CSE - 598 Intro to Deep Learning Project Report

Project Report by - Anshita Singh Bais (1222426855)

Task - Irony Detection in English Tweets

NOTE - Please refer to the git repository for easy access of all the files
Link - https://github.com/AnshitaSinghBais/Deep-Learning-Project

Reference of the Research paper

Unified Benchmark and Comparative Evaluation for Tweet Classification by Cynthia Van Hee, Els Lefever and Ve´ronique Hoste LT3 Language and Translation Technology Team Ghent University Groot-Brittannie¨laan 45, 9000 Ghent

Link for the research paper - https://arxiv.org/pdf/2010.12421.pdf

Definition of the task -

Given a tweet, predict whether the tweet includes irony intents or not. The dataset consists of 2 fields - the label and the text. The label is either 0 or 1 denoting not irony and irony respectively and the text consists of the tweets with emojis converted to UTF-8 encoding. The input of the dataset will be the tweet and the output will be either 1 or 0 based on whether the tweet has irony or not.

Label	Definition
1	Irony
0	Not Irony

Dataset Description

Used three types of dataset -

1. SemEval2018-T3-train-taskA.txt and SemEval2018-T3-train-taskB.txt

2. SemEval2018-T3-train-taskA_emoji.txt and SemEval2018-T3-train-taskB_emoji.txt
3. SemEval2018-T3-train-taskA_emoji_ironyHashtags.txt and SemEval2018-T3-train-taskB_emoji_ironyHashtags.txt

The difference between the three datasets used is that the first dataset has no emojis and hashtags. The second dataset has only emojis and no hashtags. The third dataset has emojis and hashtags. The datasets 1 has emojis and hashtags being replaced with their UTF-8 description. And the same has been implemented with hashtags in dataset 2 with hashtags.

Link of the dataset - https://github.com/Cyvhee/SemEval2018-Task3/tree/master/datasets/train

The format of the dataset has been changed from txt to CSV to make it more code friendly.

Example of dataset 1 -

Label	**Text**
1	Planned on an early night last night. Oh yaa course that happened..
0	2014 can’t end fast enough! Stupid dead battery.
1	Doesn’t lucky and fortunate mean the same thing?

Example of Dataset 2 -

Label	**Text**
1	Love these cold winter mornings 😬 best feeling everrrrrrr !
0	You're never too old for Footie Pajamas. http://t.co/ElzGqsX2yQ
1	Yay for another work at 4am day 😐

Example of Dataset 3 -

Label	**Text**
1	Sweet United Nations video. Just in time for Christmas. #imagine #NoReligion #irony http://t.co/fej2v3OUBR
0	The rain has made extra extra lazy😅
1	Luv this #not

Baseline

The model was trained using Naive Bayes and the accuracy received in classifying the labels was 64.53%. The label 0 represents that the tweet wasn't ironic, while the value 1 represents otherwise.

Evaluation Matrix

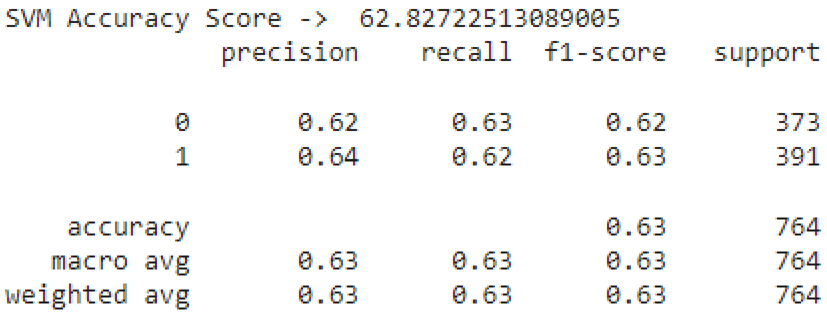
The evaluation matrix that is used is the classification report that gives an insight to the recall rate, f1-score and precision of the model.

