Tunisian Republic
Ministry of Higher Education and Scientific Research
University of Tunis El Manar
National Engineering School of Tunis

Information and Communication Technologies Department

NLP Project

Textual Data Sentiment Analysis.

Supervised by :

Ms. Sabrine Krichen and M. Fares El Kahla

Elaborated by :

Chiheb Sahbani and Ahmed Monsri

Academic year : 2022/2023

Acknowledgement

Before going through the details of this project, it is appropriate to start by thanking those who have taught us so much during this module As a tribute to their compassion, we would like to thank Our teachers M. **Fares El Kahla** and Ms. **Sabrine Krichen**, who encouraged, guided and gave us their precious pieces of advice throughout the different parts of this course.
We hope that our behavior and learning have given a good impression of **ENIT** and confirmed its image.

Table des matières

Table des figures

Introduction

Sentiment analysis is the computational study of people's sentiments, attitudes, and emotions expressed in written language. It is one of the most active research areas in natural language processing and text mining in recent years. Its popularity is mainly due to two reasons. First, it has a wide range of applications because sentiments are central to almost all human activities and are key influencers of our behaviors. Whenever we need to make a decision, our sentiments have a major part in this decision making process. Second, it presents many challenging research problems, which had never been attempted before the year 2000. Part of the reason for the lack of study before was that there was little sentimental text in digital forms. It is thus no surprise that the inception and the rapid growth of the field coincide with those of the social media on the Web. In fact, the research has also spread outside of computer science to management sciences and social sciences due to its importance to business and society as a whole.
In this Rapport, we will start with the opening chapter, titled "Description of the implementation" which was devoted to outlining the project description, outlining the suggested solutions and explains how it was implemented. The second chapter focuses on a review of the model architecture and hyperparameters. The third chapter highlights the data preprocessing while also mentioning the used method. The fourth and final chapter will be devoted to evaluating the suggested solution by showing detailed results.

Chapitre 1

Project presentation

Introduction

In this first chapter, we give an overview of our project. To do so, we will first discuss the context, then we will describe the importance and main characteristics of the project, and present the proposed solution.

1.1 Context

This project has been done as a part of our course toward obtaining the national diploma of telecommunications engineering from the National Engineering School of Tunis. Supervised by M. **Fares El Kahla** and Ms. **Sabrine Krichen**, in order to succeed in the "Deep Learning" module. The main goal of this project is to build an algorithm capable of identifying sentiment types based on the textual tweets dataset. To investigate the sentiment analysis field of deep learning since it allows us to approach natural language processing which is a very hot topic actually.

1.2 Project description

Sentiment analysis also referred to as opinion mining, is a sub-machine learning task where we want to determine which is the general sentiment of a given document. Using machine learning techniques and natural language processing we can extract the subjective information of a document and try to classify it according to its polarity such as positive or negative. It is a really useful analysis since we could possibly determine the overall opinion about a selling object, or predict stock markets for a given company, if most people think positively about it, possibly its stock markets will increase, and so on.

1.3 Problematic

One of the major challenges in this project is dealing with the complexity of natural language, including issues such as subjectivity/objectivity, negation, vocabulary, and grammar. These challenges make sentiment analysis a difficult task and also make it an interesting area to work on.

1.4 Proposed solution

In this project, we choose to work with Twitter, a micro-blogging website where people can share their feelings quickly and spontaneously by sending tweets limited to 140 characters. And try to classify tweets into "positive" or "negative" sentiments by building a sequential RNN model and also by fun-tuning a pre-trained model so that we can compare the performances.

Conclusion

In this chapter, we presented the general description of the project carried out, as well as the Problematic, And finally explained the solutions that we proposed to deal with the sentiment analysis problem.

Chapitre 2

Data Exploration and Preprocessing

2.1 Introduction

The data must be carefully investigated and prepared in order to obtain good results because this pre-processing is essential for the production of an accurate prediction at the stage of model development.
In this chapter, we will present our data, visualize it and cite the preprocessing techniques used in order to increase our model performances.

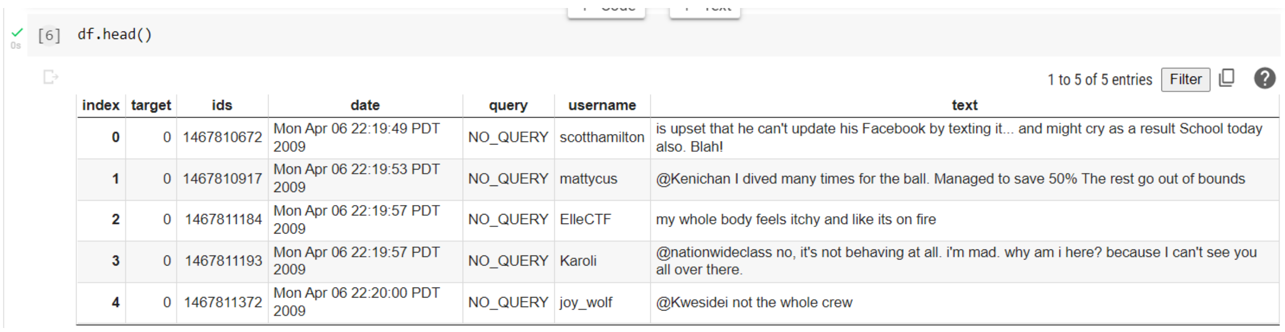
2.2 Exploratory Data Analysis

Exploratory data analysis is a data analytics procedure used to fully comprehend the data and discover its various characteristics, often using visual methods.
This enables you to better understand your data and identify insightful patterns in it.

FIGURE 2.1 – Data Exploration.

2.2.1 Data Presentation

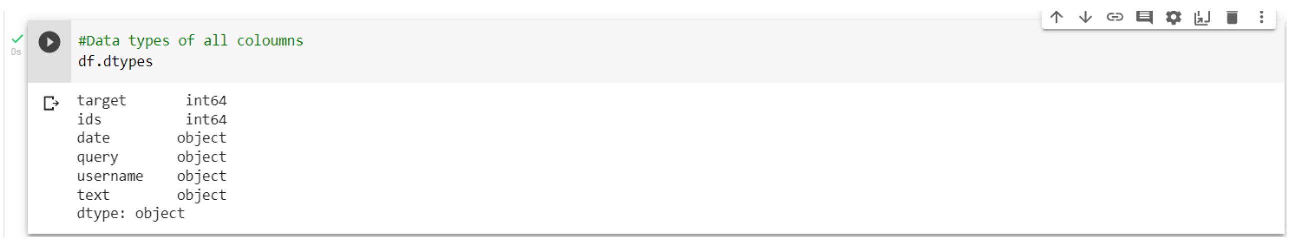
To realize this NLP project we used the sentiment140 dataset from Kaggle. It contains 1,600,000 tweets extracted using the Twitter API. The tweets have been annotated (0 = negative, 4 = positive) and they can be used to detect sentiment. It contains the following 6 fields :
target : the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)
ids : The id of the tweet (2087)
date : the date of the tweet (Sat May 16 23 :58 :44 UTC 2009)
flag : The query (lyx). If there is no query, then this value is NOQUERY.
user : the user that tweeted (robotickilldozr).
text : the text of the tweet (Lyx is cool).



```
[6] df.head()
```

index	target	ids	date	query	username	text
0	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by texting it... and might cry as a result School today also. Blah!
1	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Managed to save 50% The rest go out of bounds
2	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
3	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all. i'm mad. why am i here? because I can't see you all over there.
4	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew

