Work summary for the 3^{rd} week

Student: Xuanyu Su
Supervisor: Isar Nejadgholi

September 21, 2020

1 Modifications and Improvements of Assignment 1

1.1 Questions remain in A1.version1

Q1. The final result for our model is not good enough, especially, the F1 score for positive class(comments labeled with attack).
Q2. In the previous version of our model, we set 5 epochs, but the final ROC curve did not show all the data points.

1.2 Solutions for Q1

1. In the data preprocessing stage, no detailed analysis of the data was performed, resulting in the mistaken deletion of useful data. In the 'comment' table, there are duplicates in the 'rev-id' column. Although these comments have the same id, they correspond to different authors, and according to different authors, whether the comment is attack or not has different results, since people's comments are subjective, we cannot draw conclusions based on a single comment label. So here, we need to use voting to filter these duplicate comments, and finally take the mean value. When the mean value is greater than 0.5, we can label this comment as an attack. Related codes shows in the below:

```
new_label = result.groupby('rev_id')['attack'].mean() > 0.5
```

Figure 1: Get mean value of duplicate comments

2. Also, in the data preprocessing stage, we need to remove the punctuations in the original comment to reduce the impact on the model results. At the same time, all uppercase letters in the sentence need to be converted to lowercase, which may ensure more accurate analysis and recognition of our model.Related codes shows in the below:

```
import re
result['comment'] = result['comment'].apply(lambda x:re.sub(r'[^A-Za-z0-9 ]+', ' ', x).lower())
```

Figure 2: Convert all uppercase letters to lowercase letters

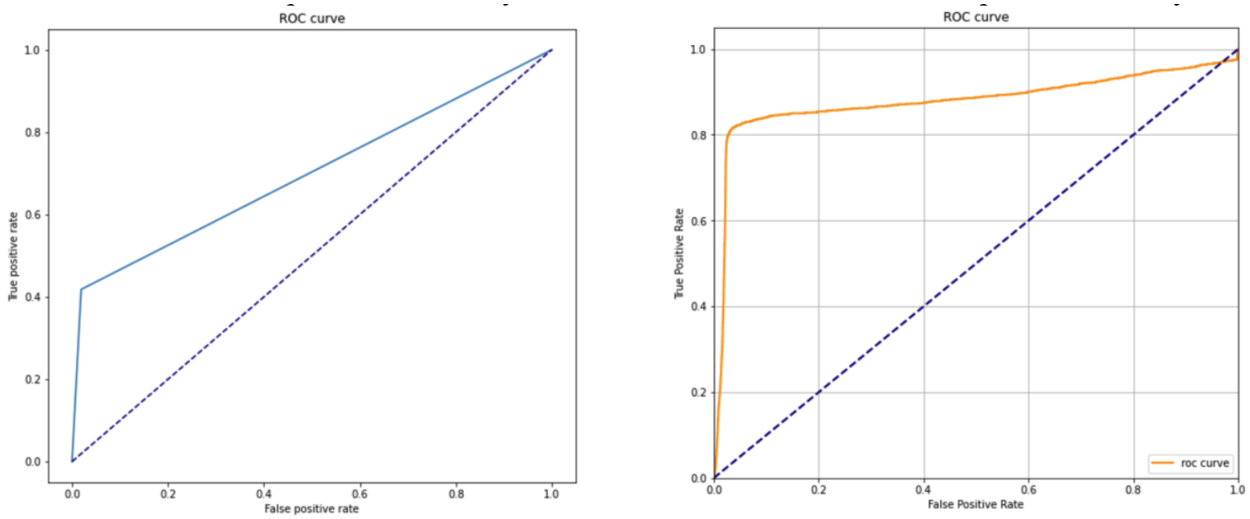
3. In the training phase of our model, our basic parameters, such as epoch and bias, should be defined and frozen before training, so that these data will not change as the model participates in prediction, thereby affecting the results of the experiment. In response to this problem, we have a more concise and efficient fine-tuning model for pytorch. Related codes shows in the below:

```
from tqdm import tqdm
symbol_list = ['!','@','#','$','%','^','&','*','(',')','-','+','?','>','<','=','/',"`",'"','.',':',';','  ','   ','
for i in tqdm(symbol_list):
    result['comment'] = result['comment'].apply(lambda x: x.replace(i, ' '))
```

Figure 3: Remove all punctuation

1.3 Solutions for Q2

In our 1st version of the experiment, the ROC curve was displayed incorrectly, and the input type of the ROC curve should be a **probability**. Therefore, after the model makes predictions on the test data, we get the probability values of both the positive class(attacked comments) and the negative class(non-attacked comments), and only keep the probability value of the positive class (attacked-comments), and then put positive probability data into the ROC function for processing, thus got the final ROC curve. As we can see from the below modified results, the curve becomes much smoother



ROC curve before modify ROC curve after modify

Figure 4: The comparison of ROC curve before and after modify

1.4 Updated Results

Through the display of the loss curve and the F1 score in Classification Report below and the comparison with our 1st version, we can find that the model has a higher accuracy on the train data after the above modifications. We can intuitively see from Figure 6. , The loss of train data shows a clear downward trend and less fluctuations. At the same time, it can be seen from the Figure 5 that the F1 score for attack data has been significantly improved. The final average has also been dramatically improved. In summary, the measures we have taken in this version are:

- Preprocess the data more carefully;
- Use weight decay to reduce over-fitting and improve the accuracy of the model.

