Work summary for the 6th week

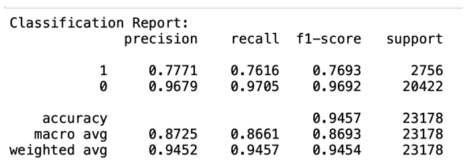
Student: Xuanyu Su
Supervisor: Isar Nejadgholi

October 14, 2020

1 More data preprocessing on the Trainer model training

In our work last week, we tried to use trainer() to replace model building blocks. Given that the results of our last model were not very good, we have performed more data preprocessing on the last model this week. We mainly removed the special characters in text like 'newline token', which may interfere with the results predicted by the model. The improved results are as follows:

1. Result on Attack Comment data: (figure 1)

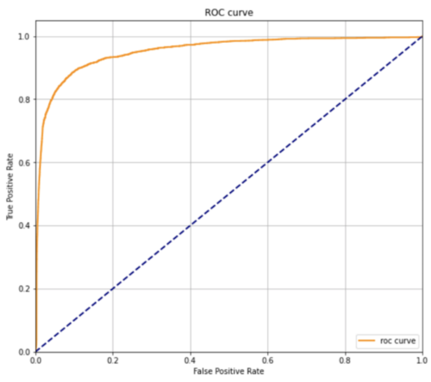


```
Classification Report:
              precision    recall  f1-score   support

           1     0.7771    0.7616    0.7693      2756
           0     0.9679    0.9705    0.9692     20422

    accuracy                         0.9457     23178
   macro avg     0.8725    0.8661    0.8693     23178
weighted avg     0.9452    0.9457    0.9454     23178
```

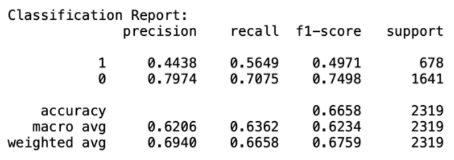
The classification report of trainer method test on Attack Comment data



The Roc curve of trainer method test on Attack Comment data

Figure 1: The classification report and ROC curve of trainer method test on Attack Comment data

2. Result on 2400 hand labeled data: (figure 2)

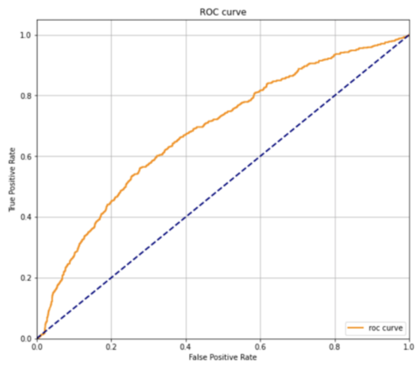


```
Classification Report:
              precision    recall  f1-score   support

           1     0.4438    0.5649    0.4971       678
           0     0.7974    0.7075    0.7498      1641

    accuracy                         0.6658      2319
   macro avg     0.6206    0.6362    0.6234      2319
weighted avg     0.6940    0.6658    0.6759      2319
```

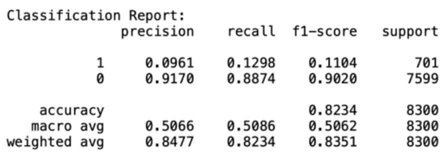
The classification report of trainer method test on 2400 hand labeled data



The Roc curve of trainer method test on 2400 hand labeled data

Figure 2: The classification report and ROC curve of trainer method test on 2400 hand labeled data

3. Result on Gab data: (figure 3)

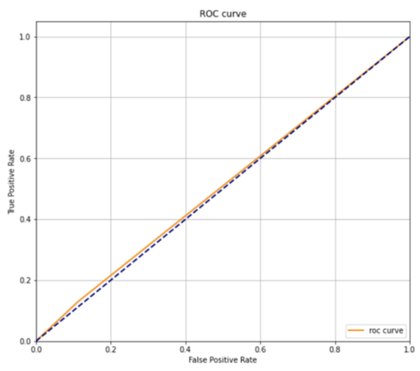


```
Classification Report:
              precision    recall  f1-score   support

           1     0.0961    0.1298    0.1104       701
           0     0.9170    0.8874    0.9020      7599

    accuracy                         0.8234      8300
   macro avg     0.5066    0.5086    0.5062      8300
weighted avg     0.8477    0.8234    0.8351      8300
```

The classification report of trainer method test on Gab data



The Roc curve of trainer method test on Gab data

Figure 3: The classification report and ROC curve of trainer method test on gab data

1.1 Summary

1. The F1 score of Attack Comment has been increase from: **0.0213 for positive class and 0.9368 for negative class** to **0.7692 for positive class and 0.9692 for negative class**, and the weight avg also increase from **0.8280 to 0.9454**.
2. The F1 score of 2400 hand labeled data has been increase from: **0.0170 for positive class and 0.8238 for negative class** to **0.4971 for positive class and 0.7498 for negative class**, and the weight avg also increase from **0.5879 to 0.6759**.
3. The comparison shows adequate data preprocessing can improve the performance of the model.

