Customer Support Sentiment Analysis

TEST PROJECT Report

SARIM SIKANDER

2023

acknowledgement

First of all, I would like to give credit to SolvPath for giving me this chance, without their explanation this project will not be a success. Solvpath helped me with the description and providing the dataset. I hereby thanks them and their team.

I would love to mentioned my beloved family who supported me through each step of the way. It’s their continuous support and understanding that have made this project a huge success. And their prayers that have guarded me through every difficulty I have faced. And they have always helped me to be the best of myself.

In the end, I would like to thank God, for letting me pass all the challenges. I have experienced your guidance every day and you are the reason I am able to finish this project. I have always trusted you for my future and I will keep doing it.

Table of Contents

[Introduction of the project: 4](#_Toc134893059)

[1. Background 4](#_Toc134893060)

[2. Aims and Objectives 4](#_Toc134893061)

[3. Operation environment 5](#_Toc134893062)

[4. Ide Used 5](#_Toc134893063)

[5. Language Used 5](#_Toc134893064)

[Data sources 6](#_Toc134893065)

[1. Sentiment analysis texts 6](#_Toc134893066)

[Used libraries: 8](#_Toc134893067)

[1. Pandas 8](#_Toc134893068)

[2. Numpy 8](#_Toc134893069)

[3. NLTK 9](#_Toc134893070)

[4. Seaborn 9](#_Toc134893071)

[5. Sklearn 10](#_Toc134893072)

[6. Keras 10](#_Toc134893073)

[7. Transformer Tensor Flow 10](#_Toc134893074)

[Data preprocessing 11](#_Toc134893075)

[Why Data preprocessing? 11](#_Toc134893076)

[1. Accuracy: 11](#_Toc134893077)

[2. Completeness: 11](#_Toc134893078)

[3. Consistency: 11](#_Toc134893079)

[4. Timeliness: 11](#_Toc134893080)

[5. Interpretability: 12](#_Toc134893081)

[Steps for Data preprocessing 12](#_Toc134893082)

[1. Removing missing values 12](#_Toc134893083)

[2. Textual preprocessing 12](#_Toc134893084)

[3. Contraction 12](#_Toc134893085)

[4. Removing stop words 13](#_Toc134893086)

[5. Tokenization 1](#_Toc134893087)

[6. One hot encoder 1](#_Toc134893088)

[Design and Architecture 1](#_Toc134893089)

[1. RNN 1](#_Toc134893090)

[2. LSTM 1](#_Toc134893091)

[Forget Gate 2](#_Toc134893092)

[Input Gate 2](#_Toc134893093)

[Output Gate 2](#_Toc134893094)

[3. Gru 3](#_Toc134893095)

[Update Gate 4](#_Toc134893096)

[Reset Gate 4](#_Toc134893097)

[Implementation 4](#_Toc134893098)

[Libraries 4](#_Toc134893099)

[Preprocessing 5](#_Toc134893100)

[Train Prework: 8](#_Toc134893101)

[Splitting Dataset 9](#_Toc134893102)

[TRANSFORMERS: 12](#_Toc134893103)

[distilbert-base-uncased-finetuned-sst-2-english 12](#_Toc134893104)

[Findings: 13](#_Toc134893105)

[Deep learning models applied: 14](#_Toc134893106)

[Best Model: 14](#_Toc134893107)

[TRANSFORMERS: 14](#_Toc134893108)

[Conclusion And Future work 16](#_Toc134893109)

[References 17](#_Toc134893110)

# Introduction of the project:

## Background

Us human beings are prone towards the information. Whether it is the information about ourselves to about others or about anything in the world. We have the need to be informed about the current affairs of the world we are living in. In old times this need was fulfilled by newspapers and then the television and then the internet. Internet have become a greater source of news than any other sources.

Everything has a positive and a negative side and as the news industry is growing over the internet, its negative side is also taking a peek. Nowadays any information is just one touch away from our finger tips and relevance of that piece of information is mostly unknown. As there are big authenticated media pages and channels sharing with us the affairs of the world, there are also people sitting behind their computer screen, sharing fake information.

This issue is growing day by day and internet is becoming an untrust worthy source of information. Most of the time the shared information has wrong sentimental impacts on people’s mind since it has not been shared with the best of the intentions.

My thesis is focused on project called “Fake News Identification and Sentiment Analysis”. Following project is derived by the need of stopping irrelevance and hurtful news that surrounds the internet and social media to be particular. We have to identify the if the news is fake and whether it is a negative information.

## Aims and Objectives

As explained earlier the whole idea of this research revolves around the identification of fake and hurtful news revolving around the social media. This project has the potential to be used in many social media platforms to identify the irrelevant information. Below are the main objectives of the project.

* We will use NLP (Natural Language Processing) to find out if the news headlines are true or fake.
* We will be working of Sentimental Analysis to find out the if the said news headline is positive, negative or neutral.

As the social media industry is growing and internet is becoming accessible to more people, this issue grows at an exponential rate. Our ability to perceive the world is dependent of information we have been delivered about the world. During the recent pandemic a lot of fake news have revolved around us that had made is impossible for a lot of people to get the help that they needed. And kind of event take place in many of the world affairs such as elections. The situation worsened as more people have accessibility to social media.

## Operation environment

These are the specifications of the computer that I have used to work on this project.

|  |  |
| --- | --- |
| *System Model* | Hp Pavilion notebook |
| *Total Physical Memory* | 8 GB |
| *Processor* | Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz, 2400 Mhz, 2 Core(s), 4 Logical Processor(s) |
| *OS Name* | Microsoft Windows 10 Pro |
| *System Type* | x64-based PC |
| *BIOS Mode* | UEFI |

## Ide Used

|  |  |
| --- | --- |
| The IDE used in this project is Jupyter Notebook, which is an open source we application. It is specifically designed for the data scientists to create and share documents and integrate live code, equations, computational output and visualizations. There are also formats where you can write explanatory texts which enables a data scientist to create the entire document and explain the complete analysis and the derived results. |  |

## Language Used

|  |  |
| --- | --- |
| Python is the core language used in this project. This project revolves around the genre of Artificial Intelligence in the field of computer science. Python is the most advanced language in today’s world to be used in data science. It provides us tools, libraries, functionalities, methods and much more selection to work with artificial intelligence. |  |