The above two major changes have had a very obvious impact on the final result of our model.

Classification Report:

	precision	recall	f1-score	support
1	0.8150	0.4179	0.5525	3869
0	0.8936	0.9810	0.9352	19275
accuracy			0.8868	23144
macro avg	0.8543	0.6994	0.7439	23144
weighted avg	0.8804	0.8868	0.8713	23144

Classification report before modify

Classification Report:

	precision	recall	f1-score	support
1	0.8453	0.7790	0.8108	2756
0	0.9705	0.9808	0.9756	20422
accuracy			0.9568	23178
macro avg	0.9079	0.8799	0.8932	23178
weighted avg	0.9556	0.9568	0.9560	23178

Classification report after modify

Figure 5: The comparison of Classification Report curve before and after modify

2 Fine-Tuning model use

By observing Figure 7, we can understand the structure of the transformer more clearly. In this experiment, there are a total of 12 encoders. Let's take the first layer as an example:

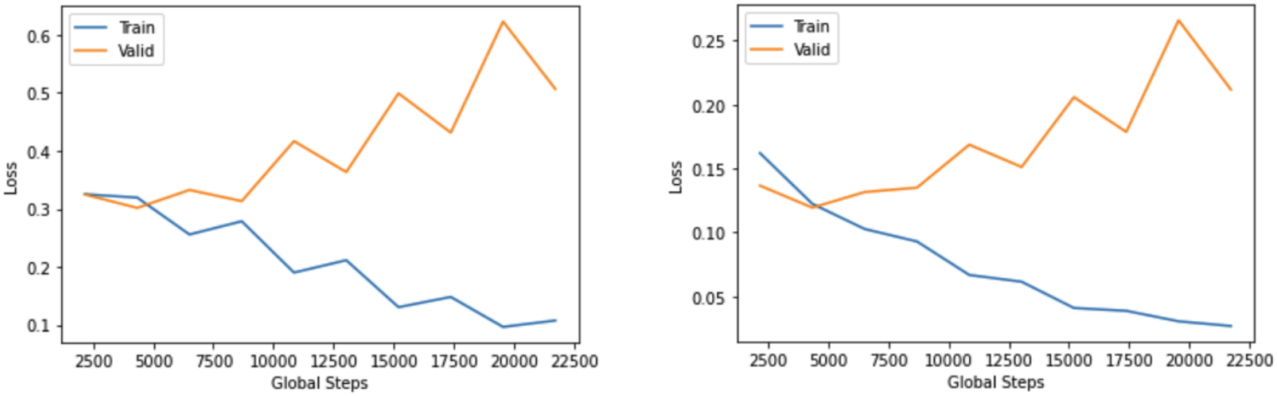


Figure 6: The comparison of Loss curve before and after modify

It can be known from the code in Figure 8 that the model performs a decay operation of **0.01** for all parameters except **bias and LayerNorm.weight**. And through the results of **part 1.4**, it can be seen that the decay operation has significantly improved the model results.

```
name encoder.bert.embeddings.LayerNorm.weight
name encoder.bert.embeddings.LayerNorm.bias
name encoder.bert.encoder.layer.0.attention.self.query.weight
name encoder.bert.encoder.layer.0.attention.self.query.bias
name encoder.bert.encoder.layer.0.attention.self.key.weight
name encoder.bert.encoder.layer.0.attention.self.key.bias
name encoder.bert.encoder.layer.0.attention.self.value.weight
name encoder.bert.encoder.layer.0.attention.self.value.bias
name encoder.bert.encoder.layer.0.attention.output.dense.weight
name encoder.bert.encoder.layer.0.attention.output.dense.bias
name encoder.bert.encoder.layer.0.attention.output.LayerNorm.weight
name encoder.bert.encoder.layer.0.attention.output.LayerNorm.bias
name encoder.bert.encoder.layer.0.intermediate.dense.weight
name encoder.bert.encoder.layer.0.intermediate.dense.bias
name encoder.bert.encoder.layer.0.output.dense.weight
name encoder.bert.encoder.layer.0.output.dense.bias
name encoder.bert.encoder.layer.0.output.LayerNorm.weight
name encoder.bert.encoder.layer.0.output.LayerNorm.bias
name encoder.bert.encoder.layer.1.attention.self.query.weight
```

