Sentiment Analysis on Twitter

Grazia Noemi Compagnino noemicompagnino97@live.it
Maqbool Thoufeeq Tharayil thrmbl95b15z222z@studium.unict.it

1. Twitter model description

We describe here **Twitter Sentiment Analysis**, a binary dataset composed by 31962 tweets. The goal is to allow to Twitter's system to individualize hates speech on Twitter's tweets (see Fig.1). Sentiment analysis is a technique through which you can analyze a piece of text to determine the sentiment behind it. It combines Machine Learning and Natural Language Processing (NLP) to achieve this. Using basic Sentiment analysis, a program can understand whether the sentiment behind a piece of text is positive, negative, or neutral.This analysis has the scope to reduce cyber-bullism given by hates tweets which are sexists or racists tweets thanks to the help of Machine Learning's negative sentiments identification. The training set is composed by labels and tweets and it is a binary one, hence labels are equal to "0" or to "1", where "0" stands for a positive sentiment and "1" for a negative one which denote the racist/sexist tweet. The goal is to predict the labels on the test dataset.

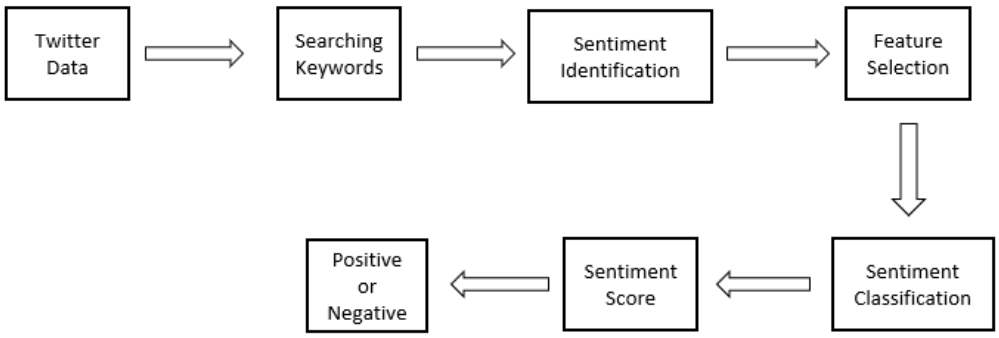

Figure 1: Neural Network of the proposed model.

We initially have a categorical dataset, since it is composed by words and then text. So, we pre-process the dataset in order to proceed with our analysis since we can analyze the dataset only by transforming it in a numeric one, because the Recurrent Neural Network (RNN) accepts numbers as parameters rather than strings. Then we are creating a vocabulary by associating to each tweet a word of the same tweet and successively we associate to each word the correspondent number of the tweets. This process is called **Embedding**.

We compute the maximum length of each tweet, because we know that The tweets have different lengths, while, we want that all tweets have the same length. Then we compute the average length of each tweets and choose a sequence length of word for Padding tweets. This process is not necessary to deal with RNN, because from theory we are able to deal with RNN of arbitrary length, but it is mandatory when we have a fixed batch structure.

2. Twitter's Dataset

The dataset was provided by Analytics Vidhya. The **Twitter Sentiment Analysis** dataset consists of 31962 tweets of which at least the 93% (29720 tweets) of dataset is Negative class (0) and only the 7 % (2242 tweets) is the Positive class (1).

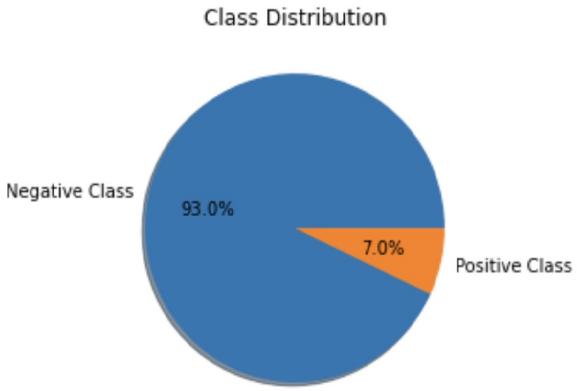

Figure 2: Piechart of Positive "1" and Negative "0" classes of Twitter's dataset not balanced

Then, labels in the binary dataset are divided in:

• **Negative Class :** 0 - which denotes the tweet is not racist/sexist, and then a positive sentiment. It represent the 93 % of tweets (29720 tweets);

• **Positive Class :** 1 - which denotes the tweet is racist/sexist, and then a negative sentiment. It corresponds to the 7 % of tweets (2242 tweets).