Experiments and Results

Preprocessing of the data - Removed all the stop-words.

1. SVM MODEL -

   Used SVM model to get trained on the dataset 1. The train data was split into 80% train data and 20% test data. The model gave an accuracy of 62.82%.



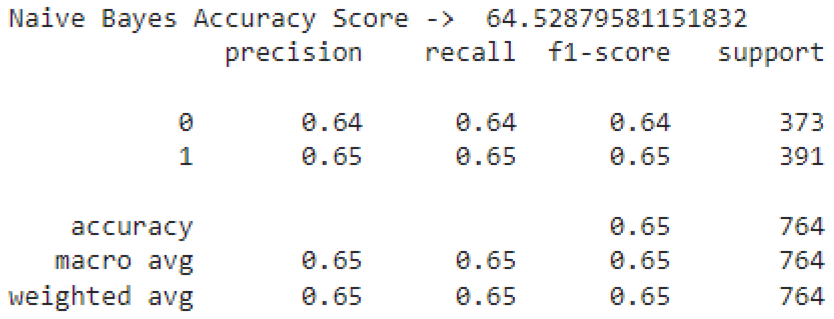
```
SVM Accuracy Score ->  62.82722513089005
              precision    recall  f1-score   support

           0       0.62      0.63      0.62       373
           1       0.64      0.62      0.63       391

    accuracy                           0.63       764
   macro avg       0.63      0.63      0.63       764
weighted avg       0.63      0.63      0.63       764
```

2. Naive Bayes Model -

   Used the Naive Bayes model to get trained on dataset 1. The train data was split into 80% train data and 20% test data. The model gave an accuracy of 64.52%.



```
Naive Bayes Accuracy Score ->  64.52879581151832
              precision    recall  f1-score   support

           0       0.64      0.64      0.64       373
           1       0.65      0.65      0.65       391

    accuracy                           0.65       764
   macro avg       0.65      0.65      0.65       764
weighted avg       0.65      0.65      0.65       764
```

3. BERT

Performed several experiments with BERT. The Steps are as follows -

- Used Bert Tokenizer (bert-base-uncased)
- Used BCEWithLogitsLoss to calculate the loss and hence no activation function was used.
- FeedForward neural network was used to train the model

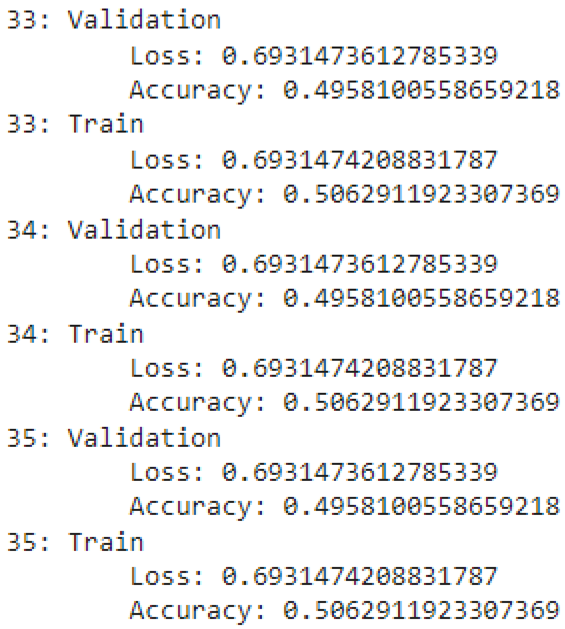
Performed experiments with the combination of the following parameters-

- Percentage combination of Training, validation and testing datasets - [80,10,10] , [80,5,15], [75,10,15], [70,10,20], [70,15,15], [90,5,5], [85,5,10], [60,20,20], [60,10,30]
- Nn Topology - {64,128,64,32,1}, {64,256,128,64,32,1}, {64,256,64,32,1}, {64, 256,32,1}, {64,128,32,1}
- Learning Rates - 0.1, 0.2, 0.01, 0.05, 0.001, 0.0001
- ePochs - 10, 15, 20, 25, 30, 50, 70, 80, 100, 150

Results - Since the model was getting trained within seconds, it was possible to experiment a lot of combinations of hyperparameters. But unfortunately, even after training the model with a lot of combinations, the model gave 50% training accuracy, 100% validation accuracy and around 50% testing accuracy. Since the model gave such high validation accuracy the model might have got overtrained. The maximum testing accuracy that was possible with BERT was 55%.