# Data sources

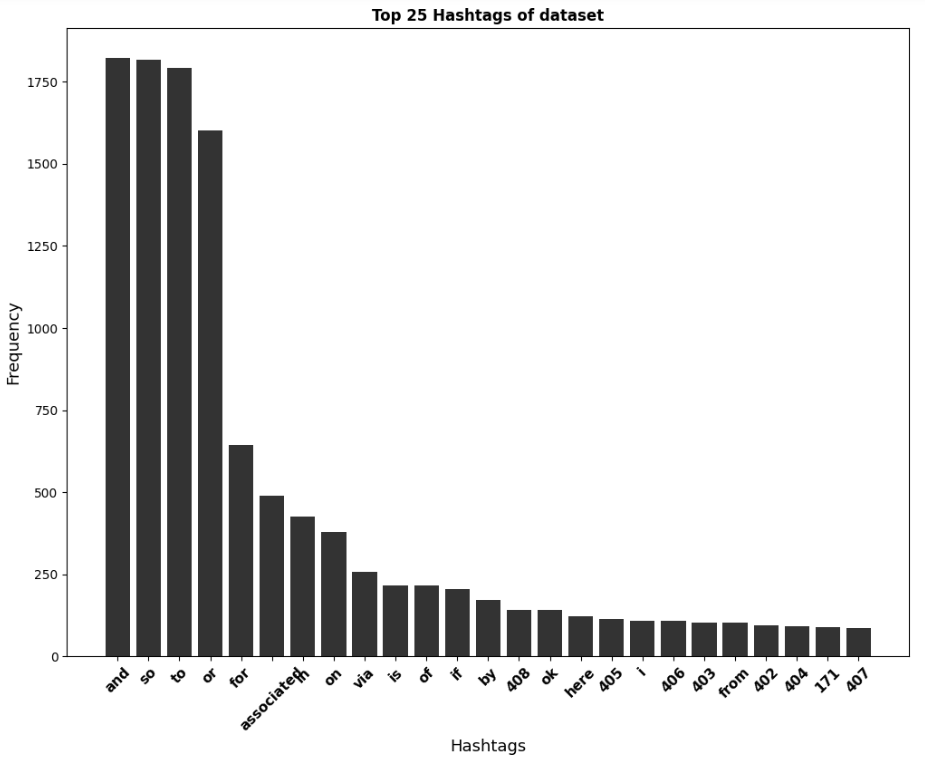
Data set is a collection of data represented in a standard format that is used to conduct studies in various topic. This following research is conducted through machine learning and having the right data set plays a vital role in that. Having the right type of data set is a crucial step of gaining the relevant result a study.

In order to conduct this study, we have used two different types of data sets. Both of the data sets have been extracted from twitters. They are a collection of texts since the twitter is a huge platform to access the information about the current affairs on the internet. There is almost all different kind of texts available. We have used two different bunch of texts.

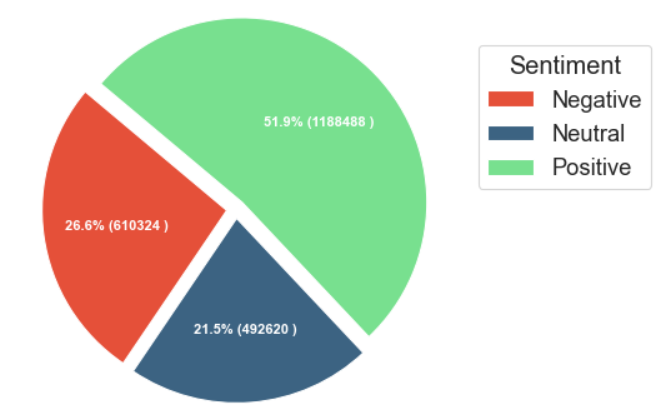
## Sentiment analysis texts

The data set we used is just a collection of random texts. The problem is not just the fake information, it is also the intentions behind that information. A lot of the news travel through social media is not just fake but it is also spread with bad intentions. Identifying the hurtful news is as much of a concern as identifying the service.

Through this study we will identify the sentiment attached to a piece of information and we can achieve that by performing Sentiment Analysis. In machine learning, sentiment analysis is study performed on a data set where we identify with emotion behind a written text and we further classify it into many categories. For our study, we have classified our data into three different categories i.e., positive, negative and neutral.



If the text is positive and give us a brighter perspective, it is a positive text and if the text is negative and hurtful, it is a negative text. Most of the time, a text is not targeting anyone or is not even a positive. And when the intentions behind a text are unclear, we classify is as neutral text. And all this working is done through probabilities. Our model assigns a percentage e.g., this text is 51.9% positive and 26.6% negative and 21.5% neutral. This label of higher probabilities is assigned to the text.



# Used libraries:

As this project has been developed in python, all the libraries we have used are written for python programming language. They have written specifically for machine learning and data analysis.

## Pandas

|  |  |
| --- | --- |
| Panda is a python library which have been solely designed to provide data structure to the raw data. As we receive a data set, we need to convert it into a data structure so they can be used in a structured format. This library has provided us help to a clear look at the data we are working on. Pandas is the base library that has to be used in all python projects. Pandas provide a data structure called Data Frame that converts the data set into a clean tabular form and makes it easier for the compiler to understand the core of the information. |  |

## Numpy

|  |  |
| --- | --- |
| NumPy is used to work with large multi-dimensional data set and arrays in the python projects. We have used this library to derive the results can conclude mathematical functions, random number generations and even algebra routines. This library can help us perform complex mathematical operations on the data set. |  |

## NLTK

|  |  |
| --- | --- |
| NLTK or Natural Language Toolkit is a core library can be used in projects centered around Natural Language Processing or NLP. Since Natural Language Processing is the root of our project, NLTK plays a vital role in all the working stages.  NLTK provided a vast number of methods and functions to work on the project that needs the attention toward textual data. This library provides us the insights on all aspects of the written information making it accessible to work efficiently with Natural Language Processing. Important features provided by NLTK in our project are Porter Stemmer and Word Net Lemmatizer. |  |

## Seaborn

|  |  |
| --- | --- |
| Seaborn is another library that works as a backbone in the field of data science. Through machine learning we perform complex analysis on the data sets and derived complex result that cannot be understand by a not technical person. For that manner, visual representation of the data plays a huge role. Seaborn library provides us a vast number of graphical representation and their methods which plays a vital role in the visual representation of the derived results. |  |