2 ROC-AUC, and F1 score

Before we start this module, let's briefly introduce what is TP, TN, FP, FN, shown in the figure 4:

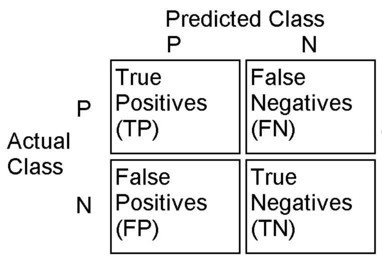


Figure 4: Structure of the confusion matrix with TP, FN, FP and TN values.

2.1 Introduction

1. **ROC**: Receiver Operating Characteristics, AUC - ROC curve is a performance measurement for classification problem at various thresholds settings. ROC is a probability curve. The ROC curve is plotted with **TPR** against the **FPR** where TPR is on y-axis and FPR is on the x-axis.

$$TPR/Recall/Sensitivity = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

2. **AUC**: Area Under The Curve, AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s.

3. **F1 Score**: combines Recall and Precision to one performance metric. F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. F1 is usually more useful than Accuracy, especially if you have an uneven class distribution.

$$F1 = 2 * \frac{Recall * Precision}{Recall + Precision}$$

2.2 How to speculate the performance of a model?

- An excellent model has AUC near to the **1** which means it has good measure of separability.
- A poor model has AUC near to the 0 which means it has worst measure of separability. In fact it means it is reciprocating the result. It is predicting 0s as 1s and 1s as 0s.
- And when AUC is 0.5, it means model has no class separation capacity whatsoever.

2.3 The best result of 'Attack Comment ' model under different ROC thresholds (to be finished)

3 Gab model

In this module, we mainly use the data obtained from the **gab.ai** website for model training. Through continuous modification and debugging of the model, we finally got the following two versions, and compared their results.The data source is from **July 18 to March 20**. Since the database is manually labeled by different people, we need to do the following operations in the data preprocessing stage.

3.1 Model Version 1.0

3.1.1 Data preprocessing

1. **Voting the duplicate texts**: since in this database, the data is manually labeled, and each text was labeled by at least three people. Therefore, there is a problem of data duplication. Here we continue to use the '**voting**' method of the same as previous model to average the label of same data and set the threshold to **0.5**. When the average is greater than **0.5**, all the duplicate text will be relabeled as positive 'Hate', then after we remove duplicates text.
2. **Remove emoji:** in our previous work we keep emojis and convert them into corresponding words to make the model to get better results. But this method does not seem to be suitable for our new dataset, since in this 'Gab' data, most of the emojis have nothing to do with the content of the text after observing the data set, but will increase the interference, hence we remove all the emojis.
3. **Remove URL**: remove all the url links;
4. **Remove punctuation**: remove all the punctuation, like: ',' , '.', '!', etc.
5. **Remove usename**: remove the usernames that have been mentioned in the texts;
6. **Remove Hashtag**: remove the hashtags mentioned in the texts;
7. **Transform uppercase to lowercase**: transform all the uppercase into lowercase.

3.1.2 Model used and result

In this version of the model construction, we still use the same model framework as the original '**Attack Comment**': After preprocessing the data, we divided the data ratio of train, validation and test into **8:1:1**, still use the **fields** and **TabularDataset** methods to do data embedding process, and then imported data into the **bert-base-uncased** model, set the criterion to **BCELoss()**, the optimizer to **AdamW**, and **weight-decay** is used to prevent overfitting. The loss curve of the model is as figure 5:

The final test results on the 'Gab' data and '2400 hand labeled' data are as figure 6 and figure 7:

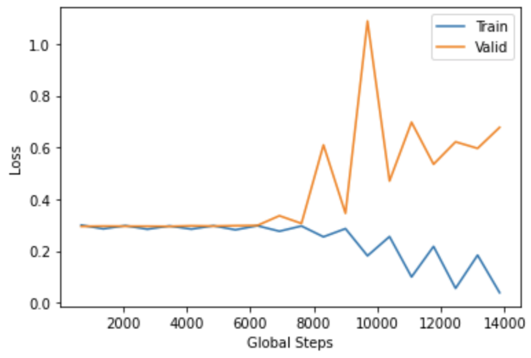
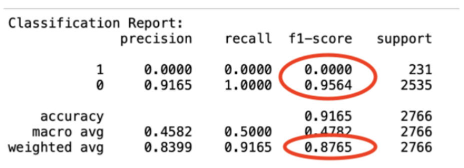


Figure 5: The train and validation loss of Gab model.