Below is the output for the Train and Validation when the hyperparameters were-

- Number of training data: 1700
- Number of validation data: 700
- Number of testing data: 1400
- Neural Network topology: [64, 128, 64, 32, 1]
- Epochs: 35
- Learning Rate: 0.01



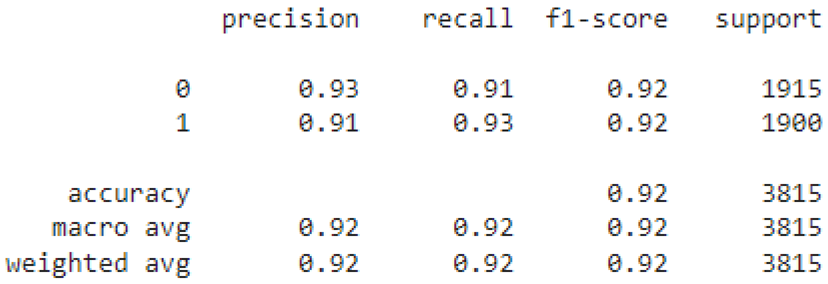
```
33: Validation
        Loss: 0.6931473612785339
        Accuracy: 0.4958100558659218
33: Train
        Loss: 0.6931474208831787
        Accuracy: 0.5062911923307369
34: Validation
        Loss: 0.6931473612785339
        Accuracy: 0.4958100558659218
34: Train
        Loss: 0.6931474208831787
        Accuracy: 0.5062911923307369
35: Validation
        Loss: 0.6931473612785339
        Accuracy: 0.4958100558659218
35: Train
        Loss: 0.6931474208831787
        Accuracy: 0.5062911923307369
```

4. RoBertA -

Used the Hugging Face transformer - twitter-roberta-base-irony.
The link for the transformer used is -
https://huggingface.co/cardiffnlp/twitter-roberta-base-irony/blame/main/README.md

The encoder-decoder transformer has been specially trained to detect irony in the tweets.
The pre-trained model performed exceptionally well. Also the default size of the dataset is 3817 rows.
The experiments performed, along with results are as follows -

- Testing on dataset 1 - 91.9% accuracy

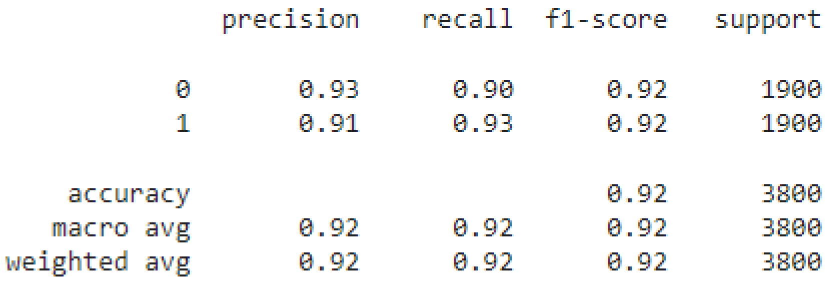


```
              precision    recall  f1-score   support

           0       0.93      0.91      0.92      1915
           1       0.91      0.93      0.92      1900

    accuracy                           0.92      3815
   macro avg       0.92      0.92      0.92      3815
weighted avg       0.92      0.92      0.92      3815
```