## Sklearn

|  |  |
| --- | --- |
| SK Learn or Scikit-learn is a python library that provide the core feature of machine learning world. It helps us in statistical modeling with many advance models written in python language. It provides us greater tools for machine learning and statistical modeling such as clustering, regression and classification. It provided complex features such as dimensionality reduction via a consistence interface in python.  We have used many core tools of sklearn in this project such as, model selection, metrics such as accuracy score, preprocessing such as standard scalar, on hot encoder, label encoder, functional transformer. Gaussian Naïve Bayes, logistic regression, MLP Classifier, Pipeline and Grid Search CV were also provided through sklearn. |  |

## Keras

|  |  |
| --- | --- |
| As we start to work on the projects that enhances enough to provide prediction using data, deep learning is involved. As the name suggests, deep learning provides us greater insights on the provided data and help the model to train according to the truer aspects of it. Deep learning heavily relies on neural network.  Keras is a library that is specifically designed to work with artificial neural networks. In our project we have used many models written in Keras such as LSTM, GRU, Simple RNN. Keras have also provided us methods and tools to work with data preprocessing, decomposition, model selection and metrics etc. Another important role that Keras has played is the Tokenizer. Tokenizer plays a huge part in data preprocessing. |  |

## Transformer Tensor Flow

|  |  |
| --- | --- |
| Transformer is a model that specializes in the genre of neural networks such as, RNNs or CNNs. It is provided by the library or tensor flow. Tensor flow is a python library that is built for the Deep Learning especially Neural Networks. It is just like Keras but provides different tools to work with. Transformer model is the most used model since it makes no assumptions about the temporal/spatial relationship across the data. The features we have used from transformers are Distil Bert Tokenizer and TF Auto Model which creates and output of vector token that is wildly used in Natural Language Processing. |  |

# Data preprocessing

Preprocessing is the initial working that is performed as a preparation before the actual work. In every genre of data science, data preprocessing is a vital phase that our data set has to go through. In order to receive the most accurate outcome from our data, we have to perform some initial working on it. Data preprocessing get rid of all the unwanted data, noises, outliers, missing data and all the data records that might affect the performance of our data models and provide us biased and in accurate errors.

## Why Data preprocessing?

As mentioned above, data preprocessing helps us provide a neat structure to the data. Along with that it shapes that data according to our needs. It is a crucial phase in any machine learning process as data preprocessing directly impacts the success rate of the project.

Data preprocessing reduces the complexity of the data under analysis. A dataset is said to be unclean if there is missing attribute, attribute values, or it contains noise or outliers and duplicate or wrong data. Presence of any of the factor will degrade the quality of the results. Data preprocessing assures us about the quality of the data. Following are the few motives that derives the need for data preprocessing;

### Accuracy:

It identifies whether the data is correct or not.

### Completeness:

It checks whether the data is available or not recorded.

### Consistency:

Data preprocessing checks whether the same data is kept in all the places that do or do not match.

### Timeliness:

It ensures that the data should be updated correctly

### Interpretability:

It makes data easier to understand.

## Steps for Data preprocessing

In machine learning, when you are working with NLP, you mostly deal with textual data. In NLP, data preprocessing is focused on the textual preprocessing and it is the first step in the process of building model. Here the steps of data preprocessing that we followed,

### Removing missing values

In python, when working with libraries like Pandas, NumPy or Scikit Learn, we mark missing value as NaN. NaN values are ignored from the operations like sum, count, etc. Pandas provide multiple functions that helps us to remove missing values from the data set. It can be performed in four most common ways.

* Removing the entire row with a missing value.
* Removing the entire column with a missing value.
* Replacing the missing value with the value above.
* Replacing the missing value with the value below.

Fortunately, in our dataset, the count of missing value was zero. But still, it is a major part to check for missing data since it can affect badly on the results of our analysis.

### Textual preprocessing

After textual data is obtained, we start with normalizing the text. Textual normalization includes the following steps:

* Converting all letters to lower or upper case or upper to lower case.
* Converting all numbers into words or removing numbers.
* Removing punctuations, accent marks and other diacritics etc.
* Removing all the white spaces.
* Expanding every abbreviation.
* Removing stop words, particular words, and sparse terms.

### Contraction

Text processing plays a vital role in NLP. Cleaning our textual data in order to convert it into a presentable form that is analyzable and predicable for our task is known as textual preprocessing and contraction is a crucial part of this process.

Contractions are words or combinations of words that are shortened by missing letters from dropping the letters or replacing them with apostrophe. In this era, data extraction from online platforms such as twitter can make process of recontractions important.

In this project we have performed the step of decontraction by finding out the contracted words from the dataset and replacing them with the decontracted words. Here is the list of all the contracted words that we replaced in this project.

|  |  |
| --- | --- |
| **CONTRACTED** | **DECONTRACTED** |
| *Specific* | |
| Won’t | Will not |
| Can’t | Can not |
| *General* | |
| n’t | Not |
| ‘re | are |
| ‘s | is |
| ‘d | would |
| ‘ll | Will |
| ‘t | Not |
| ‘ve | Have |
| ‘m | Am |

### Removing stop words

Stop words are often removed from the text before training the textual data into deep learning and machine learning models since stop words occurs in abundance, hence providing little to no unique information that can be used for clustering or classification.

Removing stop words is a necessary step when working with NLP since it will provide the only information that we need. This phase will help us to achieve the best outcome from our deep learning models. Not ignoring the fact that less data will help us to fasten up the process.