FIGURE 2.2 – Data Presentation.



```
#Data types of all coloumns
df.dtypes
```

```
target       int64
ids          int64
date        object
query       object
username    object
text        object
dtype: object
```

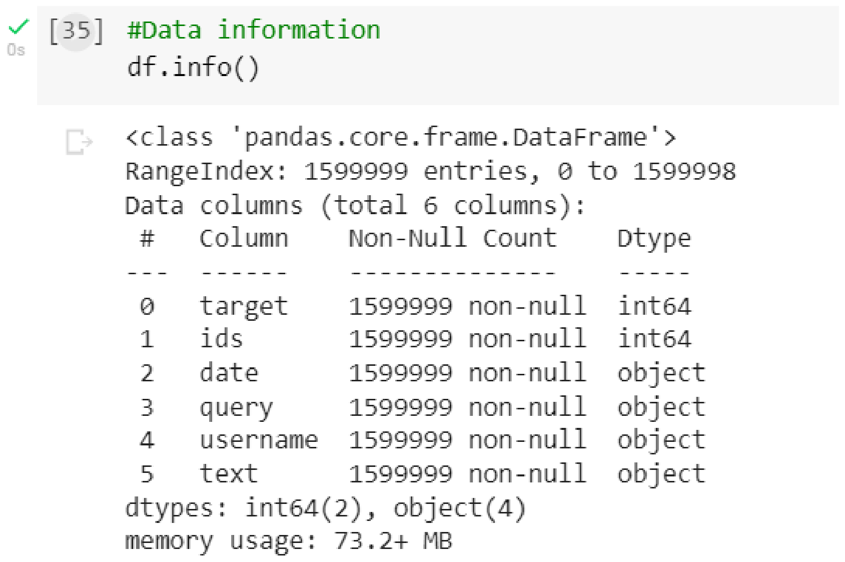
FIGURE 2.3 – Data Types.

2.2.2 Data Exploration

In our exploratory analysis, we will conduct an EDA with the help of data visualization. More specifically, we will focus on seaborn, a Python library that is built on matplotlib and supports NumPy and Pandas, allowing us to create interesting and informative statistical graphs

2.2.2.1 General variable exploration

To print information on our Data Frame, we used the Pandas library's info() function.



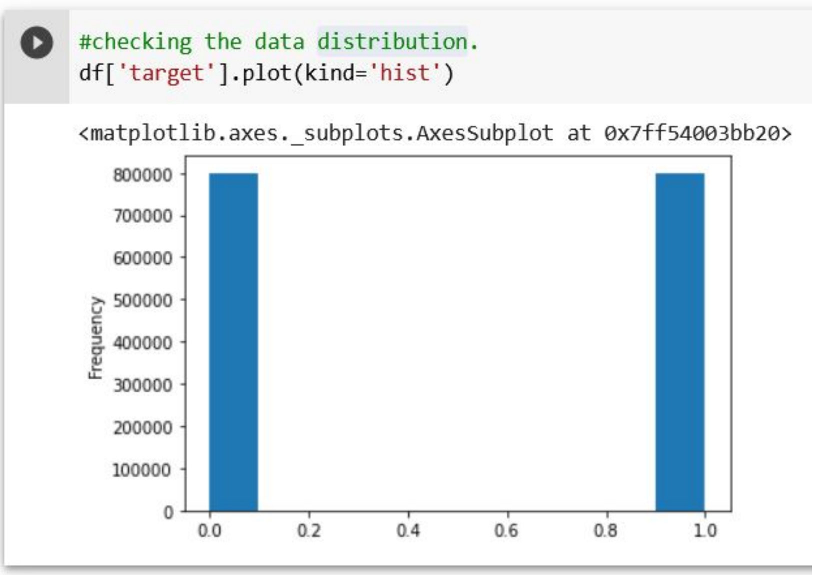
```
[35] #Data information
     df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599999 entries, 0 to 1599998
Data columns (total 6 columns):
 #   Column    Non-Null Count    Dtype 
---  ------    --------------    ----- 
 0   target    1599999 non-null  int64 
 1   ids       1599999 non-null  int64 
 2   date      1599999 non-null  object
 3   query     1599999 non-null  object
 4   username  1599999 non-null  object
 5   text      1599999 non-null  object
dtypes: int64(2), object(4)
memory usage: 73.2+ MB
```

FIGURE 2.4 – Data Information.

2.2.2.2 Data distribution

Data distribution refers to the way that values in a dataset are distributed or spread out. Understanding the distribution of the data is important in data analysis, as the distribution can affect the performance of machine learning algorithms, and may indicate the presence of outliers or other anomalies.



```
#checking the data distribution.
df['target'].plot(kind='hist')
```

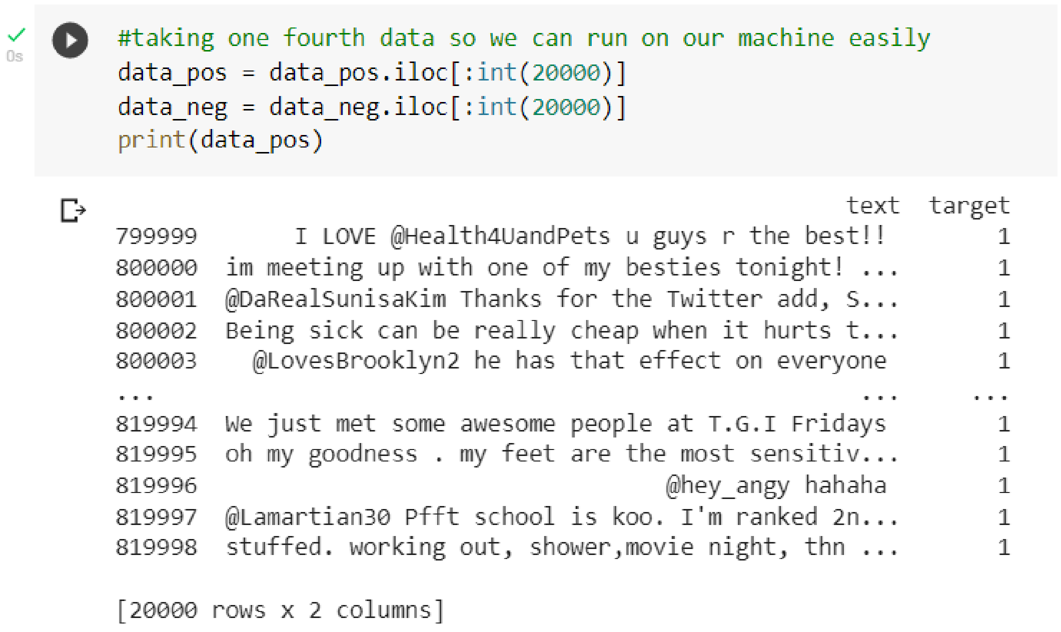
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff54003bb20>
```

FIGURE 2.5 – Data distribution

2.3 Data preprocessing

Data preprocessing is an important step in natural language processing (NLP) as it helps to make the raw text data usable for downstream NLP tasks such as text classification, sentiment analysis, and information extraction. In this project, we realized some common preprocessing steps that are typically performed on text data in NLP such as the following.

In order to easily run our model we are going to only use one-fourth of our data.



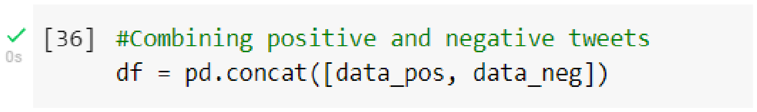
```
#taking one fourth data so we can run on our machine easily
data_pos = data_pos.iloc[:int(20000)]
data_neg = data_neg.iloc[:int(20000)]
print(data_pos)
```