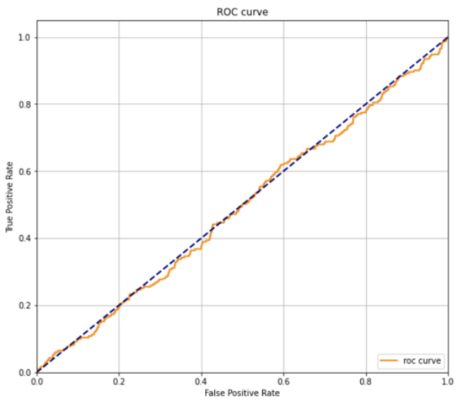


```
Classification Report:
              precision    recall  f1-score   support

           1     0.0000    0.0000    0.0000       231
           0     0.9165    1.0000    0.9564      2535

    accuracy                         0.9165      2766
   macro avg     0.4582    0.5000    0.4782      2766
weighted avg     0.8399    0.9165    0.8765      2766
```

The classification report of Gab data



The Roc curve of Gab data

Figure 6: The classification report and ROC curve of model on Gab data

3.1.3 Summary

Through the observation of the above two figures, it is not difficult to find that the F1 score of the positive class of the model on the 'Gab' and 2400 data tests is **zero**. Although the F1 score of the negative class is higher, by combining the ROC curve and the related knowledge we introduced in module2, the curve almost coincides with the diagonal, indicating that the prediction effect of the model is not good at all.
After we got the conclusion above, we started doing some modifications to improve the performance of our model.

3.2 Model Version 2.0

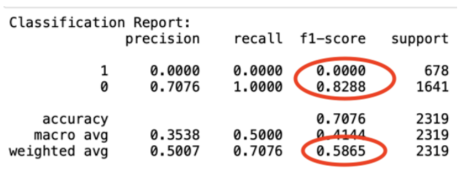
In this version of the model preprocessing, we made the following improvements on the basis of version 1.

3.2.1 Data preprocessing

1. **Stemming and Lemmatization**: Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. After observing the data, we found that words are converted in different forms according to the different tense and subjects, but it may increase the difficulty on processing for the model. Therefore, in this version, we performed stemming operations on the text to convert words into root forms, such as: 'I'm' into 'I am', 'he's' into 'he is', etc.

2. **Change the order of data preprocessing**: In the previous version, we did not consider the influence of the order of those data preprocessing operations on the experimental results. However, it can be found from the **hashtag** and **emoji** that if punctuation were removed firstly, it will destroy the structure of the hashtags and emojis, so that they cannot be removed successfully in the future Therefore, in this version of the experiment, we first removed the hashtag, and then identified all emojis and marked them as 'unknown token'. Finally, punctuations are removed.

3. **Add emoji tokens**: Do not remove emojis, and do not convert emojis into corresponding words either. Instead, store emojis with the help of a dictionary, and manually add a new token for them, then mark all emojis that

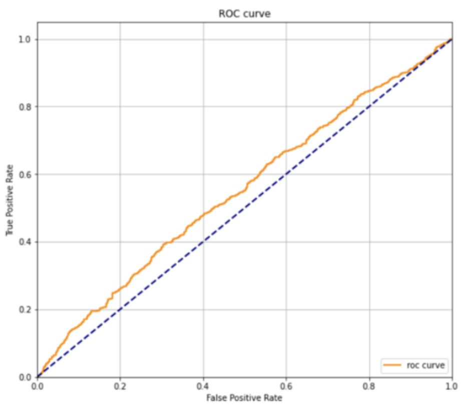


```
Classification Report:
              precision    recall  f1-score   support

           1     0.0000    0.0000    0.0000       678
           0     0.7076    1.0000    0.8288      1641

    accuracy                         0.7076      2319
   macro avg     0.3538    0.5000    0.4144      2319
weighted avg     0.5007    0.7076    0.5865      2319
```

The classification report of 2400 hand labeled data



The Roc curve of 2400 hand labeled data

Figure 7: The classification report and ROC curve of model on 2400 hand labeled data

appear in the text as this token type to ensure that emoji can also participate model training. (but in this version, after we transfer the emoji into tokens, it takes much longer time on embedding stage than before, so we currently only keep them, but thet do not participate in the model training).