Here is a list of all the stop words we have removed from our data set,

the

i

me

my

myself

we

our

ours

ourselves

you

you're

you've

you'll

you'd

your

yours

yourself

yourselves

he

him

his

himself

she

she's

her

hers

herself

it

it's

its

itself

they

them

their

theirs

themselves

what

which

who

whom

this

that

that'll

these

those

am

is

are

was

were

be

been

being

have

has

had

having

do

does

did

doing

a

an

the

and

but

if

or

because

as

until

while

of

at

by

for

with

about

against

between

into

through

during

before

after

above

below

to

from

up

down

in

out

on

off

over

under

again

further

then

once

here

there

when

where

why

how

all

any

both

each

few

more

most

other

some

such

only

own

same

so

than

too

very

s

t

can

will

just

don

don't

should

should've

now

d

ll

m

o

re

ve

y

ain

aren

aren't

couldn

couldn't

didn

didn't

doesn

doesn't

hadn

hadn't

hasn

hasn't

haven

haven't

isn

isn't

ma

mightn

mightn't

mustn

mustn't

needn

needn't

shan

shan't

shouldn

shouldn't

wasn

wasn't

weren

weren't

won

won't

wouldn

wouldn't

### Tokenization

Tokenization is an essential task that a data scientist comes across when working with text data. It is consisting of splitting an entire text into smaller units, also called tokens. Most of NLP projects have the process of tokenization as the first step since it is the foundation for developing good models and help us to better understand the insights of the textual data we have.

In our project, we have tokenized the words through tokenizer from Keras library. Keras provide a wide number of methods and functions for data preprocessing. It even contains a whole module of text preprocessing. With Keras tokenizer, we have tokenized the words and settled them into a json dictionary with max features up to 20000. After tokenizing the words, we have also set them up in a sequence from text to sequence function of Keras tokenizer.



### One hot encoder

Most of the deep learning algorithms cannot be executed on categorical data. Instead, firstly, the categorical data needs to be converted into numerical data. One hot encoding is a technique used to perform this conversion on textual data. This method is mostly used when machine learning techniques are to be applied to sequential classification problems.

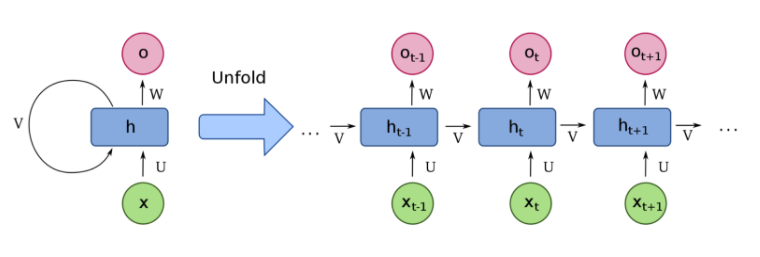
One hot encoding is basically the representation of categorical variables as binary vectors. These categorical values are firstly to be mapped to integer values. Each integer value is then represented as a binary vector. In our project, we have tokenized both data set i.e., fake news data set and text data set. After the data sets have been tokenized, all the tokens will be entered in the model for the further evaluation.

# Design and Architecture

## RNN

Traditional neural networks are not capable of storing data from the past. They rely on the current information and takes decision based on the current data. You can say that they rely on their short-term memory. That’s when the Recurrent Neural Networks (RNN) comes to the rescue. In RNN, networks have loops in them that allows them to persist the information. A recurrent neural network (RNN) can be thought of as multiple copies of the same network where each network is passing a message to a successor.

This chain-like nature reveals that RNNs are intimately related to all sequences and lists. RNNs are the natural architecture of neural network to use for such kind of data. RNNs are the state-of-the-art algorithm for sequential data and they are used by great technologies such Apple’s and Google’s voice search. RNN is the first algorithm that remember its previous input due to an assigned internal memory that makes RNN perfectly suited for machine learning problems that involve sequential data. RNN is a powerful and robust neural network and it belongs to the most promising algorithms in use since it is the only one with an internal memory.



RNN are special architectures that take into account temporal information. The hidden state of this network at a time can takes in information from both the input at time t and activations from hidden units at time t-1, in order to calculate outputs for time t. The above image represents this cycle. This provides the memory of RNN, an ability to remember the previous inputs and their outputs.

This process is extremely important for NLP, as in NLP the input data doesn’t have a fixed size and the next data is dependent on previous data. Context is crucial factor in NLP. The size of each sentence can be different as well as the input they produce. Hence this ability to accept input sizes and compute outputs of different size is highly beneficial, and this capability can be provided by RNNs. RNNs helps to build context with the data.

## LSTM

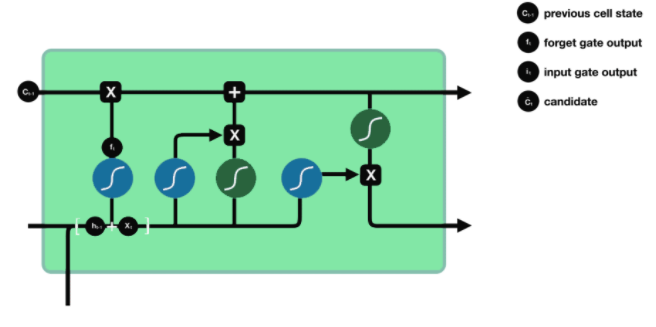
LSTM or Long-Short Term Memory is a type of Recurrent Neural Network (RNN). We can say that it is a better version of traditional RNN in terms of memory. LSTM have a good hold over memorizing certain patterns where it performs certainly better than traditional RNN. As every other Neural Network, LSTM can have multiple hidden layers and the network passes through every layer, the relevant information is kept safe while all the irrelevant information gets discarded.

If we talk about working of LSTM, LSTM has three main gates,

* FORGET gate
* INPUT gate
* OUTPUT gate

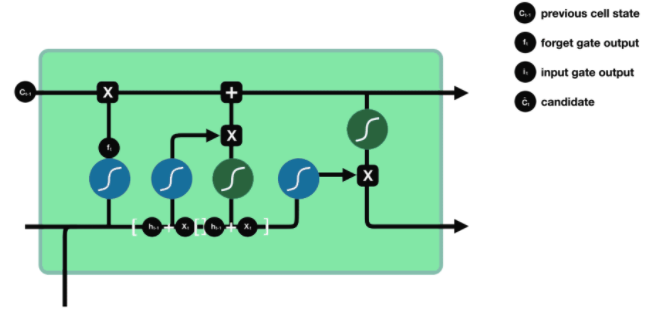
### Forget Gate

This gate decides which information should be kept for calculating the cell state and which information is irrelevant and should be discarded.