- Testing on dataset with equal number on irony and not irony labels - 91.8%

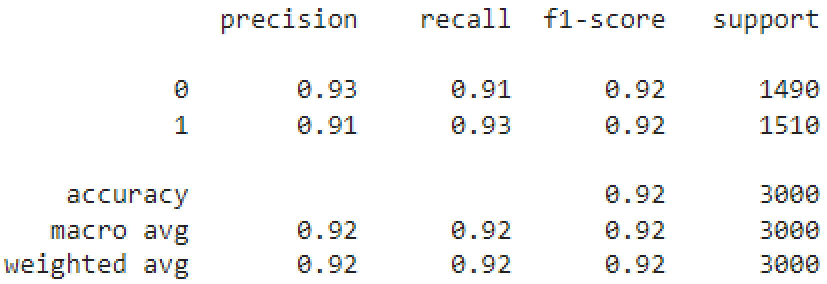


```
              precision    recall  f1-score   support

           0       0.93      0.90      0.92      1900
           1       0.91      0.93      0.92      1900

    accuracy                           0.92      3800
   macro avg       0.92      0.92      0.92      3800
weighted avg       0.92      0.92      0.92      3800
```

- Testing on Dataset 1 with 3000 rows- 91.93%

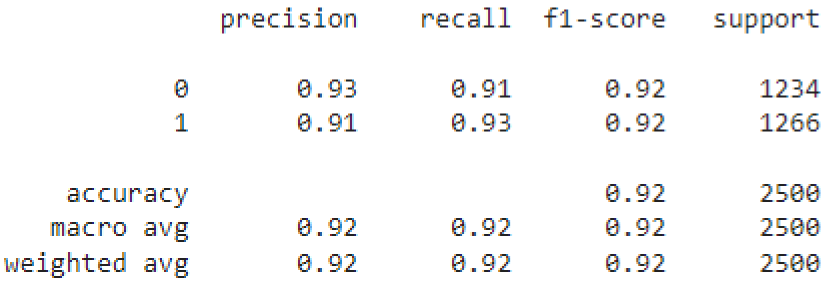


```
              precision    recall  f1-score   support

           0       0.93      0.91      0.92      1490
           1       0.91      0.93      0.92      1510

    accuracy                           0.92      3000
   macro avg       0.92      0.92      0.92      3000
weighted avg       0.92      0.92      0.92      3000
```

- Testing on Dataset 1 with 2500 rows- 92.4%

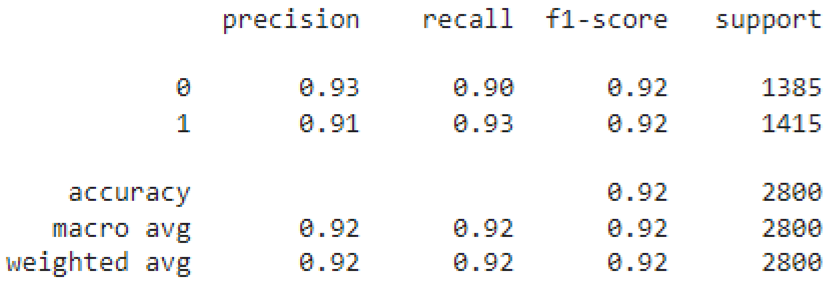


```
              precision    recall  f1-score   support

           0       0.93      0.91      0.92      1234
           1       0.91      0.93      0.92      1266

    accuracy                           0.92      2500
   macro avg       0.92      0.92      0.92      2500
weighted avg       0.92      0.92      0.92      2500
```

