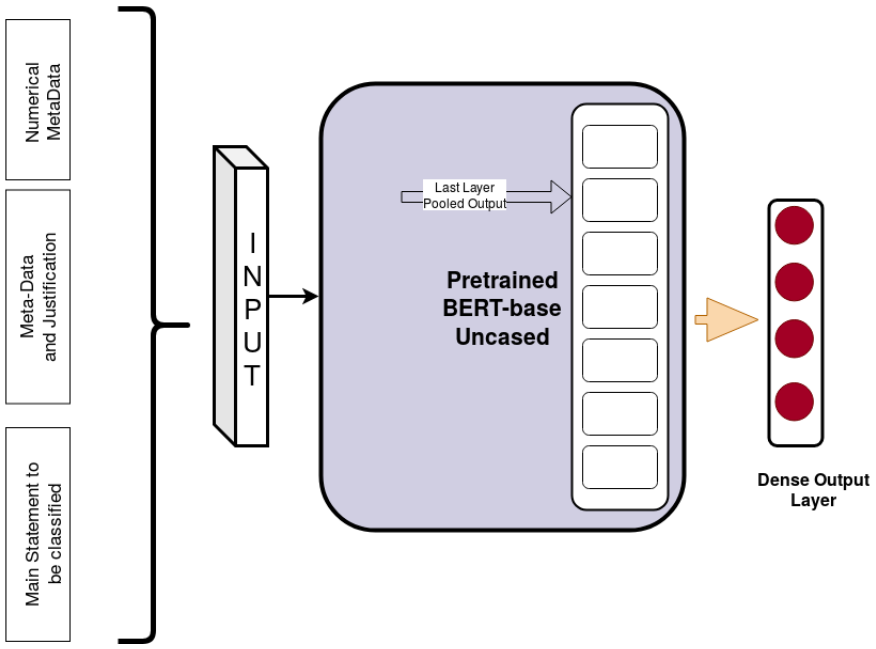

Figure 1: Model Structure

is a method of pre-training language representations, meaning that we train a general-purpose "language understanding" model on a large text corpus (like Wikipedia), and then use that model for downstream NLP tasks that we care about (like question answering).

Pre-trained representations can either be **context-free or contextual**, and contextual representations can further be unidirectional or bidirectional. Context-free models such as word2vec or GloVe generate a single "word embedding" representation for each word in the vocabulary, so *bank* would have the same representation in *bank deposit* and *river bank*. Contextual models instead generate a representation of each word that is based on the other words in the sentence.

2.2 Model Details

The model I designed uses **BERT pretrained model** for encoded representations and those encoded representations are pooled and given to a dense layer to output class probabilities. The model was trained on only 6-way

classification and not on 2-way classification because of time limitations. We do 2-way classification using the same model for 6 way classification for now. Cross entropy is used as the loss function and other parameters that are used are the ones generally used for BERT finetuning. Some of the important ones are mentioned below.

- Batch Size = 32
- Learning rate = 2e-5

3 Results

The testing was done on a whole lot of parameter variations with a variety of decoder layers. But the simple Dense Layer turns out to be the best compared to other type of sequential layers. The model was even trained with different changes in how the input is being concatenated. The current successful concatenation is as represented in the image.

To be noted: Time was less and so, I trained it only on 6 way classification problem. The model trained on 6 classes can also predict 2 classes which are nothing but batch of 3-3 classes in the previous case.

The final model training method: Training for 20 epochs

Final results		
BERT 20 epochs	Validation Accuracy	Test Accuracy
6 way classification	43.458%	41.897%
2 way classification**	0.70794%	71.30%

4 Code

The codebase contains a jupyter notebook for dataset analysis and cleaning. I did cleaning of corrupted data columns. There are 2-3 corrupted data rows in test2.tsv and 14-15 in train2.tsv. Analysis will help in getting to the conclusions.

Below mentioned settings are required for the main code:

- In the dataset folder, transfer the cleaned datafiles as train.tsv, val.tsv and test.tsv.
- In the pretrained models folder, extract the BERT uncased model from the BERT github page. Links below.
- Hardware specifications: Need GPU with atleast 14 Gb of memory. Google colab is the best option.

The below mentioned are the commands to run the code:

- For Training, Evaluation and Testing on 6 class classification:

```
python run_liar.py --task_name=liarplus --do_train=true --
   do_eval=true --do_predict=true --data_dir=./dataset/ --
   vocab_file=./pretrained_models/uncased_L-12_H-768_A-12/
   vocab.txt --bert_config_file=./pretrained_models/uncased_L
   -12_H-768_A-12/bert_config.json --init_checkpoint=./
   pretrained_models/uncased_L-12_H-768_A-12/bert_model.ckpt
   --max_seq_length=160 --train_batch_size=32 --learning_rate
   =2e-5 --num_train_epochs=20 --output_dir=./output_dir --
   save_checkpoints_steps=3198 --classification_class 6
```

- For Training, Evaluation and Testing on 2 class classification:

```
python run_liar.py --task_name=liarplus --do_train=true --
   do_eval=true --do_predict=true --data_dir=./dataset/ --
   vocab_file=./pretrained_models/uncased_L-12_H-768_A-12/
   vocab.txt --bert_config_file=./pretrained_models/uncased_L
   -12_H-768_A-12/bert_config.json --init_checkpoint=./
   pretrained_models/uncased_L-12_H-768_A-12/bert_model.ckpt
   --max_seq_length=160 --train_batch_size=32 --learning_rate
   =2e-5 --num_train_epochs=20 --output_dir=./output_dir --
   save_checkpoints_steps=3198 --classification_class 2
```

- Testing a trained model on test.tsv (Will show results for both 2 way and 6 way classification if the model trained on 6 way classification)

```
python run_liar.py --task_name=liarplus --do_train=false --
   do_eval=false --do_predict=true --data_dir=./dataset --
   vocab_file=./pretrained_models/uncased_L-12_H-768_A-12/
   vocab.txt --bert_config_file=./pretrained_models/uncased_L
   -12_H-768_A-12/bert_config.json --init_checkpoint=path/to/
   model.ckpt --max_seq_length=160 --output_dir=./
   output_results --classification {whatever the model was
   trained upon}
```

- Testing a trained model on val.tsv **(Its a bit hardcoded for now. Change the val.tsv in the dataset to test.tsv for a time being and then run the code.)

```
python run_liar.py --task_name=liarplus --do_train=false --
   do_eval=false --do_predict=true --data_dir=./dataset --
   vocab_file=./pretrained_models/uncased_L-12_H-768_A-12/
   vocab.txt --bert_config_file=./pretrained_models/uncased_L
   -12_H-768_A-12/bert_config.json --init_checkpoint=path/to/
   model.ckpt --max_seq_length=160 --output_dir=./
   output_results --classification {whatever the model was
   trained upon}
```

5 Links and References

- BERT Github
- BERT paper
- Model Download