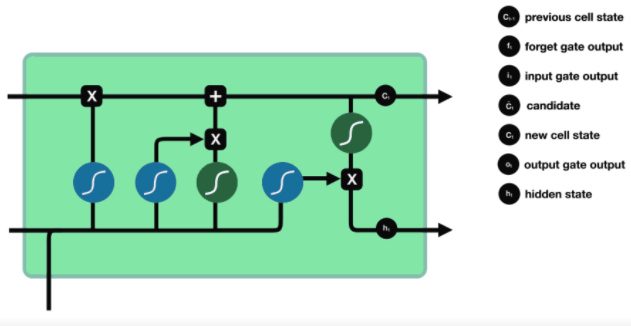
### Input Gate

This gate updates the cell state and decides which information is relevant and which is not. As forget gate get to decide which information to discard, input gate helps to find out which information is important and should be kept safe.



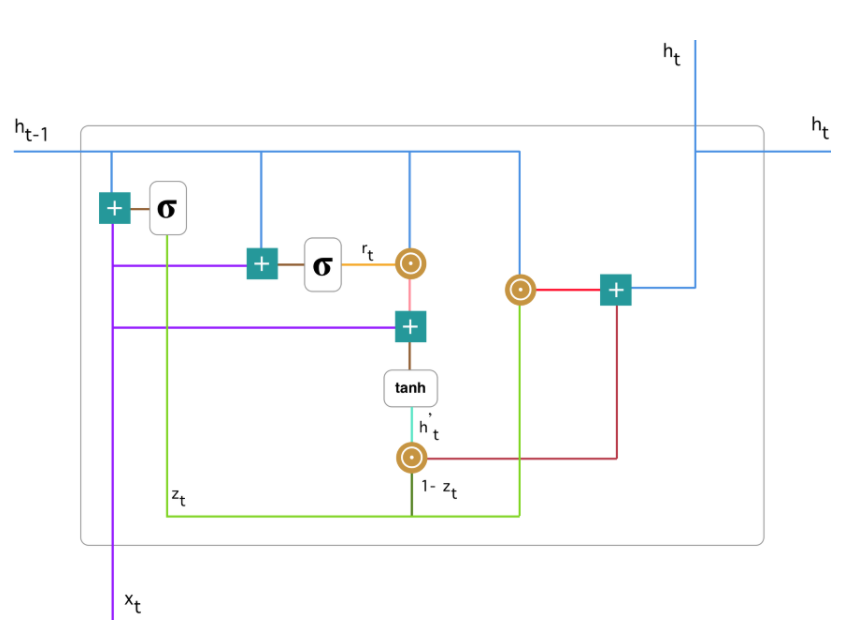
### Output Gate

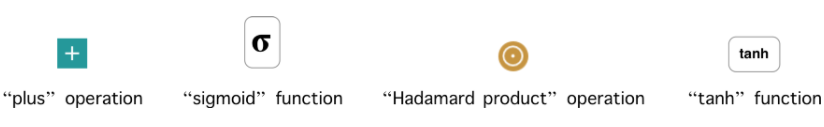
Output gate decided which state should be the next hidden state.



## Gru

Gated Recurrent Unit or GRU can be considered as a variation on the LSTM because both of these algorithms are designed similarly and, in some cases, they both produce equally excellent results. GRUs are an improved version on traditional RNN just like LSTM. But they come more helpful when you have to solve Vanish Gradient problem. GRU have two new gates called UPDATE gate and RESET gate. These two gates get to decide which information should be passed on to the output. Their specialty is that they can be trained to keep information from past without changing it through time or remove any information that is irrelevant to the prediction.





### Update Gate

This gate helps the model to decide how much information of the past needs to be passed along the present one. This phenomenon makes the GRU algorithm much powerful because the model can decide to copy all the information from the past and eliminate the risk of vanishing gradient problem.

### Reset Gate

Reset gate is used by the model to decide how much of the past to forget. Just like update gate gets to decided relevance of an information, reset gate decide the irrelevance.

# Implementation

## Libraries

import pandas as pd

import numpy as np

import nltk

import string

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler,OneHotEncoder,LabelEncoder

from collections import Counter

from keras.models import Sequential

from keras.layers.recurrent import LSTM, GRU,SimpleRNN

from keras.layers.core import Dense, Activation, Dropout

from keras.layers.embeddings import Embedding

from keras.layers import BatchNormalization

from keras.utils import np\_utils

from sklearn import preprocessing, decomposition, model\_selection, metrics, pipeline

from keras.layers import GlobalMaxPooling1D, Conv1D, MaxPooling1D, Flatten, Bidirectional, SpatialDropout1D

from keras.preprocessing import sequence, text

from keras.preprocessing.text import Tokenizer

from keras.callbacks import EarlyStopping

import tensorflow as tf

import transformers

from transformers import DistilBertTokenizer

from transformers import TFAutoModel

import re

import string

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer

from nltk.stem.wordnet import WordNetLemmatizer

from bs4 import BeautifulSoup

import re

%matplotlib inline

import warnings

from tqdm import tqdm

import os

warnings.filterwarnings("ignore")

oneE = OneHotEncoder(sparse=False)

All the work we performed was because of these libraries. Models are made with the help of Sklearn, Tensorflow and Transformers library. After training the model we visualized the model performance with the help of matplotlib and seaborn library which gave a better understanding.

!pip install transformers

Transformers (formerly known as pytorch-transformers and pytorch-pretrained-bert) provides general-purpose architectures (BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet) for Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over 32+ pretrained models in 100+ languages and deep interoperability between Jax, PyTorch and TensorFlow.

Transformer’s library isn’t preinstalled so we need to use pip package to download this library and work with encoder-decoder models. These models are usually the best models to perform on any dataset. These pretrained models are trained on large corpus which helped to fine tune it later with any similar dataset and get better accuracy and results.

