SENTIMENT ANALYSIS USING CONVOLUTIONAL NEURAL NETWORK FOR PREDICTING SOCIO-ECONOMIC PHENOMENA

A major project report submitted in partial fulfilment of the requirements for the award of the Degree of

Bachelor of Technology

In

Computer Science and Engineering

By

A.Sai Sowmya(16011A0525)

K.Shravani(16011A0532)

M.Shreya(16011A0533)

Under the guidance of

Prof. Dr.D.Vasumathi

Department of Computer Science and Engineering

JNTUH College of Engineering Hyderabad

Kukatpally, Hyderabad – 500085

2019-2020

Department of Computer Science and Engineering,
JNTUH College of Engineering Hyderabad,
Kukatpally,Hyderabad – 500085

DECLARATION BY THE CANDIDATES

We, A.SaiSowmya(16011A0525), K.Shravani(16011A0532), M.Shreya(16011A0533) here by declare that the major-project report entitled "**Sentiment Analysis using Convolutional Neural Network for predicting Socio-Economic Phenomena**", carried out by us under the guidance of **Dr.D.Vasumathi**, is submitted in partial fulfilment of the requirements for the award of the degree of ***Bachelor of Technology*** in ***Computer Science and Engineering***. This is a record of bonafide work carried out by us and the results embodied in this project have not been reproduced/ copied from any source and have not been submitted to any other University or Institute for the award of any other degree or diploma.

A.Sai Sowmya(16011A0525)
K.Shravani(16011A0532)
M.Shreya(16011A0533)
Department of Computer Science & Engineering,
JNTUH College of Engineering,
Hyderabad.

Department of Computer Science and Engineering,
JNTUH College of Engineering Hyderabad,
Kukatpally,Hyderabad – 500085

CERTIFICATE BY THE SUPERVISOR

This is to certify that the project report entitled "**Sentiment Analysis using Convolutional Neural Network for predicting Socio-Economic Phenomena**", being submitted by **A.Sai Sowmya**(16011A0525), **K.Shravani**(16011A0532), **M.Shreya** (16011A0533) in partial fulfilment of the requirements for the award of the degree of ***Bachelor of Technology*** in ***Computer Science and Engineering***, is a record of bonafide work carried out by them. The results are verified and found satisfactory.

Dr.D.Vasumathi,
Professor,
Department of Computer Science and Engineering,
JNTUH College of Engineering,
Hyderabad.

Date:

Department of Computer Science and Engineering,
JNTUH College of Engineering Hyderabad,
Kukatpally,Hyderabad – 500085

CERTIFICATE BY THE HEAD OF DEPARTMENT

This is to certify that the project report entitled "**Sentiment Analysis using Convolutional Neural Network for predicting Socio-Economic Phenomena**", being submitted by **A.Sai Sowmya**(16011A0525), **K.Shravani**(16011A0532), **M.Shreya** (16011A0533) in partial fulfilment of the requirements for the award of the degree of ***Bachelor of Technology*** in ***Computer Science and Engineering***, is a record of bonafide work carried out by them. The results are verified and found satisfactory.

Dr.R. Sri Devi,
Professor & Head of the Department,
Department of Computer Science and Engineering,
JNTUH College of Engineering,
Hyderabad.

Date:

ACKNOWLEDGEMENT

We would like to express sincere thanks to our Supervisor **Dr.D.Vasumathi** mam, Professor for her admirable guidance and inspiration both theoretically and practically and most importantly for the drive to complete project successfully. Working under such an eminent guide was our privilege.

We owe a debt of gratitude to **Dr.R.SriDevi** mam, Professor & Head of the department of Computer Science & Engineering, for her kind considerations and encouragement in carrying out this project successfully.

We are grateful to the Project Review Committee members and Department of Computer Science & Engineering who have helped in successfully completing this project by giving their valuable suggestions and support.

We express thanks to our parents for their love, care and moral support without which we would have not been able to complete this project. It has been a constant source of inspiration for all our academic endeavours. Last but not the least, we thank the Almighty for making us a part of the world.

