Documentation

1. Project Submission

- Project Code: https://github.com/yhui288/ClassificationCompetition
- Presentation Video: https://drive.google.com/file/d/1WlLuUBaauCe_yQBEOWXbl8YxxzWSVEQF/view?usp=sharing

2. Overview of The Function of The Project

Our team participated in the Text Classification Competition. This competition aims to predict if tweets with given context are sarcasm. We are given training data with labels. The goal is to correctly predict if tweets in test data are sarcasm.

Team Member: Changcheng Fu (cf7), Jiaying Li (jl63), Yanghui Pang (yanghui2)
Leaderboard Name: Yanghui Pang
Rank: ~55

Project File Structure (Download our project here):

- train.py/train.ipynb: the program we train the classifier and predict the results for test data
- answer.txt: the file storing predicted results for test data
- saved_weight.pt: the saved model (the file is too large, we didn't push it to github)

3. Code Implementation

- The process of training (what train.py/train.ipynb does):

1) Install packages and import necessary packages
2) Preprocessing Data: Data Loading + Data cleaning
3) Import the pre-trained BERT model and tokenizer
4) Tokenize the data
5) Create train/validate dataloaders
6) Initialize a BERT model, optimizer and loss function
7) Define functions for training and evaluating
8) Start training the model (also save the best model)
9) Predict the test data with the saved model

- The models we tried to beat the baseline:

First, we try to use traditional models from sklearn to access this problem. We tried Naive Bayes, SVM, LogisticRegression, RandomForestClassifier, KNeighborsClassifier and SGDClassifier from sklearn. Among all these classifiers, the SGDClassifier performed the best which has the highest f1 among them, but the score still cannot beat the baseline.
Then, we analyzed the data we have and decided to train a convolutional neural net with pooling, but still found the result is not ideal. We also tried to use fine-tune

BERT model with linear layer and dropout layer, but we found that the loss rate is relatively high. Finally, we tried the pre-trained BERT basic model from huggingface. By referencing the research papers, we adjusted the some hyperparameters such as learning rate and the number of epochs, and also tried various loss function. This pre-trained BERT model with adjusted parameters give us prediction that could beat the baseline.

4. Usage of Software (how to run our code)

- Run Locally (With python version 3.8)
  pip3 install -r requirements.txt
  python3 train.py

Or

- Run on Colab (Tutorial for running our code on Colab)
  1) Upload "train.ipynb" from our project
  2) Create "data" folder, and upload "train.jsonl" and "test.jsonl" to "data" folder
  3) Run the code block step by step
- The trained model will be saved in "saved_weight.pt";
  Predicted results for test data will be saved in "answer.txt"

5. Division of Labor

- Changcheng Fu: Tried training a neutral network with CNN and pooling layer; Tried BERT with linear and dropout layer; Tried a pretrained BERT basic model, achieved the best performance and beat the baseline; Wrote the progress report.
- Jiaying Li: Tried built-in modules from sklearn such as RandomForestClassifier, KNeighborsClassifier, LogisticRegression; Wrote documentation; Presentation.
- Yanghui Pang: Wrote the proposal; Tried built-in modules from sklearn such as Naive Bayes, SVM, SGDClassifier; Revised documentation; Presentation.