## Preprocessing

sentiment = pd.read\_csv('Sentiment.csv’)

sentiment['sentiment'].isnull().sum()

for sent in range(len(sentiment['sentiment'].values)):

if sentiment['sentiment'][sent] == 0.0:

sentiment['sentiment'][sent] = 1.0

sentiment['sentiment'].value\_counts()

First step when building a model is to preprocess and check the labels as well as features to select and de select. In this model we have used two different datasets, one is sentiment and another is fake news. Both these datasets and taken from different sources to collect different types of labels and situations.

|  |  |  |  |
| --- | --- | --- | --- |
| Datasets | Labels | Length | Data type |
| Sentiment | **Positive (2)** | **12900** | **Integer64** |
|  | **Neutral (1)** | **41500** | **Integer64** |
|  | **Negative (-1)** | **5300** | **Integer64** |

Upper table shows the number of labels and Their length in dataset individually with data types. In this dataset we get no null values that’s why dataset was already cleaned and ready to use for feature engineering and feature extraction.

def decontracted(phrase):

# specific

phrase = re.sub(r"won't", "will not", phrase)

phrase = re.sub(r"can\'t", "can not", phrase)

# general

phrase = re.sub(r"n\'t", " not", phrase)

phrase = re.sub(r"\'re", " are", phrase)

phrase = re.sub(r"\'s", " is", phrase)

phrase = re.sub(r"\'d", " would", phrase)

phrase = re.sub(r"\'ll", " will", phrase)

phrase = re.sub(r"\'t", " not", phrase)

phrase = re.sub(r"\'ve", " have", phrase)

phrase = re.sub(r"\'m", " am", phrase)

return phrase

Decontraction in Feature preprocessing is a very important part of model to perform well, these contracted words which are popularly used in the English. These words aren’t understandable by the computer/machine language.

We import the data and try to clean it as much as possible, with minimum loss of meaning. We're using a deep learning model we don't need to clean a lot of words out, because the deep learning model can find out about each word itself even it's never seen it before in its vocabulary

stopwords= set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\

"you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \

'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\

'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \

'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \

'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \

'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\

'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\

'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\

'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \

's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \

've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\

"hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\

"mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \

'won', "won't", 'wouldn', "wouldn't"])

def preprocess(data):

preprocessed\_reviews = []

for sentance in tqdm(data):

sentance = re.sub(r"http\S+", "", sentance)

sentance = BeautifulSoup(sentance, 'lxml').get\_text()

sentance = decontracted(sentance)

sentance = re.sub("\S\*\d\S\*", "", sentance).strip()

sentance = re.sub('[^A-Za-z]+', ' ', sentance)

sentance = ' '.join([word for word in sentance.split() if len(word) > 1])

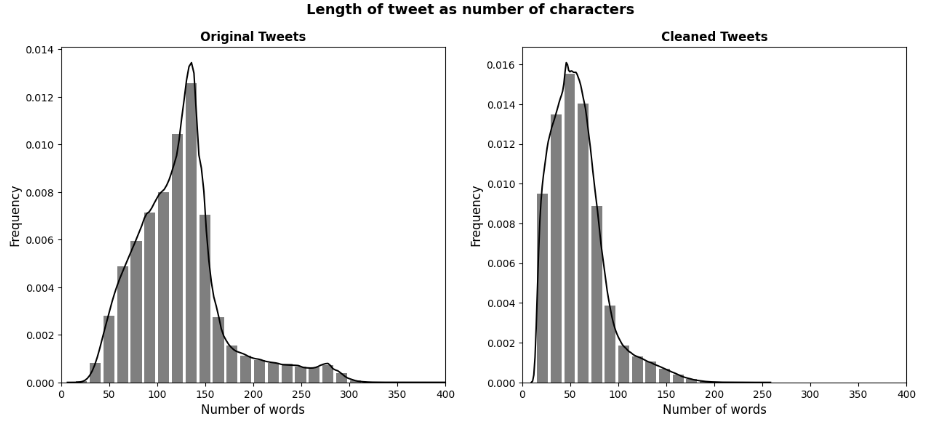
sentance = ' '.join(e.lower() for e in sentance.split() if e.lower() not in stopwords)

preprocessed\_reviews.append(sentance.strip())

return preprocessed\_reviews

sentiment\_text = preprocess(sentiment['text'].values)

Then we did text preprocessing for more cleaning of the text so that no stop words those have no meaning in sentiment can be removed as well as we removed all the html tags with beautiful soup library.



Beautiful soup is the library in python that uses Parser to parse the html of the web and scrap different information from the tags. After this we used Regular expressions library to provide some more cleaning in text like removing any word without any sense or number in between, like dates or contact numbers.

import json

def tokenize(data,max\_features=20000):

max\_fatures=max\_features

tokenizer = Tokenizer(num\_words=max\_fatures)

tokenizer.fit\_on\_texts(data)

dictionary = tokenizer.word\_index

with open('wordindex.json', 'w') as dictionary\_file:

json.dump(dictionary, dictionary\_file)

return tokenizer,dictionary,max\_fatures

Tokenization function takes in data and max features to convert sentences into words and then words into Index of the word. Each word is given a separate index that shows the individuality of the word and every index represent single word which helps the machine to understand which index belongs to which word. These are many tokenization techniques but the technique we have used is from Keras library, because the model we will be using throughout the Starting will be deep learning models from tensor flow.

tokenizer,dictionary,max\_features = tokenize(X)

def text\_to\_seq(tokenizer,CodeSnippet,seq=200):

X = tokenizer.texts\_to\_sequences(CodeSnippet)

X = sequence.pad\_sequences(X,seq)

return X

X = text\_to\_seq(tokenizer,X)

def one\_hot\_encoder(y,inverse=False):

if inverse:

Y = oneE.inverse\_transform(y)

else:

y = np.array(y)

y = y.reshape(-1,1)

Y = oneE.fit\_transform(y)

return Y,one

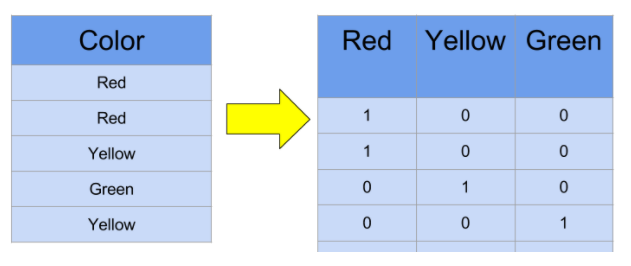
Y,oneE = one\_hot\_encoder(y)

## Train Prework:

* Use to categorical to encode train Label (target)
* Train split to train and validation
* Keras tokenizer initialization
* num and max length
* transfer to Keras tokenizer

One hot encoding is the most widespread approach, and it works very well unless your categorical variable takes on a large number of values (i.e., you generally won't it for variables taking more than 15 different values. It'd be a poor choice in some cases with fewer values, though that varies.)

One hot encoding creates new (binary) columns, indicating the presence of each possible value from the original data. Let's work through an example.