```
                                                     text  target
799999         I LOVE @Health4UandPets u guys r the best!!       1
800000  im meeting up with one of my besties tonight! ...       1
800001  @DaRealSunisaKim Thanks for the Twitter add, S...       1
800002  Being sick can be really cheap when it hurts t...       1
800003    @LovesBrooklyn2 he has that effect on everyone        1
...                                                   ...     ...
819994  We just met some awesome people at T.G.I Fridays        1
819995  oh my goodness . my feet are the most sensitiv...       1
819996                               @hey_angy hahaha           1
819997  @Lamartian30 Pfft school is koo. I'm ranked 2n...       1
819998  stuffed. working out, shower,movie night, thn ...       1

[20000 rows x 2 columns]
```

FIGURE 2.6 – Used Data.

After that, we combined the positive data with the negative tweets.

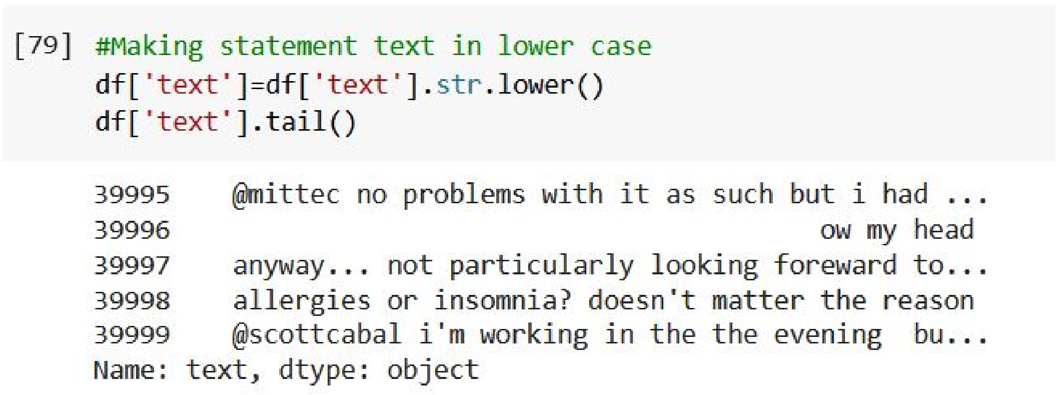


```
[36] #Combining positive and negative tweets
     df = pd.concat([data_pos, data_neg])
```

FIGURE 2.7 – Combined Data.

2.3.1 Lowercasing

Text data is often converted to lowercase to ensure that words are not treated as different due to different capitalization.



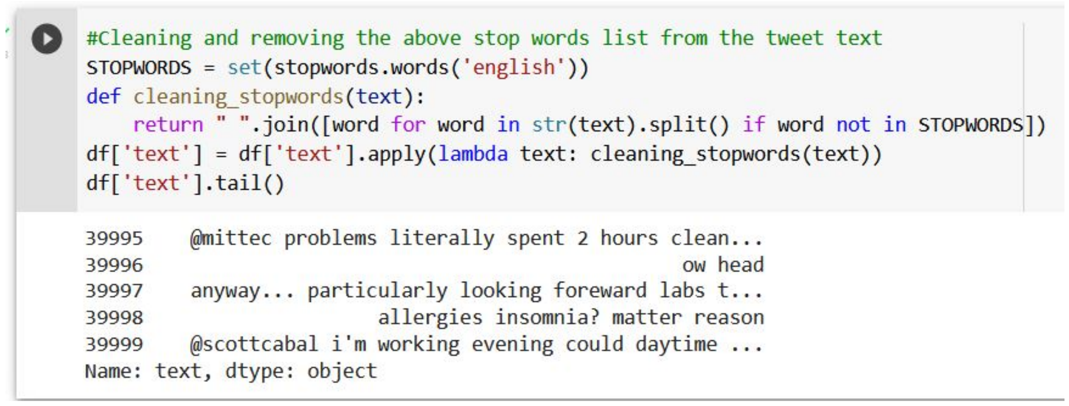
```
[79] #Making statement text in lower case
     df['text']=df['text'].str.lower()
     df['text'].tail()

     39995    @mittec no problems with it as such but i had ...
     39996                                            ow my head
     39997    anyway... not particularly looking foreward to...
     39998    allergies or insomnia? doesn't matter the reason
     39999    @scottcabal i'm working in the the evening  bu...
     Name: text, dtype: object
```

FIGURE 2.8 – Lowercased data

2.3.2 Removing stop words

Stop words are common words that do not carry much meaning, such as "the", "and", "is", etc. Removing stop words can help to reduce the dimensionality of the data and focus on the most important words. To do so, we used the Natural Language Toolkit (NLTK) library for Python includes a built-in list of stopwords that can be used in various NLP tasks, put all the words in this list in a single string variable, and then we created this function to remove this words list from the tweet text.



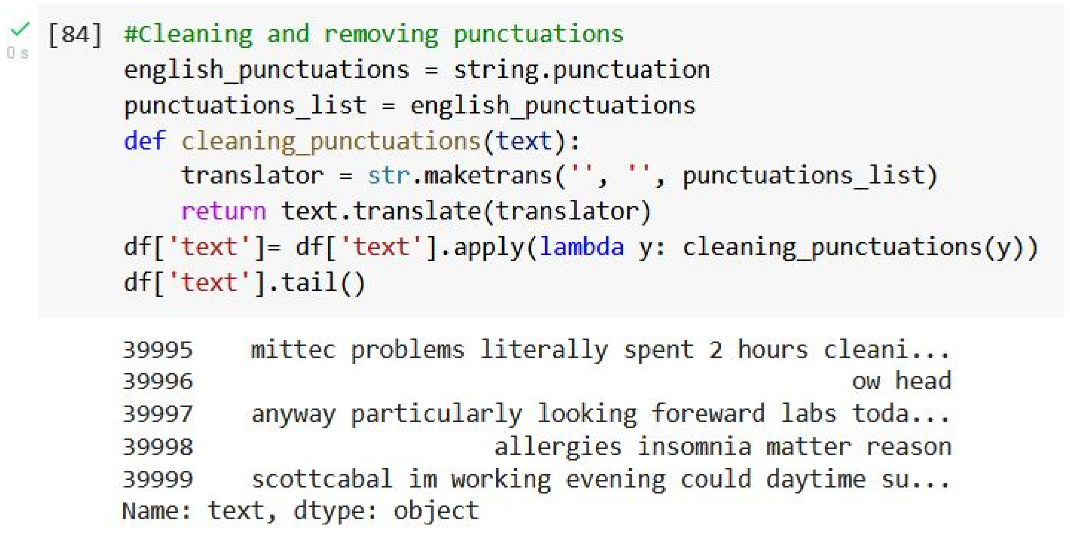
```
#Cleaning and removing the above stop words list from the tweet text
STOPWORDS = set(stopwords.words('english'))
def cleaning_stopwords(text):
    return " ".join([word for word in str(text).split() if word not in STOPWORDS])
df['text'] = df['text'].apply(lambda text: cleaning_stopwords(text))
df['text'].tail()