How we can see, this dataset is not well balanced since it has got a a very big number of tweets belonging to the Negative Class rather than the Positive one.
If we try to fit this initial model we see that the training accuracy is so high, about the 99 %, but we can say that this result is not true. This means that, since the dataset is composed by more elements of Negative Class the model is used to predict the majority class as normality. So we do different trials to obtain the best performance of the model. The first time, we try to fit the model by balancing only test set and validation set. So, we'll have 400 in validation and 800 in test samples, respectively, and the remaining part forms the training sample (30762 tweets). The second trial provides that we proceed to fit the model by balancing the dataset through Weighted-Cross-Entropy-Loss by giving to each Class different weights in the way that the minority Class (Positive Class) has more weight (the weight for 1 is about 9.36 and for 0 is 0.53) and elements for each set remain the same as before. The third attempt consists in the under-sampling technique to reduce the initial

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

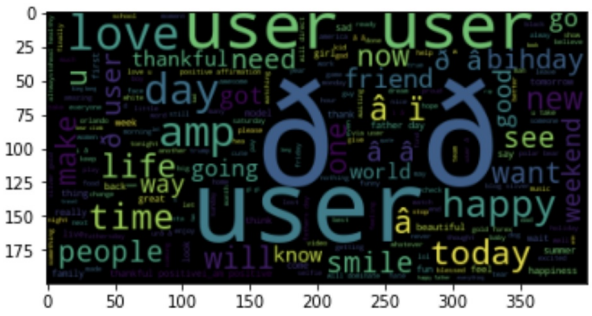

Figure 3: The head of the Twitter's dataset on the left side and a random sample of tweets on the right one.

training dataset. Now, we have a dataset composed by 4484 tweets but well balanced for each subset, since it is represented for the 50 % by the Negative Class and for the remaining percentage by the Positive one.

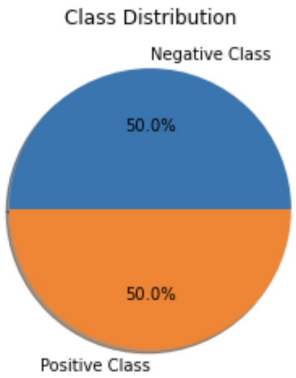

Figure 4: Piechart of Positive "1" and Negative "0" classes of Twitter's dataset balanced.

So, we will have 4484 tweets in the dataset, so divided: 3284 in training sample, and 400 in validation and 800 in test samples.

3. Training procedure

RNN is a network that works on the present input by taking into account the previous output and storing in its memory for a short period of time (short-term memory). It's a type of artificial neural network which uses sequential data or time series data. The same deep learning algorithms is used for ordinal or temporal problems, such as language translation, Natural Language Processing (NLP), speech recognition and image captioning; they are incorporated into popular applications such as Siri, voice search, and Google Translate. Recurrent neural networks are distinguished by their "memory" as they take information from prior inputs to influence the current input and output. The model is composed by

an embedding layer, a Recurrent Neural Network (RNN) cell and a linear output for the prediction of dimension 2, which represents Positive and Negative Classess, where the RNN uses back-propagation algorithm to compute errors for each epoch. We have a RNN cell of dimension (1024,256) for each step. Model parameters are length of vocabulary + 1, embedded size (1024) and RNN cell of size 256 (where 256 stands for neurons for each layer). The embedding layer is a vector of dimension (5,10) where 5 stands for the initial number of embedding i.e. the number of words in our vocabulary, and 10 stands for the embedding dimension that is a vector that the model has to learn for each word. This configuration represents the initial embedding layer chosen by our-selves. This is the initial one but will be the model itself to find the optimal solution to solve the problem. The embedding layer is formed by all words in the vocabulary + 1 and has a dimension of 10.