A.Sai Sowmya(16011A0525)

K.Shravani(16011A0532)

M.Shreya(16011A0533)

DECLARATION

We declare that this written submission represents our ideas in our own words and other ideas or words have been included we have adequately cited and referenced the original sources. We also declare that we have adhered to all the principles of the academic honesty and integrity and have not misinterpreted or fabricated or falsified any idea/data/fact/sources in our submission. We understand that any violation of the above will be cause for disciplinary action by the institute and also can evoke penal action from the sources which we have thus not been properly cited or from whom proper permission has not been taken when needed.

A.Sai Sowmya(16011A0525)

K.Shravani(16011A0532)

M.Shreya(16011A0533)

ABSTRACT

Social media has received more attention nowadays. Public and private opinion about a wide variety of subjects are expressed and spread continually via numerous social media. Twitter has been growing in popularity and nowadays, it is used every day by people to express opinions about different topics, such as products, movies, music, politicians, events, social events, among others. This project addresses the problem of sentiment analysis in twitter; that is classifying tweets according to the sentiment expressed in them: positive, negative or neutral. Due to this large amount of usage we hope to achieve a reflection of public sentiment by analyzing the sentiments expressed in the tweets. Analyzing the public sentiment is important for many applications such as firms trying to find out the response of their products in the market, predicting political elections and predicting socioeconomic phenomena like stock exchange. The aim of this project is to develop a functional classifier for accurate and automatic sentiment classification of an unknown tweet stream. We apply deep learning techniques to classify sentiment of Twitter data. The two deep learning techniques used are Long Short Term Memory (LSTM) and Convolutional Neural Network (CNN).

GUIDE: Dr.D.Vasumathi

GROUP MEMBERS:

A.Sai Sowmya (16011A0525)

K.Shravani (16011A0532)

M.Shreya (16011A0533)

TABLE OF CONTENTS

1. Introduction ……9-10

1.1. Existing Approach…….10

2. Literature Survey ……..11-13

3. Proposed Work……..14

3.1. Process Flow of the project……14

4. Implementation……..15-24

4.1. Definitions…….15-20

4.2. Packages……21-23

4.3. Platforms……24

5. Results……..25-26

6. Code……..27-48

7. Conclusion and Future Work……49

8. References……..50-51

1. INTRODUCTION

The age of internet has changed the way people express their views and opinions. It is now mainly done through blog posts, online forums, product review websites, social network sites like Twitter, Facebook, Instagram etc. Social media is generating a large volume of sentiment rich data in the form of tweets, blog posts, comments, reviews. The amount of content generated by users is too vast for a normal user to analyze. So to automate this, various sentiment analysis techniques are widely used. Sentiment analysis (a.k.a opinion mining) is the automated process of identifying and extracting the subjective information that underlies a text. This can be either an opinion, a judgment, or a feeling about a particular topic or subject. The most common type of sentiment analysis is called 'polarity detection' and consists in classifying a statement as 'positive', 'negative' or 'neutral'. Sentiment analysis is particularly useful for social media monitoring because it goes beyond metrics that focus on the number of likes or retweets, and provides a qualitative point of view.

This project intends to perform Sentiment Analysis on twitter data. Twitter is a gold mine of data. Unlike other social platforms, almost every user's tweets are completely public and pullable. With more than 330 million active users, sending a daily average of 500 million Tweets, Twitter allows businesses to reach a broad audience and connect with customers without intermediaries. Twitter data is pretty specific and can be a large door into the insights of the general public, and how they receive a topic. That, combined with the openness and the generous rate limiting of Twitter's API, can produce poweful resuls. Therefore, sentiment analysis of tweets provides exciting opportunities across many fields, from business to politics and also for monitoring and analysing social phenomena.

1.1. EXISTING APPROACH

Sentiment Classification can be carried out on three levels of extraction: the aspect or feature level, the sentence level, and the document level. The approaches that are currently used to address the problem of sentiment analysis are:

Lexicon-based techniques were the first to be used for sentiment analysis. It is performed by using a dictionary of terms or by statistical analysis of the contents of collection of documents, using techniques based on k-nearest neighbours(k-NN),conditional random field(CRF) and hidden Markov models(HMM).