Figure 7: The inner structure of our model

```python
# weight decay

from transformers import AdamW
model = BERT().to(device)
no_decay = ['bias', 'LayerNorm.weight']
optimizer_grouped_parameters = [
{'params': [p for n,p in model.named_parameters() if not any(nd in n for nd in no_decay)], 'weight_decay': 0.01},
{'params': [p for n,p in model.named_parameters() if any(nd in n for nd in no_decay)], 'weight_decay': 0.0}
]
optimizer = AdamW(optimizer_grouped_parameters, lr=1e-5)
train(model=model, optimizer=optimizer)
```

Figure 8: Parameters decay

3 Paper Reading and Data

3.1 The main content of the paper

This paper can be divided into three parts:

1. Introduction to the background and preliminary classification of the existing situation.

2. The source and screening of the database and data preprocessing, etc., and then put the processed data into different models for training, and finally select the BERT model as the final prediction one by comparing the final results.

   Among them, in the data screening and processing stage, because there are already quite a lot of databases available on the Internet, but because the data labels are different from the requirements of this paper, the author obtained the **COVID-HATE** database after summarizing, Nevertheless, there are still about 2,400 data that need to be manually classified. The author distinguishes these data according to the following three categories:

   - **Anti-Asians COVID-19 Hate Tweets**: contains one or more COVID-19 keywords. directed towards to Asian people or assigns blame for the creation, spread;
   - **Pro-Asians COVID-19 Counterhate Tweets**: Contains vocabulary that actively oppse racism or support and defend of Asian people;
   - **Hate Neutral Tweets**: tweets the are neither implicitly nor explicitly hateful, nor counterhateful.

   After the classification and preprocessing of the data, the author used the **GloVe** model and the **BERT** model to train the data. The results show that: after analyzing and predicting the three types of data, the BERT model has excellent performance in the two categories. So BERT becomes the model to be used in the future.

3. Evaluate the results of the model and analyze the reasons for the impact of these hate words, and at the same time conduct a deep analysis of the spread and impact of these hate words. Then the paper analyzes the reasons for this kind of hateful or offensive tweets from a deeper level: starting from the time and frequency of hate words, it integrates the nationality of the authors, user activity and interactions, and the activity of these users. Finally, they analyzed the structural characteristics of social networks and came to the following conclusions:

   (a) These hateful or racial tweets will cause different levels of mental harm to the Asian people;

   (b) But these current studies also have limitations:

   - Only limited to tweets in English;
   - Only for anti-asian analysis, there may be attacks and harm to other groups;
   - Only limited to text-based attacks, but it also provides more reference value for future research, such as image or video analysis.

   This experiment laid the foundation and played a huge reference role for the future work.

3.2 Data Analyse

After reading the paper, we had downloaded the original dataset related with is: the COVID-HATE dataset, it consists of two part:

1. **All hate, counterhate, and neutral COVID-19 tweets**: This dataset contains **30,929,269** COVID-19 tweets. All tweets are classified into **hate, counterhate**, and **neutra**l categories:

   - **Hate**: 891,204 tweets. These tweets are classified as hateful by the classifier.
   - **Counterhate**: 200,198 tweets. These tweets are classified as counterhate.
   - **Neutral**: 26,837,429 tweets. These tweets are classified as neutral.
   - **Other**: 3,000,439 tweets. These tweets were classified into more than one of the above categories.

2. **Annotated hate, counterhate, and neutral tweets**: This is a manually-annotated dataset of **2,319** COVID-19 related racial hate tweets categorized into four categories:

   - **Hate**: 678 tweets. These tweets are directed towards Asians, and are abusive, derogatory, or assign blame for COVID-19.
   - **Counterhate**: 359 tweets. These tweets oppose racist and hateful content towards Asians, or support and defend Asians.
   - **Neutra**l: 961 tweets. These tweets are neither hateful or counterhateful.
   - **Non-Asian Aggression**: 321 tweets. These tweets are hateful, abusive, or aggressive towards non-Asian targets.

Property	Statistic
Duration	Jan 15–Apr 17, 2020
Number of tweets	30,929,269
Number of (frac.) hateful tweets	891,204 (2.88%)
Number of (frac.) counterhate tweets	200,198 (0.65%)
Number of (frac.) neutral tweets	26,837,429 (86.77%)
Number of users	7,833,194
Number of (frac.) hateful users	393,897 (5.03%)
Number of (frac.) counterhate users	136,154 (1.74%)
Number of (frac.) neutral users	6,812,695 (86.97%)
Number of nodes in the network	87,851,137
Number of edges in the network	717,087,317

Figure 9: The dataset statistics

The overall dataset statistics is as follow: From the data, we can observe that due to user privacy constraints, we can only access tweets ID and publication Time. To get specific content, we need to use Twitter's API. We have submitted an application last week and we are waiting for verification. After that, the subsequent data processing is performed.