All the models are trained from scratch for 100 epochs on a Nvidia GeForce RTX 2060 with 32 GB of VRAM. Adam's optimizer is chosen as optimizer algorithm using a learning rate of 0.001 and a batch size of 64.

4. Experimental Results

After the three attempts we have done on **Twitter Sentiment Analysis** model, we have to choose the best technique to fit the model. We can say that, basing on the highest Validation Accuracy of epochs in each training of the model and considering also the result of the test, that the technique of Weighted-Cross-Entropy-Loss is the best to predict the model in a correct way. Thus we analyze these data from model computing. After **Twitter Sentiment Analysis** model's training the average accuracy is the 0.9729, 0.8415 and 0.8089 for training, validation and test set, respectively. Instead, the average loss is the 0.0699, 1.0987 and 0.9043 for training, validation and test set, respectively (Table 1). The Sentiment Analysis gives a Positive Sentiment of 100 % since the Negative Class (0) corresponds to the Positive Sentiment for which there is not a racism or sexism in tweet. Model was trained as described in the previous section.

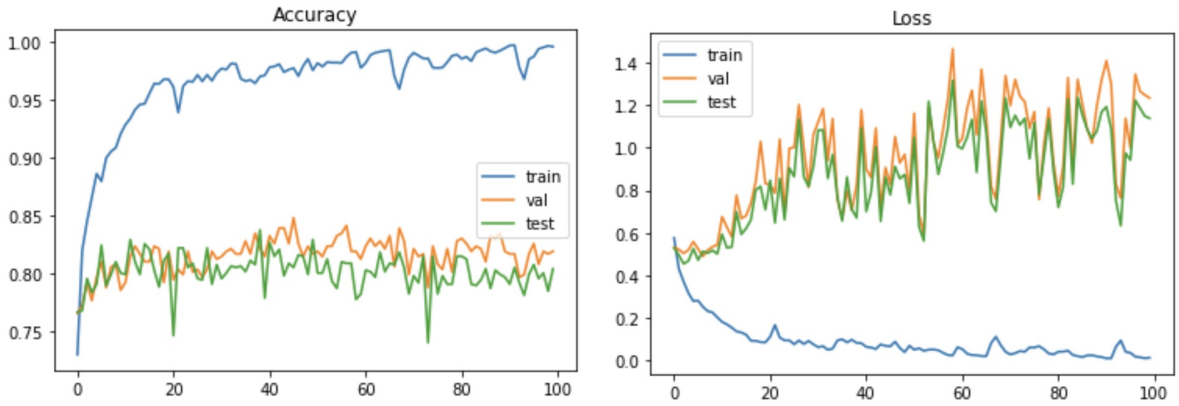

Figure 5: The accuracy of our model on the left side and the loss of tweets on the right one of Twitter's dataset.

Subset	Accuracy	Loss
Training set	0.97	0.07
Validation set	0.84	1.10
Test set	0.81	0.90

Table 1: Training, validation and test performance of the model.

After that, we've predicted the model for the training, validation and test set and we obtain that the predicted tweet is equal to the actual tweet in each of these cases. Labels are the known values for old data and prediction is your predicted value for new data, where you do not have a label. Then during training, you try to make your predictions match labels. For example we have:

```
tweets_or[tvalue_test[0]]
```

```
'i am thankful for my school thankful positive     '
```

```
print('Label:', tweets_or[test_dataset[label][0][0]], ', Predicted:', tweets_or[predict_text_test(text, model)[0][0]])
```

```
Label: i am thankful for my school thankful positive      , Predicted: i am thankful for my school thankful positive
```

Figure 6: An example of predicted and actual review from the code.