Machine-learning based techniques refer to techniques such as the naive Bayes classifier, or support vector machines(SVM).The input to those algorithms include lexical features, parts of speech, or adjectives and adverbs. The accuracy of these systems depends on which features are chosen. Considering Naive Bayes, for example, we have a few limitations like:

- Accurate classifications rely on representative datasets, i.e., if the training set is biased towards a certain polarity (e.g., neutral) our classifications will likely be biased as well.
- Naïve Bayes makes strong assumptions on the independence of the features, making the underlying probabilities unreliable.
- Using documents as the granularity of the training labels often leads to poor classifications when using Naïve Bayes with Bag of Words models.

2. LITERATURE SURVEY

Authors: Aliaksei Severyn,Alessandro Moschitti

Paper: Twitter Sentiment Analysis with deep Convolutional Neural Networks(2015)

- This paper describes deep learning system for sentiment analysis of tweets.
- The main contribution of this work is a new model for initializing the parameter weights of the convolutional neural network, which is crucial to train an accurate model while avoiding the need to inject any additional features.
- Briefly, they used an unsupervised neural language model to train initial word embeddings that are further tuned by their deep learning model on a distant supervised corpus.
- At a final stage, the pre-trained parameters of the network are used to initialize the model.The latter is trained on the supervised training data.

Authors: Nal Kalchbrenner,Edward Grefenstette,Phil Blunsom

Paper: A Convolutional Neural network for Modelling Sentences(2014)

- This paper describes a convolutional architecture dubbed the Dynamic Convolutional Neural Network (DCNN) that is adoptedfor the semantic modelling of sentences.
- The network uses Dynamic k-Max Pooling, a global pooling operation over linear sequences.
- The network handles input sentences of varying length and induces a feature graph over the sentence that is capable of explicitly capturing short and long-range relations.

- The network does not rely on a parse tree and is easily applicable to any language. They tested the DCNN in four experiments: small scale binary and multi-class sentiment prediction, six-way question classification and Twitter sentiment prediction by distant supervision.
- The network achieves excellent performance in the first three tasks and a greater than 25% error reduction in the last task with respect to the strongest baseline.

Authors: Qurat Tul Ain , Mubashir Ali , Amna Riaz , Amna Noureen, Muhammad Kamran, Babar Hayat and A. Rehman.
Paper: Sentiment Analysis Using Deep Learning Techniques: A Review(2017)

- The challenge for sentiment analysis is lack of sufficient labeled data in the field of Natural Language Processing (NLP). And to solve this issue, the sentiment analysis and deep learning techniques have been merged because deep learning models are effective due to their automatic learning capability.
- This Review Paper highlights latest studies regarding the implementation of deep learning models such as deep neural networks, convolutional neural networks and many more for solving different problems of sentiment analysis such as sentiment classification, cross lingual problems, textual and visual analysis and product review analysis, etc. .
- Deep learning networks are better than SVMs and normal neural networks because they have more hidden layers as compared to normal neural networks that have one or two hidden layers.
- Deep learning networks are capable to provide training in both supervised/unsupervised ways. Deep learning networks carry out

automatic feature extraction and doesnt involve human intervention therefore it can save time because feature engineering is not needed.

- Sentiment Analysis comprises different kinds of problem statements. The capability of settling in the task variations by having little alterations in system itself includes a feather in strength of Deep Learning standard.

