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

Demo video / code walkthrough: https://youtu.be/OUu71EapO5U

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. Tokenize data with huggingface tokenizer to get encodings, then put in a tensorflow dataset for training
2. 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.

Validation and Test results:

1. Predict on validation and get f1 score, then predict on test set and write to answer.txt file and upload to leaderboard
2. Model achieves .84 F1 score on validation set(internally) and .777192 F1 on test set / leaderboard(under tmmai)

Experimentation:
For the parameters, we adjusted the max length for tokenization, training encodings and validation encodings. We tried lengths of 64, 80, 128, 150, 256. We also experimented with adjusted the learning rates from 2e-5 up and down just slightly, such as 1e-5 and 3e-5 but these met with poor results.

We also tried Bert-large, but this gave worse results than roberta-large. Maxlength of 128 for tokenization, training and validation with a learning rate of 2e-5 delivered the best results.

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.

Two ways to run code: as python scripts or on google colab as notebook

For running python scripts:

1. Clone the repo
2. Cd into repo and run pip install --upgrade -r requirements.txt, make sure you are using python3
3. Run python cs410_FINAL.py for training and testing. If you want to simply test on validation data and test data, then run python cs410_FINAL_EVAL.py
4. Both codes will evaluate on validation set and print f1 score
5. It will also write a data/answer.txt file with predictions on tests
6. If you use the training script, it will also save the model in huggingface hf format as data/roberta_model. In this folder, it will contain the tensorflow .h5 model
7. If using validation script, download our model from here: https://drive.google.com/drive/folders/1raCd-g4chwuufv1JBDNXyNxITJDMk_K2. Place

these files in a folder called roberta_model within the data subdirectory i.e. data/roberta_model

For running the colab notebook:

1. Download our model from here: https://drive.google.com/drive/folders/1raCd-g4chwuufv1JBDNXyNxITJDMk_K2.
2. Get our train.jsonl and test.jsonl files and place them in your google drive in subdirectory called data i.e. drive/MyDrive/data/train.jsonl
3. Run through our notebook, https://colab.research.google.com/drive/1zcMMw8xe6vh9rMPlBB_i-HZGrxsk5UJj#scrollTo=Qbr6a4LYcl_w, but skip training step. Instead, load our model into your drive and run on validation and test set. Then, submit answer.txt from last cell to livedatalab.

Link to the demonstration video / code walkthrough: https://youtu.be/OUu71EapO5U

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, trained the RoBERTa model to beat baseline, and provided good resources for the team to learn more about implementing BERT and RoBERTa. Also created python script to run model and contributed heavily to documentation showing how to run software.

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. Extensively experimented with various hyper parameter and Colab settings to get to optimal results. Contributed heavily 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
Keras: https://keras.io/api/
** A lot of code utilizes huggingface and keras api's