39995    @mittec problems literally spent 2 hours clean...
39996                                              ow head
39997    anyway... particularly looking foreward labs t...
39998                    allergies insomnia? matter reason
39999    @scottcabal i'm working evening could daytime ...
Name: text, dtype: object
```

FIGURE 2.9 – Cleaning stop words list from the tweet text

2.3.3 Removing punctuation

Punctuation marks are removed from the text data as they generally do not carry much meaning in NLP tasks.



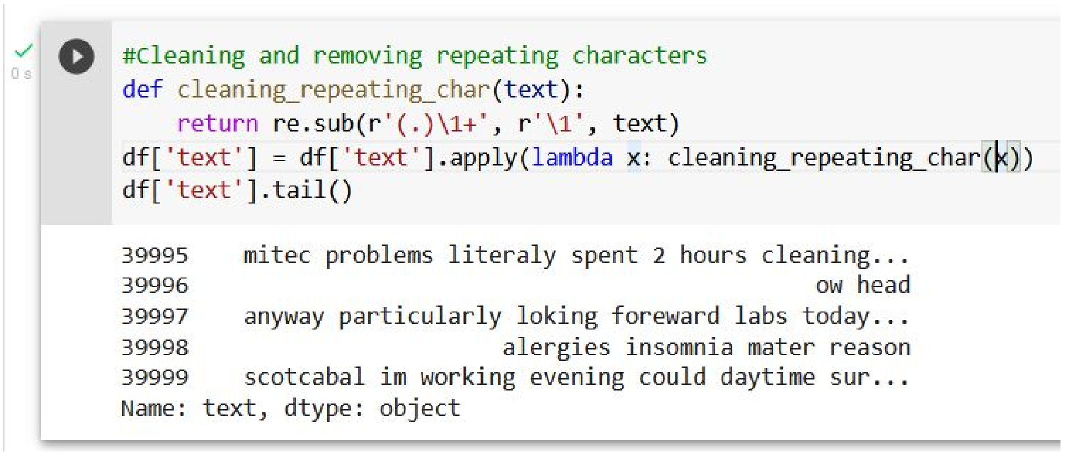
```
[84] #Cleaning and removing punctuations
     english_punctuations = string.punctuation
     punctuations_list = english_punctuations
     def cleaning_punctuations(text):
         translator = str.maketrans('', '', punctuations_list)
         return text.translate(translator)
     df['text']= df['text'].apply(lambda y: cleaning_punctuations(y))
     df['text'].tail()
```

```
39995    mittec problems literally spent 2 hours cleani...
39996                                              ow head
39997    anyway particularly looking foreward labs toda...
39998                   allergies insomnia matter reason
39999    scottcabal im working evening could daytime su...
Name: text, dtype: object
```

FIGURE 2.10 – Cleaning punctuations

2.3.4 Removing repeating characters

Cleaning text data and removing repeating characters is an important step in natural language processing (NLP) as it can help to improve the performance of the sentiment analysis task.



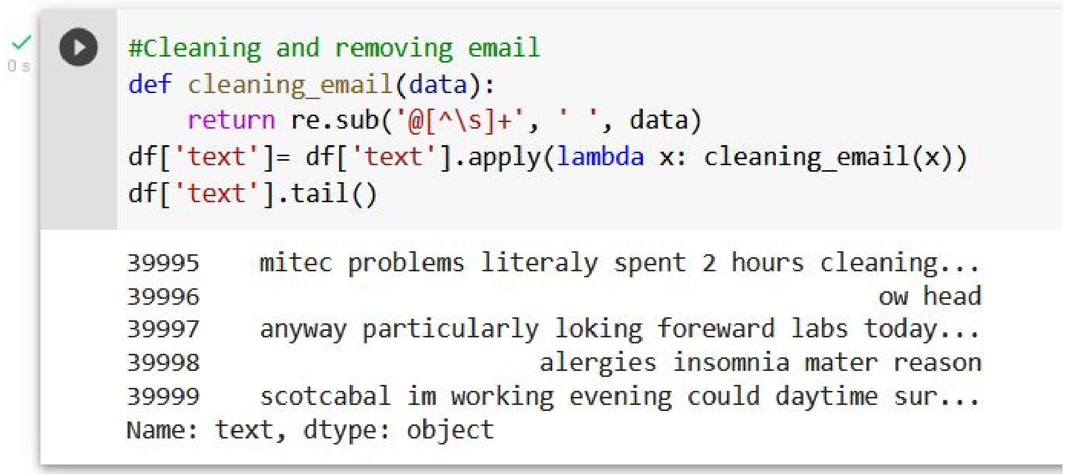
```
#Cleaning and removing repeating characters
def cleaning_repeating_char(text):
    return re.sub(r'(.)\1+', r'\1', text)
df['text'] = df['text'].apply(lambda x: cleaning_repeating_char(x))
df['text'].tail()
```

```
39995    mitec problems literaly spent 2 hours cleaning...
39996                                           ow head
39997    anyway particularly loking foreward labs today...
39998                      alergies insomnia mater reason
39999    scotcabal im working evening could daytime sur...
Name: text, dtype: object
```

FIGURE 2.11 – Cleaning repeating characters

2.3.5 Deleting Email Addresses

Removing email addresses from text data is an important step in natural language processing (NLP), as it protects the privacy and prevents unauthorized use of personal information. We can apply this using regular expressions, as shown in the figure below.



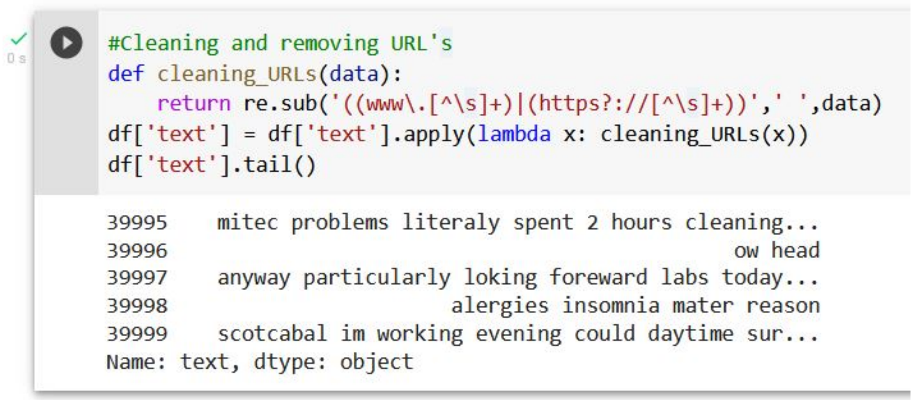
```
#Cleaning and removing email
def cleaning_email(data):
    return re.sub('@[^\s]+', ' ', data)
df['text']= df['text'].apply(lambda x: cleaning_email(x))
df['text'].tail()
```