Authors: Yoon Kim

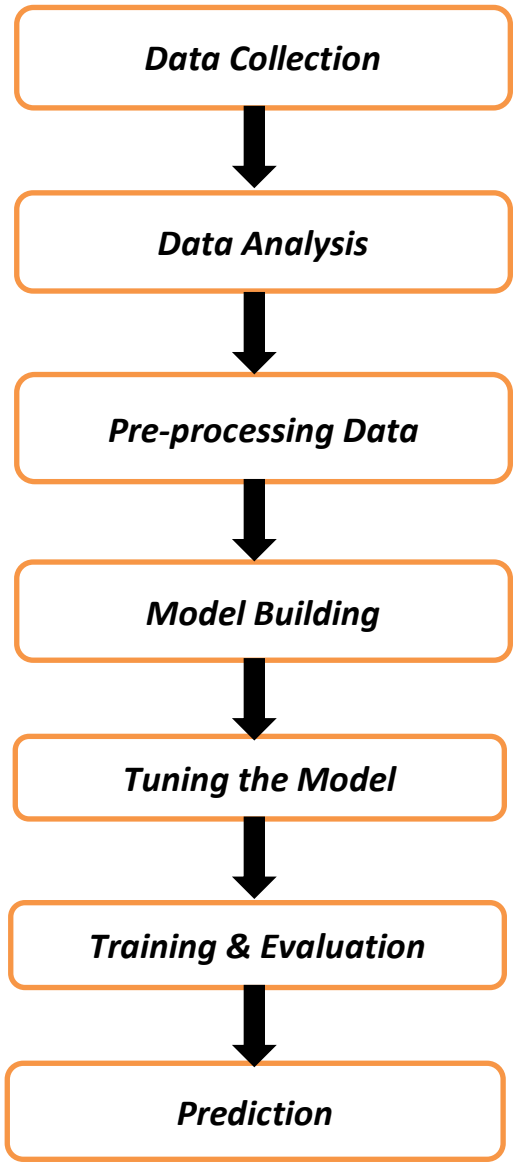
Paper: Convolutional Neural networks for Sentence Classification(2014)

- This paper reports on a series of experiments with convolutional neural networks (CNN) trained on top of pre-trained word vectors for sentence-level classification tasks.
- The author showed that a simple CNN with little hyperparameter tuning and static vectors achieves excellent results on multiple benchmarks.
- Learning task-specific vectors through fine-tuning offers further gains in performance.
- This work additionally proposed a simple modification to the architecture to allow for the use of both task-specific and static vectors. The CNN models discussed like CNN-rand, CNN-static, CNN-non static improved upon the state of the art on 4 out of 7 tasks, which include sentiment analysis and question classification.
- The results add to the well-established evidence that unsupervised pre-training of word vectors is an important ingredient in deep learning for NLP.

3. PROPOSED WORK

In this report we implemented deep learning models to classify sentiment of twitter data. In terms of data, deep learning is a promising approach and it efforts to learn high level abstractions by exploiting the hierarchical architectures. Our goal is to use Neural networks to classify any tweet into a "positive" or "negative" one. We proposed two different Neural network models that aim to combine the popular LSTM(long short term memory neural networks)with CNNs(Convolutional Neural Networks).In these models, features are learned and extracted automatically, achieving better accuracy and performance leading to better results than traditional models.