4. **Racism word. csv**: since we hope to focus our model's data mainly on **racial discrimination** (because this database involves 11 different kinds of language attacks, which is too many for our model training). Therefore, we manually created the '**racism word. csv**' file, which contains discriminatory words for people from different countries and regions. By using this dictionary, we can more accurately filter out more relevant data from the 'Attack Comment' database, thereby effectively expanding our 'Gab' data. (Data Source: `https://en.wikipedia.org/wiki/List_of_ethnic_slurs`)

5. **Train data expansion**: we found that the ratio of positive and negative data in the original database is: **0.08: 0.92**, which is a manifestation of **unbalanced** data types. Therefore, in order to eliminate the impact of uneven data distribution on the results, we merged the data that had been marked with 'Hate' in 'Attack Comment' dataset with the 'Gab; database. The proportion of positive data after expansion is 0.23. The data is divided as shown below figure 8:

Note: Because we only want to try to improve the accuracy of the model by increasing the amount of **positive data**, under the premise that the test data is the same. Therefore, before performing the data expansion operation, we first divide the 'Gab' data into **train** and **test**, and then expand the data **only on the train data set**.

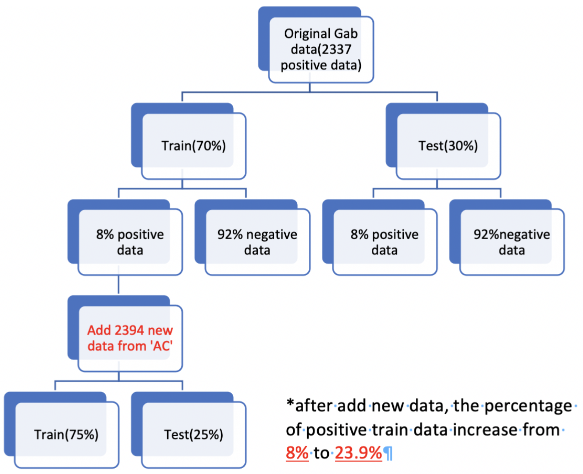

Original Gab data(2337 positive data)
Train(70%)
Test(30%)
8% positive data
92% negative data
8% positive data
92%negative data
Add 2394 new data from 'AC'
Train(75%)
Test(25%)
*after add new data, the percentage of positive train data increase from 8% to 23.9%

Figure 8: The data division flow chart

3.2.2 Model used and result

In this section, we used the same model and model parameters as in the Model Version 1.0. And the loss curve of the model shows in the below:

1. The loss curve of Gab model 2.0: (figure 9)

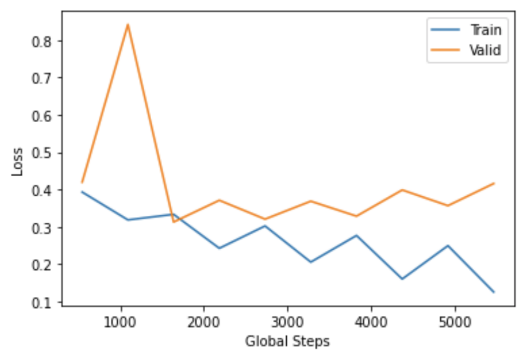
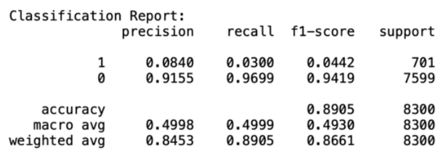


Figure 9: The loss curve of Gab model 2.0

2. Result of Gab model 2.0 on test data set: (figure 10)

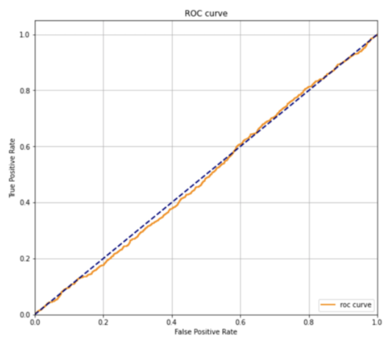


```
Classification Report:
              precision    recall  f1-score   support

           1     0.0840    0.0300    0.0442       701
           0     0.9155    0.9699    0.9419      7599

    accuracy                         0.8905      8300
   macro avg     0.4998    0.4999    0.4930      8300
weighted avg     0.8453    0.8905    0.8661      8300
```

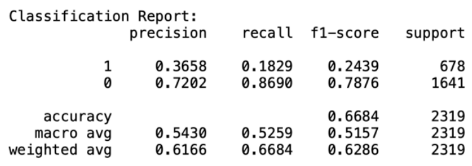
The classification report of Gab model2.0 on test dataset