```
39995    mitec problems literaly spent 2 hours cleaning...
39996                                              ow head
39997    anyway particularly loking foreward labs today...
39998                         alergies insomnia mater reason
39999    scotcabal im working evening could daytime sur...
Name: text, dtype: object
```

FIGURE 2.12 – Cleaning emails

2.3.6 Removing URLs

Removing URLs from text data is an essential step as it can help improve the performance of the model. URLs can also contain sensitive information and their removal can help protect privacy.



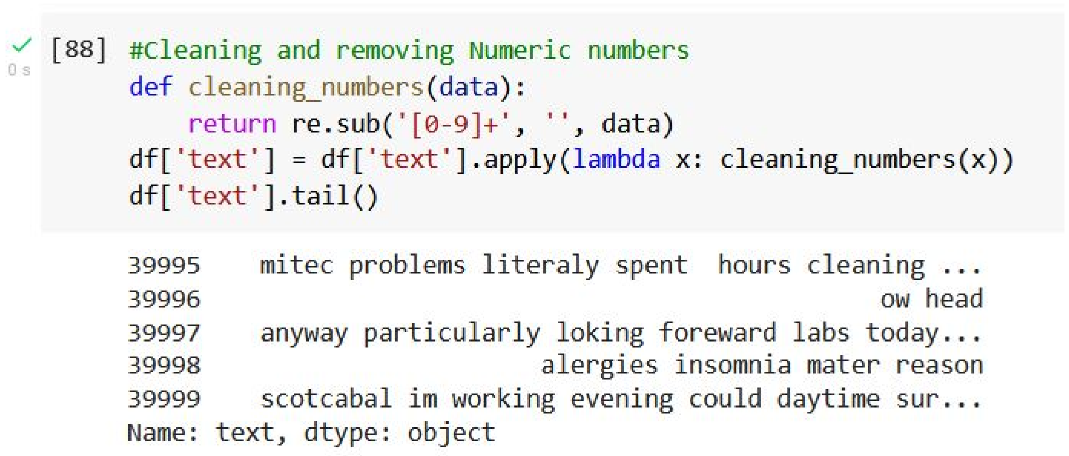
```
#Cleaning and removing URL's
def cleaning_URLs(data):
    return re.sub('((www\.[^\s]+)|(https?://[^\s]+))',' ',data)
df['text'] = df['text'].apply(lambda x: cleaning_URLs(x))
df['text'].tail()
```

```
39995    mitec problems literaly spent 2 hours cleaning...
39996                                              ow head
39997    anyway particularly loking foreward labs today...
39998                         alergies insomnia mater reason
39999    scotcabal im working evening could daytime sur...
Name: text, dtype: object
```

FIGURE 2.13 – Cleaning URL's

2.3.7 Removing numeric numbers

Numbers are removed, as they don't carry much meaning in NLP tasks.



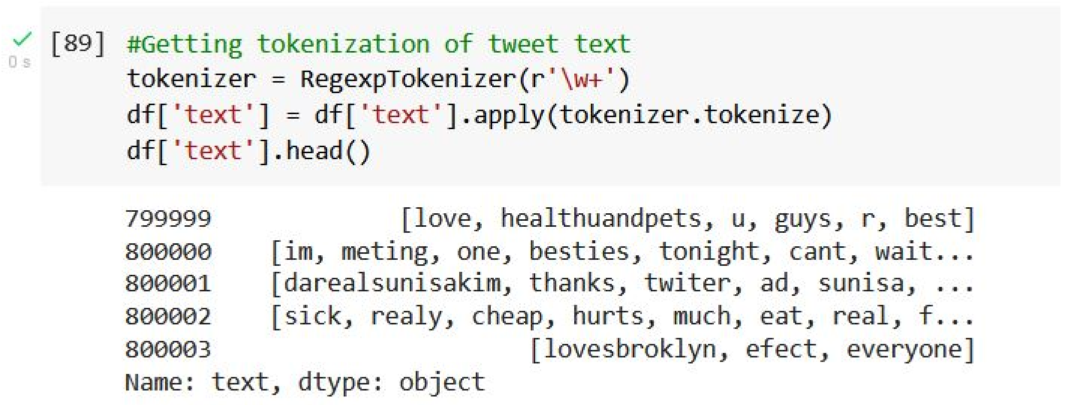
```
[88] #Cleaning and removing Numeric numbers
     def cleaning_numbers(data):
         return re.sub('[0-9]+', '', data)
     df['text'] = df['text'].apply(lambda x: cleaning_numbers(x))
     df['text'].tail()
```

```
39995    mitec problems literaly spent  hours cleaning ...
39996                                             ow head
39997    anyway particularly loking foreward labs today...
39998                       alergies insomnia mater reason
39999    scotcabal im working evening could daytime sur...
Name: text, dtype: object
```

FIGURE 2.14 – Cleaning numeric numbers.

2.3.8 Tokenization

Tokenization is the process of splitting a piece of text into individual words, phrases, or other meaningful elements, known as tokens.



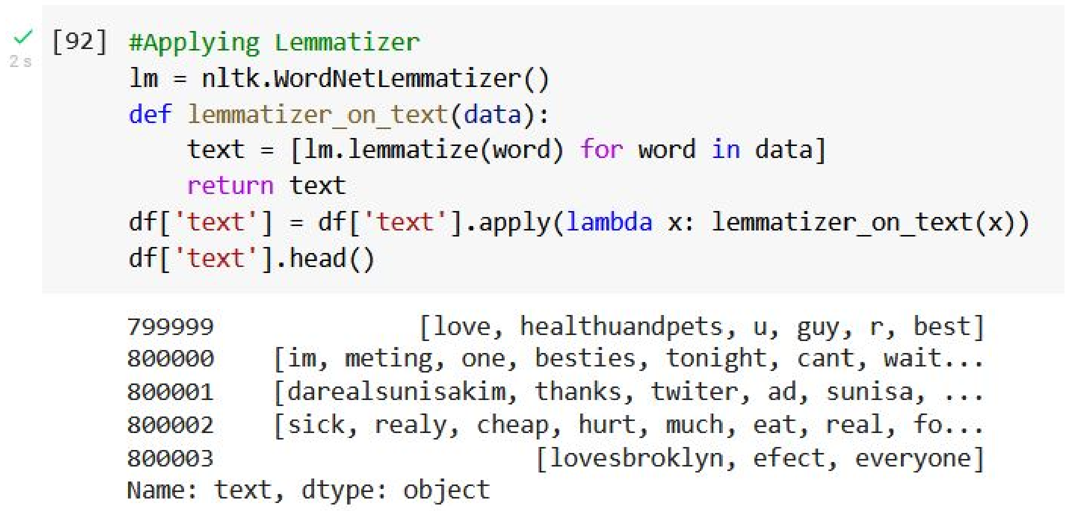
```
[89] #Getting tokenization of tweet text
     tokenizer = RegexpTokenizer(r'\w+')
     df['text'] = df['text'].apply(tokenizer.tokenize)
     df['text'].head()
```

```
799999              [love, healthuandpets, u, guys, r, best]
800000    [im, meting, one, besties, tonight, cant, wait...
800001    [darealsunisakim, thanks, twiter, ad, sunisa, ...
800002    [sick, realy, cheap, hurts, much, eat, real, f...
800003                   [lovesbroklyn, efect, everyone]
Name: text, dtype: object
```

FIGURE 2.15 – Getting tokenization of tweet text

2.3.9 Lemmatization

Both Lemmatization and Stemming are used to reduce words to their base or root form. This is useful for reducing the dimensionality of the data and removing inflectional endings on words. Lemmatization is more sophisticated than stemming as it uses a WordNet dictionary to get the root word, while stemming is based on simple heuristics and may not get the root form of the word.