- Testing on Dataset 1 with 2800 rows - 91.89%

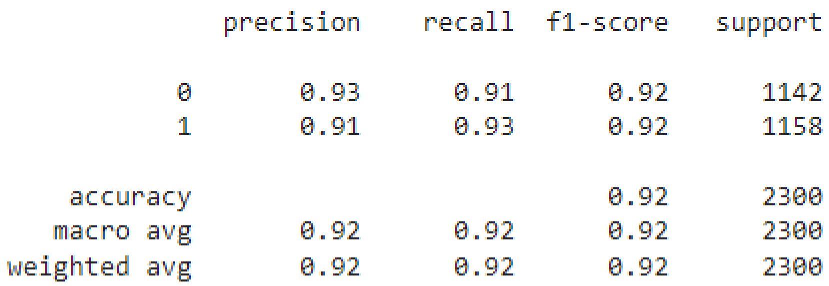


```
              precision    recall  f1-score   support

           0       0.93      0.90      0.92      1385
           1       0.91      0.93      0.92      1415

    accuracy                           0.92      2800
   macro avg       0.92      0.92      0.92      2800
weighted avg       0.92      0.92      0.92      2800
```

- Testing on Dataset 1 with 2300 rows -

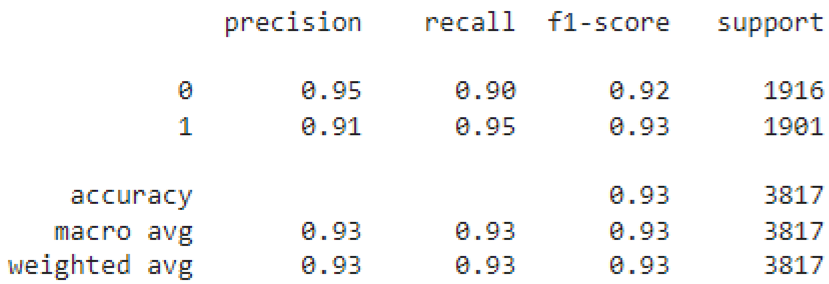


```
              precision    recall  f1-score   support

           0       0.93      0.91      0.92      1142
           1       0.91      0.93      0.92      1158

    accuracy                           0.92      2300
   macro avg       0.92      0.92      0.92      2300
weighted avg       0.92      0.92      0.92      2300
```

- Testing with Dataset 2 - 92.58% accuracy

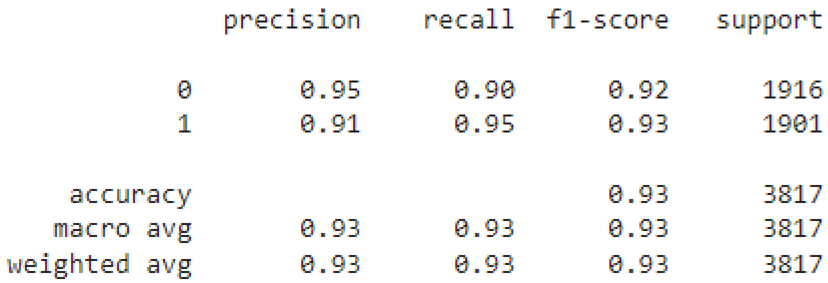


```
              precision    recall  f1-score   support

           0       0.95      0.90      0.92      1916
           1       0.91      0.95      0.93      1901

    accuracy                           0.93      3817
   macro avg       0.93      0.93      0.93      3817
weighted avg       0.93      0.93      0.93      3817
```

- Testing with Dataset 3 - 92.58% accuracy



```
              precision    recall  f1-score   support

           0       0.95      0.90      0.92      1916
           1       0.91      0.95      0.93      1901

    accuracy                           0.93      3817
   macro avg       0.93      0.93      0.93      3817
weighted avg       0.93      0.93      0.93      3817
```

Conclusion

The dataset gave the best accuracy with the pre-trained RoberTa transformer which is 92.58% when tested on dataset having both emojis and hashtags. The result table for the major tests performed on Roberta is as follows -

S. no	DATASET	ACCURACY
1	Dataset without emojis and hashtags.	91.9
2	Dataset with emojis and without hashtags	92.58
3	Dataset with emojis and hashtags	92.58

Future Scope

The dataset can also be used to fine-tune the GPT-3 model and then analyze the accuracy as GPT-3 gives good results with text classification tasks, but because of the limitations of the GPT-3, the model wouldn’t have been tested only on a few samples as it is not completely free.