The values in the original data are *Red*, *Yellow* and *Green*. We create a separate column for each possible value. Wherever the original value was *Red*, we put a 1 in the *Red* column.

So far, you've one-hot-encoded your training data. What about when you have multiple files (e.g., a test dataset, or some other data that you'd like to make predictions for)? Scikit-learn is sensitive to the ordering of columns, so if the training dataset and test datasets get misaligned, your results will be nonsense. This could happen if a categorical had a different number of values in the training data vs the test data.

## Splitting Dataset

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=0)

Xtrain, Xtest, ytrain, ytest = train\_test\_split(X\_, Y\_, test\_size=0.2, random\_state=0)

Now we are splitting dataset into Train and test. We gave 0.2 ratio for test dataset. And random state of 0 that will keep the data constant throughout the notebook every time it is ran.

word\_index = tokenizer.word\_index

max\_len = 200

model = Sequential()

model.add(Embedding(len(word\_index) + 1,

300,

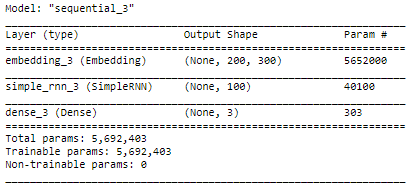
input\_length=max\_len))

model.add(SimpleRNN(100))

model.add(Dense(Y.shape[1], activation='sigmoid'))

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()



model = Sequential()

model.add(Embedding(len(word\_index) + 1,

300,

weights=[embedding\_matrix],

input\_length=max\_len,

trainable=False))

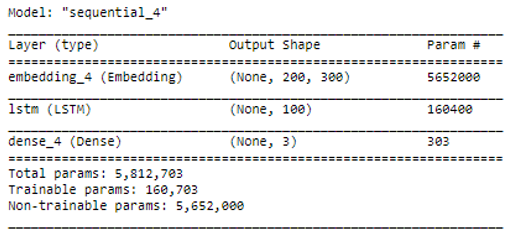
model.add(LSTM(100, dropout=0.3, recurrent\_dropout=0.3))

**model.add(**Dense(Y.shape[1], activation='sigmoid'))

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()

history = model.fit(X\_train, y\_train,validation\_data=(X\_test,y\_test),epochs=10, batch\_size=64)



model = Sequential()

model.add(Embedding(len(word\_index) + 1,

300,

weights=[embedding\_matrix],

input\_length=max\_len,

trainable=False))

model.add(SpatialDropout1D(0.3))

model.add(GRU(300))

model.add(Dense(Y.shape[1], activation='sigmoid'))

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()

history = model.fit(X\_train, y\_train,validation\_data=(X\_test,y\_test),epochs=10, batch\_size=64)

model = Sequential()

model.add(Embedding(len(word\_index) + 1,

300,

weights=[embedding\_matrix],

input\_length=max\_len,

trainable=False))

model.add(Bidirectional(LSTM(300, dropout=0.3, recurrent\_dropout=0.3)))

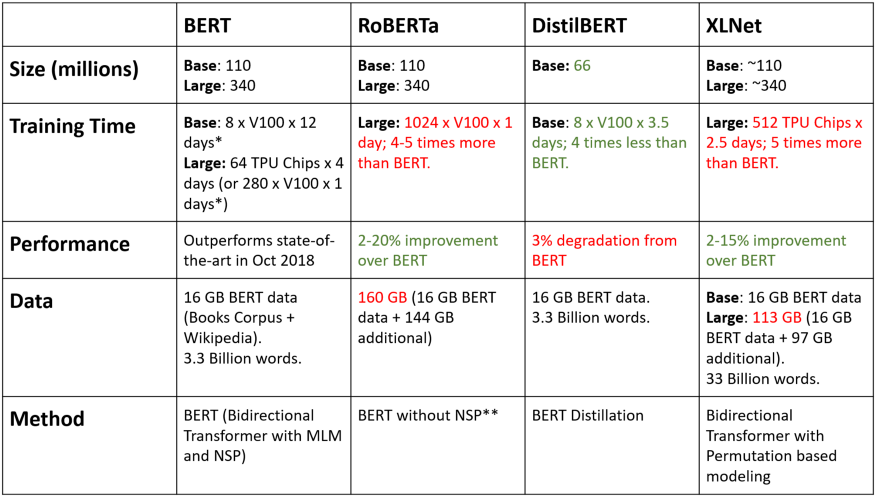
model.add(Dense(Y.shape[1], activation='sigmoid'))

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()

history = model.fit(X\_train, y\_train,validation\_data=(X\_test,y\_test),epochs=10, batch\_size=64)

Here I used deep learning models like RNN, LSTM and bidirection models. Neither of them performed so well that BERT did. So here is the explanation of BERT.



Transformers brings all these models together and makes it very easy to use each with only a few lines of code. In fact they even provide us with cool tools like [pipelines](https://huggingface.co/transformers/main_classes/pipelines.html) or [live demo](https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english?text=I+like+you.+I+love+you) that we can classify our text without any training or long periods of coding. But as you can guess these simple and ready to use models have their weaknesses. For example, you can't classify the text with them with the number of labels you want because they've been pretrained on a text with specific labels. Also, not all models used by them are as strong and accurate as we want them to be (for example the default model for sentiment analysis is uncased distillbert which is not the best model we can find out there). With all these in mind, we want to train. Transformers models on our own data with the models that we prefer.