```
[92] #Applying Lemmatizer
     lm = nltk.WordNetLemmatizer()
     def lemmatizer_on_text(data):
         text = [lm.lemmatize(word) for word in data]
         return text
     df['text'] = df['text'].apply(lambda x: lemmatizer_on_text(x))
     df['text'].head()
```

```
799999               [love, healthuandpets, u, guy, r, best]
800000    [im, meting, one, besties, tonight, cant, wait...
800001    [darealsunisakim, thanks, twiter, ad, sunisa, ...
800002    [sick, realy, cheap, hurt, much, eat, real, fo...
800003                      [lovesbroklyn, efect, everyone]
Name: text, dtype: object
```

FIGURE 2.16 – Applying Lemmatizer

2.4 Conclusion

In this chapter, we presented the dataset used in the project, detailed the Exploratory Data Analysis, and finally explained the different phases of the data preprocessing.

Chapitre 3

Model Definition

3.1 Introduction

In this chapter, we will create, train, and evaluate our own defined model, and then we will tune its hyperparameters in order to optimize it.
Finally, we will implement a few pre-trained models and compare the results.

3.2 Defined Model

3.2.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of neural network architecture that is particularly well suited for processing sequences of data, such as time series data, speech, and natural language. Unlike feedforward neural networks, which only have connections between layers in one direction, RNNs have connections that form a directed cycle, which allows information to flow in a "memory" throughout the network. This allows RNNs to remember information from previous time steps and use it to inform the current output, which is particularly useful for tasks like language modeling and speech recognition.

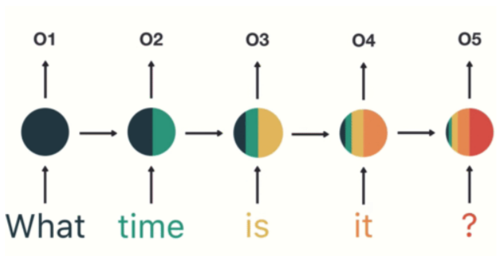


FIGURE 3.1 – RNN.

3.2.2 Long Short Term Memory

Long Short-term Memory (LSTM) is a type of recurrent neural network (RNN) architecture that is particularly well suited for processing sequences of data that have long-term dependencies. LSTMs were developed to address the vanishing gradient problem, which occurs in traditional RNNs when trying to learn long-term dependencies.

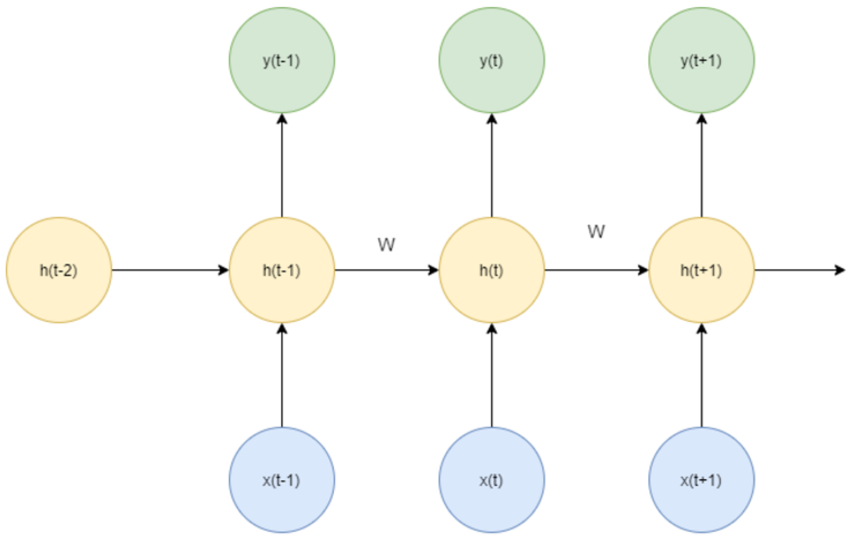
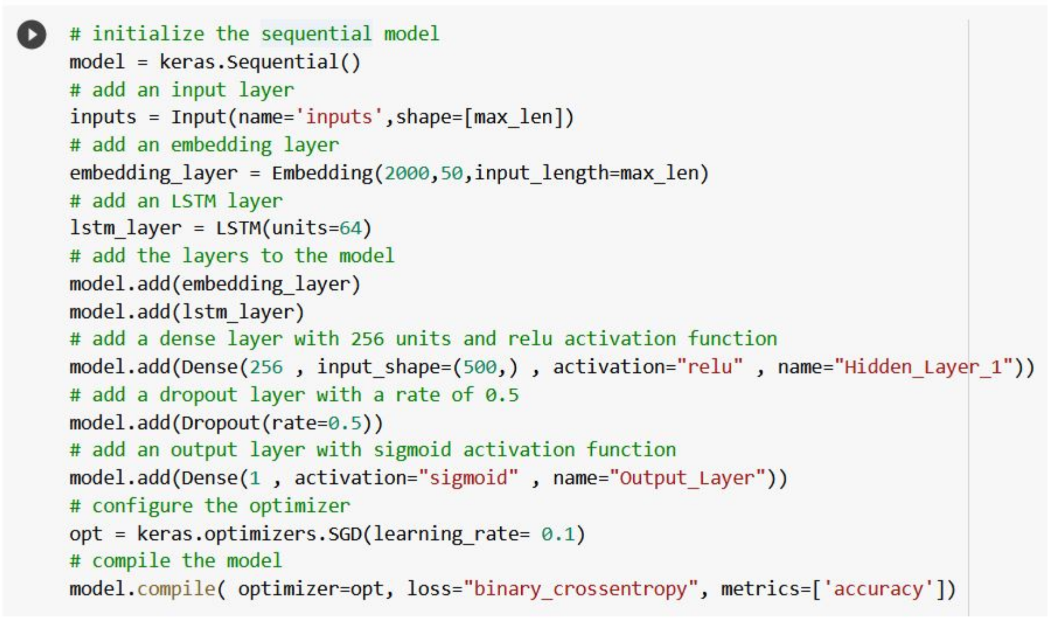


FIGURE 3.2 – LSTM.

3.2.3 Model Architecture

Through the project, several Recurrent neural networks were developed. They were developed with the same architecture, but the parameters for each layer were different. The architecture in figure 3.3 is the architecture used in all models. Besides that, the following parameters are changed in the models : the dropout, the optimizers and the learning rates.