We also plot the confusion matrix: it's a 2x2 matrix since we have only two labels (Negative Class "0", to which is associated a Positive Sentiment, and Positive Class "1", to which is associated a Negative Sentiment). The confusion matrix is a contingency table containing the information about actual and predicted classifications, which indicates how much better the used model classifies the values in a correct way by categorizing predicted values against actual values, arranging them in each row and in each column, respectively. The sum of the diagonal elements of the matrix gives the number of right classified units. Instead, the sum of the off-diagonal elements gives the number of wrong classified units. If we divide the sum of the off-diagonal elements by the number of the total observations and multiply for 100, we obtain the misclassification error that is the percentage of the misclassified units. There are two kinds of misclassifications: False Positive Misclassification and False Negative Misclassification. They are:

• **True Negative value (TN):** in which the model predicts what is expected. This is not an event;

• **False Positive value (FP):** It was predicted by the model that there is an event, but actually there is not;

• **False Negative value (FN):** the model predicts that there is not an event, but actually there is an event;

• **True Positive value (TP):** the model predicts what is expected. This is an event.

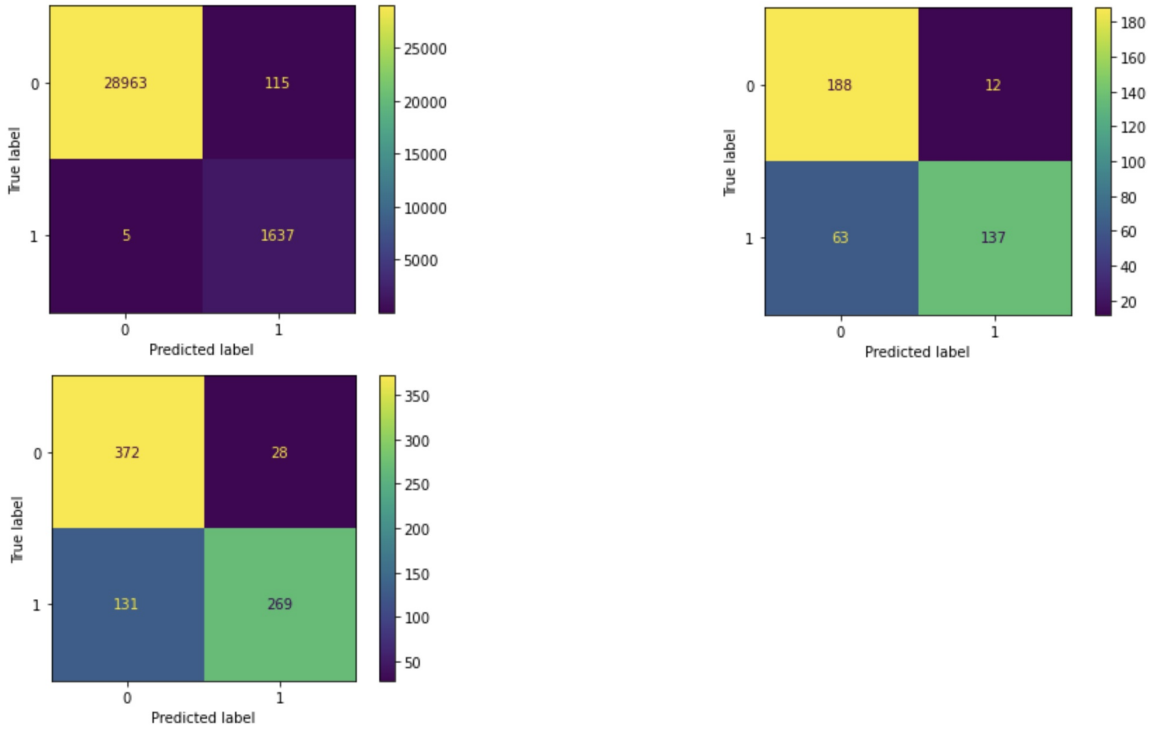

Figure 7: Confusion Matrix on train, validation and test sets.