## TRANSFORMERS:

### distilbert-base-uncased-finetuned-sst-2-english

tokens = prep\_data('global climate is not good')

probs = bert\_model.predict(tokens)

pred = np.argmax(probs)

pred = label\_encoder.inverse\_transform([pred])

results = []

names = []

std= []

for i in grid\_search.cv\_results\_['split0\_test\_score']:

results.append(i)

for i in grid\_search.cv\_results\_['param\_clf']:

names.append(str(i))

for i in grid\_search.cv\_results\_['std\_test\_score']:

std.append(int(i))

names[0] = 'Pipeline With Transformers'

# results.append(grid\_search.cv\_results\_['split0\_test\_score'])

# names.append(grid\_search.cv\_results\_['param\_clf'])

# std.append(grid\_search.cv\_results\_['std\_test\_score'])

fig = plt.figure(figsize=(8,5))

fig.suptitle('Algorithm Comparison')

ax = fig.add\_subplot(111)

barlist = plt.bar(names,height=results)

plt.xlabel('Models Applied')

plt.ylabel('Model accuracy')

plt.xticks(rotation=60)

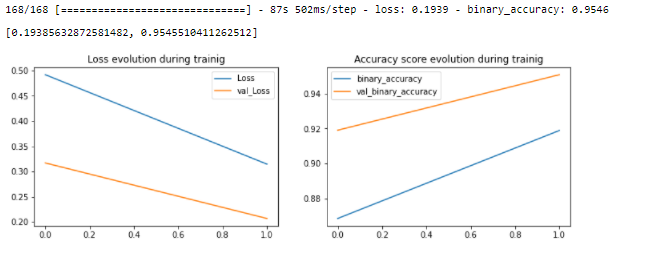
barlist[0].set\_color('r')

barlist[1].set\_color('g')

barlist[2].set\_color('b')

barlist[3].set\_color('purple')

plt.show()



Here are some visualization showing the performance of the model and the loss of the model. You can see the loss is gradually decreasing and accuracy has increased magnificently. This shows that model performed very well. As well as either Overfit nor underfit is there because accuracies are close to each other and giving the same accuracy on the training set and test set this shows that the information model learned and performed is same.

### Findings:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Model* | *Dataset* | *Accuracy* | *Val Accuracy* | *Loss* | *Val Loss* | *Epochs* |
| *SimpleRNN* | **Sentiment** | **97.80%** | **60.49%** | **0.0575** | **0.4175** | **50 (100)** |
| *Lstm+WE* | **Sentiment** | **84.97%** | **83.90%** | **0.0871** | **0.0871** | **30 (50)** |
| *Bidirectional* | **Sentiment** | **92.5%** | **93.08%** | **0.0108** | **0.0108** | **40 (70)** |
| *Lstm+CNN* | **Sentiment** | **99.88%** | **63.08%** | **0.0108** | **0.0108** | **50 (70)** |
| *BERT* | **Sentiment** | **98.18%** | **98.17%** | **0.0308** | **0.0302** | **5** |

Every model we used here performed very well on veracity dataset for being the easiest for all models to process. Hence sentiment didn’t get much acceptable results but due to the effort of every model it got up to 87% with LSTM+WE. Although we can clearly see some models are under fitting because of its accuracy and Val accuracy. It is mostly because of the length of data as well as classes of the data. Whenever the classes of the data are too unbalanced their accuracies get either under or over fit. Well for me all models with deep learning performed very well and for my side they could go better if we train them longer on GPU.

After training Bert all models just lost its battle as the BERT took highest accuracy all over.

### Deep learning models applied:

* Simple RNN
* LSTM + Word Embedding
* LSTM+GRU
* LSTM + Bidirectional
* LSTM+CNN-1D

## Best Model:

### TRANSFORMERS:

Distilbert-base-uncased-finetuned-sst-2-english

bert\_model = tf.keras.Model(inputs=[input\_ids, mask], outputs=y)

# freeze bert layers

# bert\_model.layers[2].trainable = False

optimizer = tf.keras.optimizers.Adam(lr=1e-5)

loss = tf.keras.losses.CategoricalCrossentropy()

acc = tf.keras.metrics.BinaryAccuracy()

bert\_model.compile(optimizer=optimizer, loss=loss, metrics=[acc])

history = bert\_model.fit(

train\_ds,

validation\_data=val\_ds,

epochs=2,

batch\_size=batch\_size

)

def plot\_learning\_evolution(r):

plt.figure(figsize=(12, 8))

plt.subplot(2, 2, 1)

plt.plot(r.history['loss'], label='Loss')

plt.plot(r.history['val\_loss'], label='val\_Loss')

plt.title('Loss evolution during trainig')

plt.legend()

# Conclusion And Future work

This project definitely holds a greater scope for the future. It has much potential to be used in many social media platforms. From where I stand, we can do a lot more work to produce much more accurate results from the algorithm. We can work to increase the accuracy decrease the run time. In my opinion, the program is fast enough for the current usage.

Furthermore, we can work on variety of language to implement it in different parts of the world. This study is not just limited for the English language as the issue stand in place for all the other languages in the world. We can work on different language and increase the scope.

This project can be converted into an API that can be implemented on different social media platform where it will help us to remove hurtful and fake news. Social media plays a vital role in the problem that we are trying to remove and that is why we should start our implementation with the root cause.

Overall, the project was a success. We received desired amount of accuracy and exactly wanted results. The speed was as well moderate enough for the scope of this study. We can see this project rise up to decrease the negativity in internet world. Spreading of the fake news and hurtful content is a rising issue and from where I stand, we have taken a big step to decrease it.

# References

<https://towardsdatascience.com/3-neural-network-architectures-you-need-to-know-for-nlp-5660f11281be>

<https://www.analyticsvidhya.com/blog/2021/06/lstm-for-text-classification/>

<https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be>

<https://humanisingautonomy.medium.com/design-methodologies-for-deep-learning-fdbc160deff7#:~:text=The%20main%20power%20of%20deep,of%20a%20user%2C%20in%20detail>.

<https://www.digitalengineering247.com/article/deep-learning-and-design-engineering>

<https://searchenterpriseai.techtarget.com/definition/machine-learning-ML#:~:text=Machine%20learning%20(ML)%20is%20a,to%20predict%20new%20output%20values>.

<https://en.wikipedia.org/wiki/Fake_news>

<https://guides.lib.umich.edu/fakenews>

<https://www.webwise.ie/teachers/what-is-fake-news/>

<https://au.reachout.com/articles/dealing-with-bad-world-news>

<https://towardsdatascience.com/natural-language-processing-nlp-for-machine-learning-d44498845d5b#:~:text=NLP%20is%20a%20field%20in,Gmail%20structures%20events%20from%20emails>).

<https://machinelearningmastery.com/natural-language-processing/>

<https://towardsdatascience.com/natural-language-processing-nlp-for-machine-learning-d44498845d5b>

<https://www.datacamp.com/tracks/deep-learning-for-nlp-in-python>

<https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270>

<https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/>

<https://www.ibm.com/cloud/learn/recurrent-neural-networks>

<https://searchenterpriseai.techtarget.com/definition/recurrent-neural-networks>