```
# initialize the sequential model
model = keras.Sequential()
# add an input layer
inputs = Input(name='inputs',shape=[max_len])
# add an embedding layer
embedding_layer = Embedding(2000,50,input_length=max_len)
# add an LSTM layer
lstm_layer = LSTM(units=64)
# add the layers to the model
model.add(embedding_layer)
model.add(lstm_layer)
# add a dense layer with 256 units and relu activation function
model.add(Dense(256 , input_shape=(500,) , activation="relu" , name="Hidden_Layer_1"))
# add a dropout layer with a rate of 0.5
model.add(Dropout(rate=0.5))
# add an output layer with sigmoid activation function
model.add(Dense(1 , activation="sigmoid" , name="Output_Layer"))
# configure the optimizer
opt = keras.optimizers.SGD(learning_rate= 0.1)
# compile the model
model.compile( optimizer=opt, loss="binary_crossentropy", metrics=['accuracy'])
```

Figure 3.3 – Model Architecture.

3.3 Pre-trained Model

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained language representation model developed by Google. It is a transformer-based neural network architecture that is trained on a large corpus of text data to learn general-purpose language representations. BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. This makes BERT well-suited for tasks that involve understanding the context of a piece of text, such as sentiment analysis based on the tweets in our case. BERT is available in several pre-trained models, such as BERT-Base and BERT-Large, which have different numbers of layers and parameters. In our project we tried to implement some Bert variance precisely bert-base-uncased, bert-large-uncased, and bert-base-cased.

3.3.1 BERT-Base

BERT-Base is a pre-trained version of the BERT model. It has 12 layers (transformer blocks), 12 attention heads, and 110 million parameters. BERT-Base was trained on a large corpus of text data, including the English Wikipedia and the BooksCorpus dataset, which includes books and other text from a wide range of genres.The main difference between the BERT-Base cased and BERT-Base uncased versions is in the way that the text data was preprocessed before training the model.

3.3.1.1 BERT-Base cased

In the "cased" version, the model was trained on text data that has been lowercased and tokenized, preserving the case of the characters. This means that the model has learned to distinguish between words that have different capitalization, such as "Apple" and "apple".

3.3.1.2 BERT-Base uncased

In the "uncased" version, the model was trained on text data that has been lowercased and tokenized, without preserving the case of the characters. This means that the model has learned to treat words with different capitalization as the same word, such as "Apple" and "apple".

3.3.2 BERT-Large

BERT-Large is a larger version of the pre-trained BERT model. It has 24 layers (transformer blocks), 16 attention heads, and 340 million parameters. BERT-Large was also trained on a large corpus of text data, including the English Wikipedia and the BooksCorpus dataset. And due to its larger size, BERT-Large is able to achieve better performance on some tasks, especially those that require a more powerful model to understand the context of a piece of text.

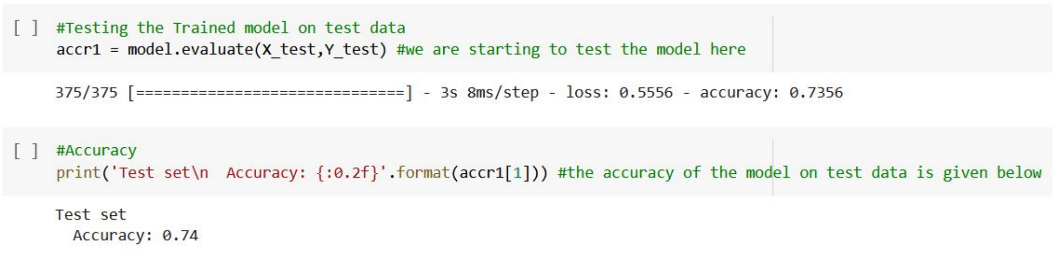
3.4 Models Evaluation

Evaluating a model is an important step in the machine learning process, it allows you to measure the performance of your model and understand how well it is able to generalize to new unseen data. Here are some common ways that we perform to evaluate the RNN and the pre-trained model :

3.4.1 The Defined Model Evaluation

3.4.1.1 Accuracy

This is the most common evaluation metric for classification tasks, it measures the percentage of correct predictions made by the model. It is a good metric when the classes are well-balanced and this is the case.



```
[ ] #Testing the Trained model on test data
    accr1 = model.evaluate(X_test,Y_test) #we are starting to test the model here

    375/375 [==============================] - 3s 8ms/step - loss: 0.5556 - accuracy: 0.7356

[ ] #Accuracy
    print('Test set\n  Accuracy: {:0.2f}'.format(accr1[1])) #the accuracy of the model on test data is given below

    Test set
      Accuracy: 0.74
```

Figure 3.4 – Defined Model Accuracy

3.4.1.2 Confusion Matrix

confusion matrix is a table that is used to define the performance of a classification algorithm. It allows you to see true positive and true negatives, as well as false positives and false negatives.

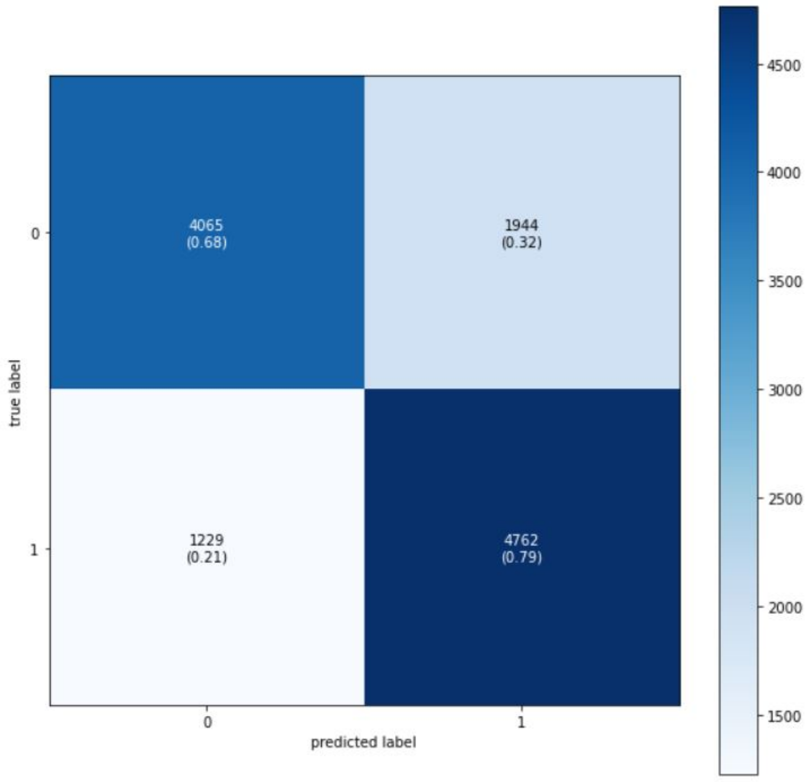


Figure 3.5 – Defined Model Confusion Matrix

3.4.1.3 ROC Curve

ROC curve and AUC are commonly used for binary classification tasks. ROC curve plots the true positive rate against the false positive rate, and AUC measures the area under the ROC curve. AUC is a value between 0 and 1, where a value of 1 represents a perfect classifier and a value of 0.5 represents a random classifier.

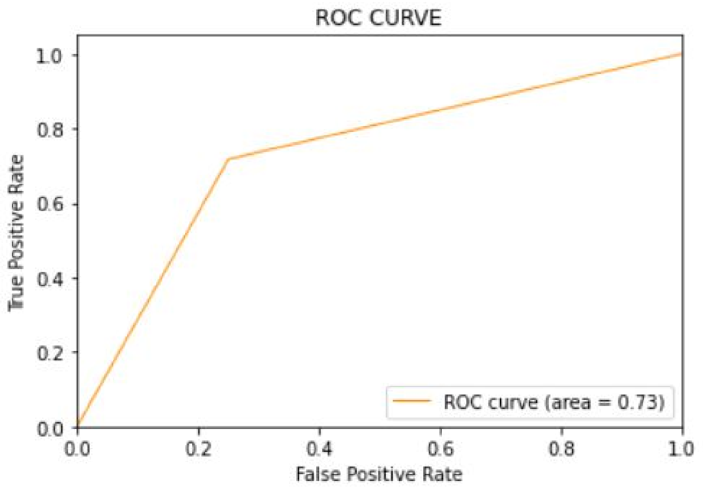


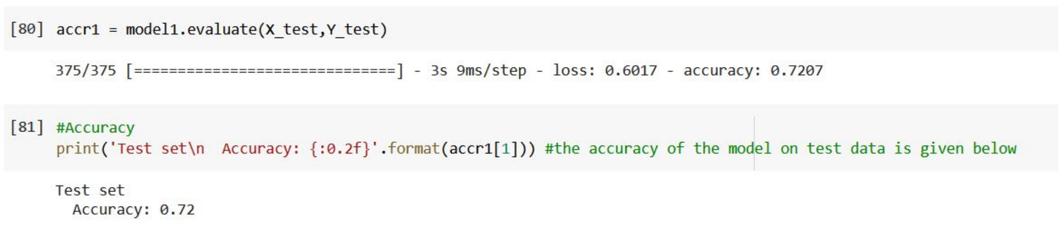
FIGURE 3.6 – Defined Model ROC Curve

3.4.2 The Pre-trained Model Evaluation

Since all the pre-trained variants of BERT, we tried to give very close evaluation results. We will present in the following figures just the evaluation of the BERT-base-uncased model.

3.4.2.1 Accuracy

This figure represents the accuracy of the BERT-base-uncased model :



