Assignment 2

DDA4220 Assignment Report

Xiangyi Li (119010153)
School of Data Science
Chinese University of Hong Kong, Shenzhen
xiangyi.li@cuhk.edu.cn

1 Introduction

The problem is a binary text classification task where the model needs to predict if a tweet is about a real disaster or not.

1. Baseline method: With the GloVe model I am able to tokenize the data loaded from than train and test CSV files through glove.stoi API. Then with the provided baseline RNN class that extends the torch nn module, I am able to call model.train and model.eval for the splitted train and validation set. Finally I exported the result with a timestamp as submission.csv.
2. RoBERTa method: Used the RobertaTokenizer and the RobertaForSequenceClassification over the baseline model.

2 Approach

Baseline model: The RNN mmodel processes input sequences by iterating over the elements and maintaining an internal state that captures information about the history of processed elements. **Improved model:** RoBERTa (Robustly Optimized BERT Pretraining Approach) is a variant of BERT, a transformer-based language model, developed by researchers at Facebook AI. It uses self-attention to process input sequences and generate contextualized representations of words in a sentence.

3 Experiments

3.1 Data

I implemented three data preprocessing pipelines, including turning all words to lowercase, remove URLs, remove special characters. I also tried enabling different combination of those preprocessing pipelines and it turned out that lowercase has the most significant impact on the validation accuracy.

3.2 Experimental details

I implemented a manual vocabulary building function when first encourtering bugs when using GloVE. I also tried spaCy and basic_english tokenizer in the baseline mode since I noticed some foreign languages in the dataset. I also tried modifying some of the hyperparameters to the baseline model but it turned out to have little impact on the final validation performance after enough epochs. The model stops to improve in terms of validation accuracy with default configuration after 5-10 epochs, which is converging quickly given the baseline model trains rather fast. In the RoBERTa scenario the model reaches somewhat the full capacity under my initial configurations at around the 2nd epoch.

3.3 Results and Analysis

Baseline model got a 0.72234 score on Kaggle, while RoBERTa gets a 0.83113 score.

```
Epoch 1/20: train_loss=0.6848, val_loss=44.7483, val_acc=0.5530
Epoch 2/20: train_loss=0.6832, val_loss=44.6181, val_acc=0.5543
Epoch 3/20: train_loss=0.6747, val_loss=35.6731, val_acc=0.6427
Epoch 4/20: train_loss=0.6346, val_loss=29.9440, val_acc=0.7005
Epoch 5/20: train_loss=0.5830, val_loss=29.0562, val_acc=0.7093
Epoch 6/20: train_loss=0.5495, val_loss=26.6256, val_acc=0.7338
Epoch 7/20: train_loss=0.5226, val_loss=25.9549, val_acc=0.7404
Epoch 8/20: train_loss=0.5072, val_loss=25.0000, val_acc=0.7500
Epoch 9/20: train_loss=0.4937, val_loss=24.7435, val_acc=0.7522
Epoch 10/20: train_loss=0.4730, val_loss=23.4138, val_acc=0.7662
Epoch 11/20: train_loss=0.4619, val_loss=23.3349, val_acc=0.7662
Epoch 12/20: train_loss=0.4589, val_loss=23.2678, val_acc=0.7671
Epoch 13/20: train_loss=0.4441, val_loss=22.5339, val_acc=0.7741
Epoch 14/20: train_loss=0.4301, val_loss=22.3801, val_acc=0.7758
Epoch 15/20: train_loss=0.4250, val_loss=22.4708, val_acc=0.7745
Epoch 16/20: train_loss=0.4247, val_loss=22.3603, val_acc=0.7758
Epoch 17/20: train_loss=0.4101, val_loss=21.7527, val_acc=0.7820
Epoch 18/20: train_loss=0.4082, val_loss=22.5103, val_acc=0.7745
Epoch 19/20: train_loss=0.3991, val_loss=22.0131, val_acc=0.7793
Epoch 20/20: train_loss=0.3944, val_loss=21.9223, val_acc=0.7806
```

Figure 1: Baseline Training

```
Epoch 1/4: 100%|██████████| 334/334 [01:55<00:00,  2.90it/s]
Validation Epoch 1/4: 100%|██████████| 143/143 [00:16<00:00,  8.73it/s]

Epoch 1/4 - Training Loss: 0.4827 - Validation Loss: 0.4415 - Validation Accuracy: 0.8192

Epoch 2/4: 100%|██████████| 334/334 [01:59<00:00,  2.79it/s]
Validation Epoch 2/4: 100%|██████████| 143/143 [00:16<00:00,  8.65it/s]

Epoch 2/4 - Training Loss: 0.3607 - Validation Loss: 0.3642 - Validation Accuracy: 0.8538

Epoch 3/4: 100%|██████████| 334/334 [01:59<00:00,  2.79it/s]
Validation Epoch 3/4: 100%|██████████| 143/143 [00:16<00:00,  8.70it/s]

Epoch 3/4 - Training Loss: 0.3071 - Validation Loss: 0.4001 - Validation Accuracy: 0.8485

Epoch 4/4: 100%|██████████| 334/334 [01:59<00:00,  2.79it/s]
Validation Epoch 4/4: 100%|██████████| 143/143 [00:16<00:00,  8.67it/s]
Epoch 4/4 - Training Loss: 0.2768 - Validation Loss: 0.4203 - Validation Accuracy: 0.8507
```

Figure 2: RoBERTa Training

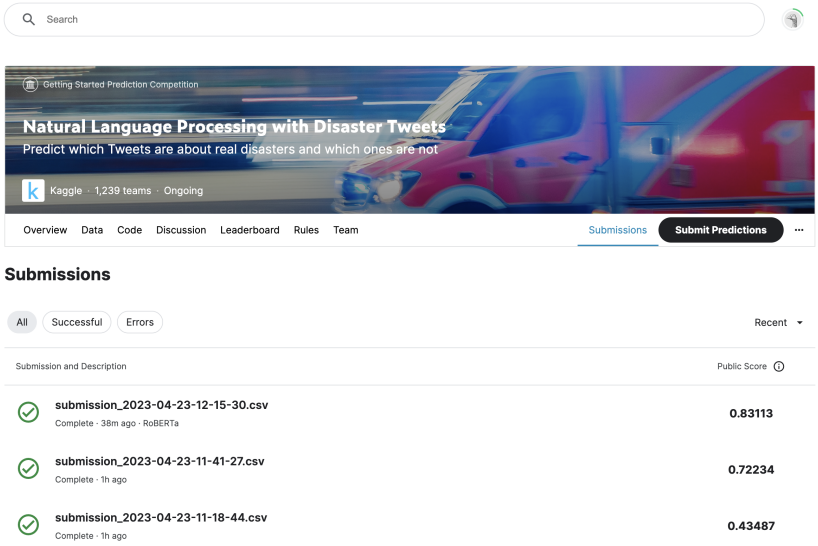

Figure 3: Kaggle Result

There are also some interesting findings during my experimentation with the baseline model and the RoBERTa model. When building the vocab for the baseline model I found out that some vocabs were missing from the GloVE vocab builder, which became one of the reasons why I turned to pre-trained models like RoBERTa since it has a way larger vocab. I also chose RoBERTa because it optimizes the BERT model which uses the Transformer architecture and the attention mechanism that made the model able to remember longer sequence. It's one of the drawbacks of RNN and its variants like LSTM. However, the RoBERTa is not as performant as the RNN baseline and an online BERT example when training.

I initially attempted to train the model locally, but I soon realized how slow the training process can get and migrated the execution to Colab where I could utilize CUDA GPU accelerations. It boosted training speed hugely.