The Roc curve of Gab model2.0 on test dataset

Figure 10: The classification report and ROC curve of Gab model 2.0 on the test dataset

3. Result of Gab model 2.0 test on 2400 hand labeled data: (figure 11)

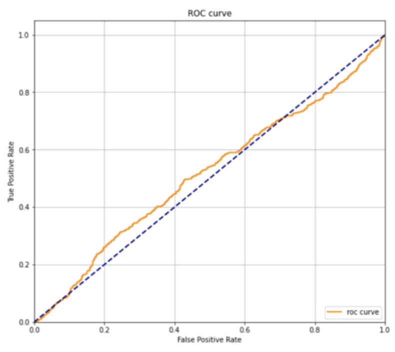


```
Classification Report:
              precision    recall  f1-score   support

           1     0.3658    0.1829    0.2439       678
           0     0.7202    0.8690    0.7876      1641

    accuracy                         0.6684      2319
   macro avg     0.5430    0.5259    0.5157      2319
weighted avg     0.6166    0.6684    0.6286      2319
```

The classification of Gab model 2.0 test on 2400 hand labeled data

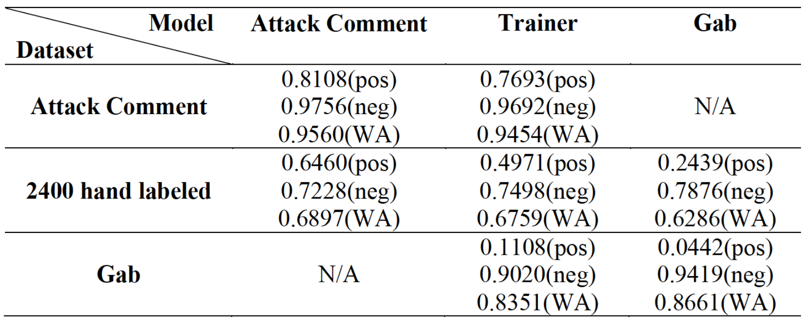


The Roc curve of Gab model2.0 test on 2400 hand labeled data

Figure 11: The classification report and ROC curve of Gab model 2.0 on 2400 hand labeled data

3.3 Compare and summary

Through the learning and continuous improvement of the three models(Attack Comment model, Trainer method and Gab model) so far, we compare the performance of the three models on different datasets (Wiki dataset also know as Attack Comment, Anti-Asian 2400 hand labeled and Gab dataset) through the following table.



Model Dataset	Attack Comment	Trainer	Gab

Attack Comment	0.8108(pos)
	0.9756(neg)
	0.9560(WA)
		0.7693(pos)
		0.9692(neg)
		0.9454(WA)
			N/A

2400 hand labeled	0.6460(pos)
	0.7228(neg)
	0.6897(WA)
		0.4971(pos)
		0.7498(neg)
		0.6759(WA)
			0.2439(pos)
			0.7876(neg)
			0.6286(WA)

Gab	N/A	0.1108(pos)
		0.9020(neg)
		0.8351(WA)
			0.0442(pos)
			0.9419(neg)
			0.8661(WA)

Figure 12: The performance of three models on different datasets

Note: pos for positive, neg for negative and WA means weighted average.
Through the observation of the above table, we found that after two modifications and adjustments to the Gab model, the accuracy of the model's positive prediction increased from 0 to 0.0442, and the fluctuations in loss curve become less,(in version 1.0, after several fluctuations, this loss curve of valid begins to show a sharp upward trend. But in model version 2.0, this trend has been significantly reduced, although the improvement effect is not very significant, it confirms our conjecture: the performance and accuracy of the model are directly related to the data. Therefore, we hope to achieve a better result by more detailed data preprocessing. Our two experiments are based on the **bert-base model**, so we are trying to find other models that can perform better than Bert in the future.

4 Future work

4.1 Model Version 3.0 (ongoing)

In this section, we will make the following improvements on the basis of **data expansion** module in version 2.

4.1.1 Data preprocessing

- **Add and use emoji tokens**: Do not remove emojis, and do not convert emojis into corresponding words either. Instead, store emojis with the help of a dictionary, and manually add a new token for them, then mark all emojis that appear in the text as this token type to ensure that emoji can also participate model training.

4.1.2 Model used and result

In this version of experiment, we will use **RoBERTa** as our training model, as RoBERTa is developed based on BERT, they share lots of configs. RoBERTa performs better than BERT by applying the following adjustments:

- Bigger training data (16G vs 161G);
- Using dynamic masking pattern (BERT use static masking pattern);
- Replacing the next sentence prediction training objective;
- Training on longer sequences.