```
[80] accr1 = model1.evaluate(X_test,Y_test)

     375/375 [==============================] - 3s 9ms/step - loss: 0.6017 - accuracy: 0.7207

[81] #Accuracy
     print('Test set\n  Accuracy: {:0.2f}'.format(accr1[1])) #the accuracy of the model on test data is given below

     Test set
       Accuracy: 0.72
```

FIGURE 3.7 – BERT-base-uncased model accuracy

3.4.2.2 Confusion Matrix

This figure represents the confusion Matrix of the BERT-base-uncased model :

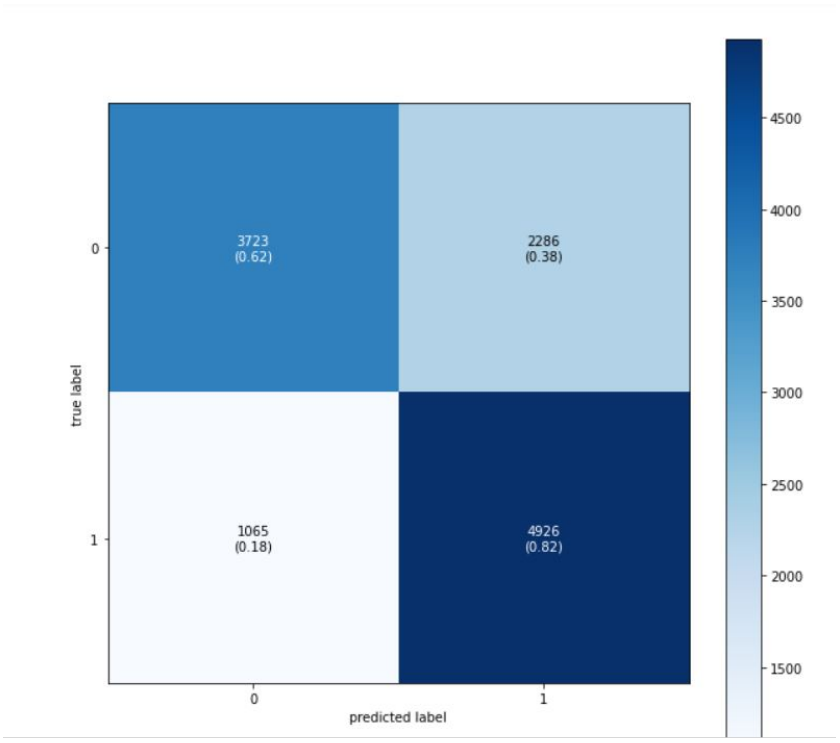


FIGURE 3.8 – BERT-base-uncased model confusion Matrix

3.4.2.3 ROC Curve

This figure represents the ROC Curve of the BERT-base-uncased model :

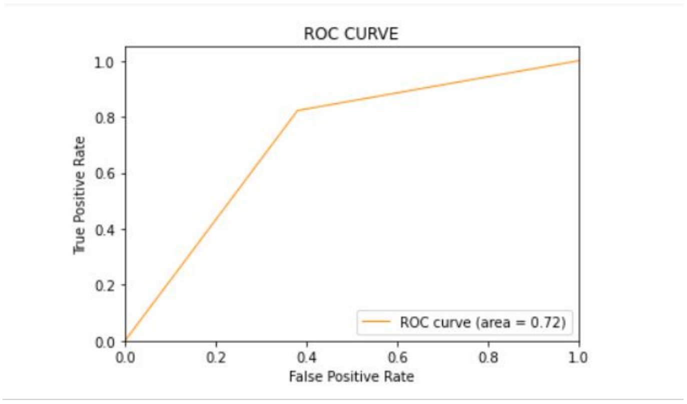


FIGURE 3.9 – BERT-base-uncased model ROC Curve

3.5 Results Interpretation

In this project, It's interesting that the RNN model and the pre-trained BERT model both performed similarly well in terms of accuracy, confusion matrix, and ROC curve for this sentiment analysis project. One possible explanation for this is that the nature of the data we were working with was well-suited to the strengths of both models. The pre-trained BERT model provides a strong understanding of language and context, which could be beneficial for sentiment analysis tasks, while RNNs are known to be good at processing sequential data. It's also worth noting that pre-trained models like BERT are trained on a large amount of diverse data which gives them a strong generalization ability but that doesn't mean that in all cases they will be better than models trained on a smaller dataset. Sometimes, a model trained on a smaller, more specific dataset can be better suited to a particular task, due to the fact that it has seen similar data and learned to generalize in a similar way.

3.6 Conclusion

In this chapter, we presented the deep learning model that we defined, explained its architecture, and then we presented the pre-trained models used in this project, detailed the models' evaluation, and finally interpreted the results of the different models tried out through this project.

General conclusion

In this report, a deep learning model was implemented to perform sentiment analysis, a task that aims to determine the sentiment or emotion expressed in a piece of text. Sentiment analysis is a rapidly growing area of research in natural language processing and text mining. The project began with an overview of the problem and proposed solution, followed by a presentation of the dataset and a detailed exploration of the data. The data was then preprocessed to prepare it for modeling. In the final chapter, the deep learning model was explained and its architecture was presented, along with the pre-trained models used in the project. The models were evaluated and compared, and the results were interpreted. The project was a valuable opportunity to gain experience in the field of natural language processing and deep learning, but there is still room for improvement, such as using the entire dataset and performing a grid search to optimize model hyperparameters.