CS410 Final Project: Text Classification / Twitter Sarcasm

Team: The Tardy Slackers
Wei Dai: weidai6@illinois.edu
Michael Huang: mh54@illinois.edu
Tom McNulty: tmcnu3@illinois.edu

An overview of the function of the code (i.e., what it does and what it can be used for).

The function of the code provided is text classification. Specifically, the model is meant for classifying whether a response tweet given its context is sarcastic or not. While this code can be used for classifying any type of text into any number of classifications with some minor adjustments, the code has been specifically designed to fit the text in the provided testing and training files.

The program takes in the provided 3-part Twitter interactions made up of 1 tweet that is a potentially sarcastic response and two tweets that occurred just before the response. These two tweets are collectively the “context”.

We use the RoBERTa pretraining language model. RoBERTa builds on the original BERT program which stands for Bidirectional Encoder Representations from Transformers, or BERT, is a revolutionary self-supervised pretrained architecture that has learned to predict intentionally hidden (masked) sections of text. RoBERTa builds on BERT’s language masking strategy and modifies key hyperparameters in BERT, including removing BERT’s next-sentence pretraining objective, and training with much larger mini-batches and learning rates. RoBERTa was also trained on an order of magnitude more data than BERT, for a longer amount of time. This allows RoBERTa representations to generalize even better to downstream tasks compared to BERT.

After reading in and preparing the data, we tokenize the data, train the RoBERTa model, do a prediction on the validation set, prepare the test input, perform a prediction on the test set and then prepare the output, which is a text document simply listing out where each line in the test file is sarcasm or not.

Documentation of how the software is implemented with sufficient detail so that others can have a basic understanding of your code for future extension or any further improvement.

Code: All code is labeled with comments for easy understanding. We extensively use the HuggingFace library for tokenization and model loading. We use Tensorflow to train the model.

Data Preparation:

1. Have train.jsonl and test.jsonl in folder called data in present working directory
2. We used an 80-20 train-val split

3. We simply combined the context together and appended it to the response tweet and used that as a string, using only the first 128 tokens in our model as max input sequence length

Training:

1. Start training from pretrained roberta-large and finetune with our dataset for 3 epochs with Adam optimizer with learning rate of 2e-5 with batch size of 16.

Results:

1. Model achieves .84 F1 score on validation set(internally) and .777192 F1 on test set / leaderboard(under tmmai)

Documentation of the usage of the software including either documentation of usages of APIs or detailed instructions on how to install and run the software, whichever is applicable.

Installation:

1. Clone the repo
2. Cd into repo and run pip install --upgrade -r requirements.txt, make sure you are using python3

Data Preparation:

1. Have train.jsonl and test.jsonl in folder ./data/

Run code (data transformation, training, testing):

1. Run python cs410_FINAL.py
2. This code will train the model, then it will evaluate on validation set and print f1 score
3. It will also write a data/answer.txt file with predictions on tests
4. It will also save the model in huggingface hf format as data/roberta_model. In this folder, it will contain the tensorflow h5 model

For running the colab notebook .ipynb file:

This software is largely self-contained and does not require significant installation and running effort. In the interest of time, we advise that you run this in the cloud rather than a laptop.

Before running the code, be sure to run “pip install transformers”

Then, All the key downloads, such as TensorFlow, Keras, RoBERTa, sklearn, transformers, etc are already coded to download automatically. The training and testing folders of data should be in your working directly and then within a folder called “data”. e..g. working/data/train.jsonl)

For the parameters, we adjusted the max length for tokenization, training encodings and validation encodings, ranging from 64 to 256. We ran the program in the cloud using Google

Colab and purchased the pro version. However, max lengths over 150 exhausted resources, so we were not able to see the results from those larger max lengths. We also experimented with adjusted the learning rates from 2e-5 up and down just slightly, such as 19e-6 and 21e-6 but these met with poor results. We later learned that the model is optimized for 1e-5 to 5e-5 going up a full number up each time (2e-5, 3e-5, etc). Anything in between supposedly will not work well.

Since we were getting results that put us in the top 5 on the leaderboard, and since the Google Colab took about 20 minutes to run each option, we only experimented with about 12 combinations.

Maxlength of 128 for tokenization, training and validation with a learning rate of 2e-5 delivered the best results

Brief description of the contribution of each team member in case of a multi-person team.

Wei Dai: Researched ideas, attempted to develop code in parallel to the team. Studied theories under BERT and RoBERTa. Actively discuss questions with other team members. Learned tensorflow, transformers and other libraries. Tried to discover parameters following Michael and Tom's draft.

Michael Huang: Researched ideas/models and determined RoBERTa was likely our best model to use. Developed the best working code of the team and provided good resources for the team to learn more about implementing BERT and RoBERTa.

Tom McNulty: Team captain, set up meetings, drafted the first drafts of all deliverables and requested team members to edit. Researched initial ideas and models, tried to develop a decent draft of text classification code in parallel with other team members but Michael developed the superior draft. Then worked on adjusting Michael's code/ parameters to optimize results. Contributed significantly to final documentation.

Citations / Resources

RoBERTa model: https://arxiv.org/abs/1907.11692
BERT model: https://www.aclweb.org/anthology/N19-1423/
Huggingface blog / libraries: https://huggingface.co/transformers/model_doc/roberta.html