3.1. PROCESS FLOW OF THE PROJECT:

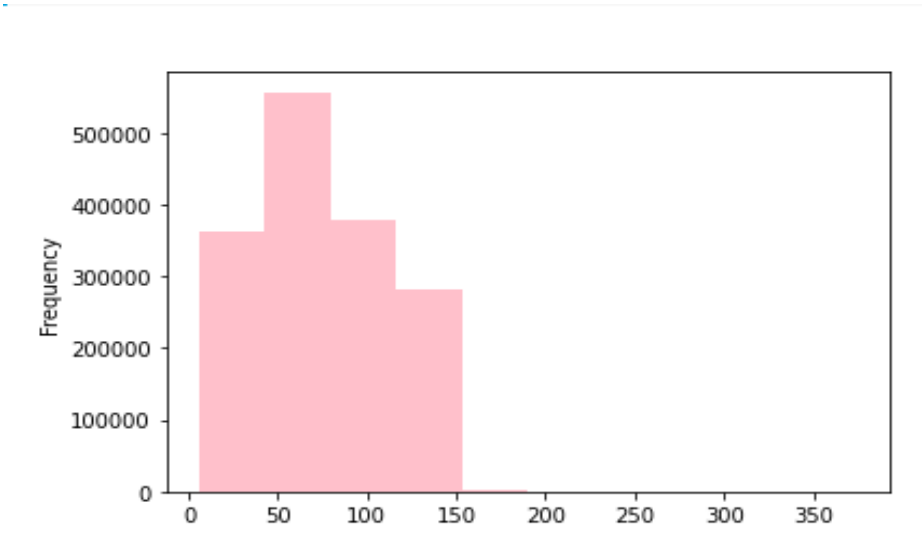


4. IMPLEMENTATION

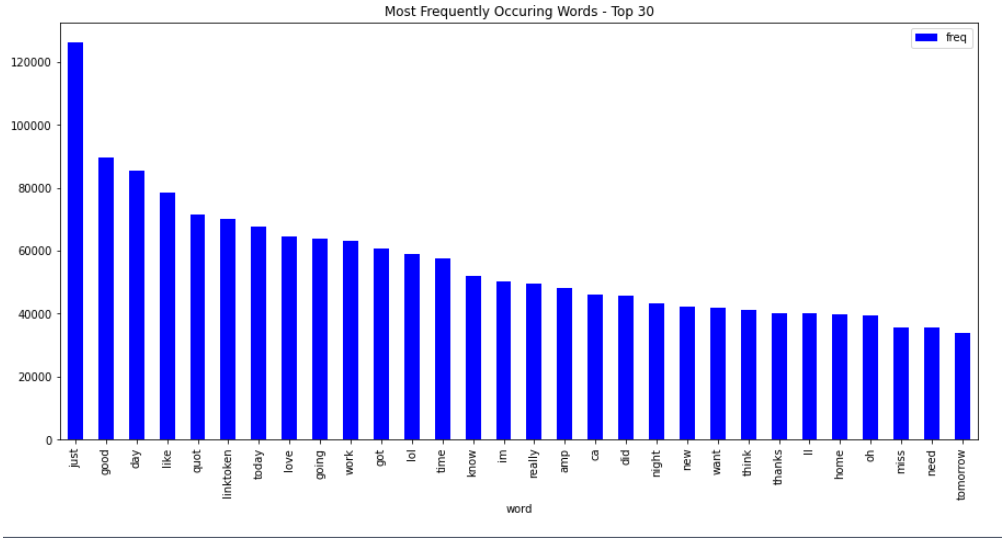
➢ *4.1. Definitions:*

Exploratory data Analysis:
Exploratory data analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. It refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

We have done the following analysis to get some insights of our Twitter data.



The above graph gives the distribution of tweets in the data.



Data Preprocessing (or) Data Cleaning:

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviours or trends, and is likely to contain many errors. Data preprocessing prepares raw data for further processing.

The considered data set is comprised of very much unstructured tweets which should be preprocessed . In this project, we tried out the following techniques of preprocessing the raw data.

- *Removal of Punctuations:*

  Punctuations will be always a disturbance in NLP specially hashtags and "@" play a major role in tweets. TextBlob's word extraction feature from a sentence removes punctuations in an optimal level. The left out punctuations and other unusual notations will be removed in the upcoming preprocessing techniques.

- *Removal of Stop Words:*

  In an NLP task the stopwords (most common words e.g: is, are, have) do not make sense in learning because they don't have connections with sentiments. So removing them saves the computational power as well as increases the accuracy of the model.

- *Normalisation of words:*

  All the unusual symbols and the numerical values were removed . But still we may encounter multiple representations of the same word.(e.g: play, plays, played, playing) Even though the words are different they bring us the same meaning as the normal word “play”. So we need to do Lexicon Normalization approach to solve this issue. NLTK’s built-in WordNetLemmatizer does this requirement.

Model Building:

We built a model that combines Convolutional Neural Networks(CNN) and Long Short Term Neural Networks(LSTM).The background information about these models is given below.

Convolutional Neural Networks:

Initially designed for image recognition, Convolutional Neural Networks (CNNs) have become an incredibly versatile model used for a wide array of tasks. CNNs have the ability of recognizing local features inside a multi-dimensional field. For example, on an image, CNNs will be able to spot particular features such as a wheel or a smile, regardless of where these might be located.

Basic CNNs (as outlined in the figure) work by feeding multidimensional data (e.g. images, word embeddings, etc.) to a Convolutional layer which will be composed of multiple filters that will learning different features. Notice that these filters are sequentially applied to different sections of the input. The output is usually pooled or sub-sampled to smaller dimensions and later fed into a connected layer.

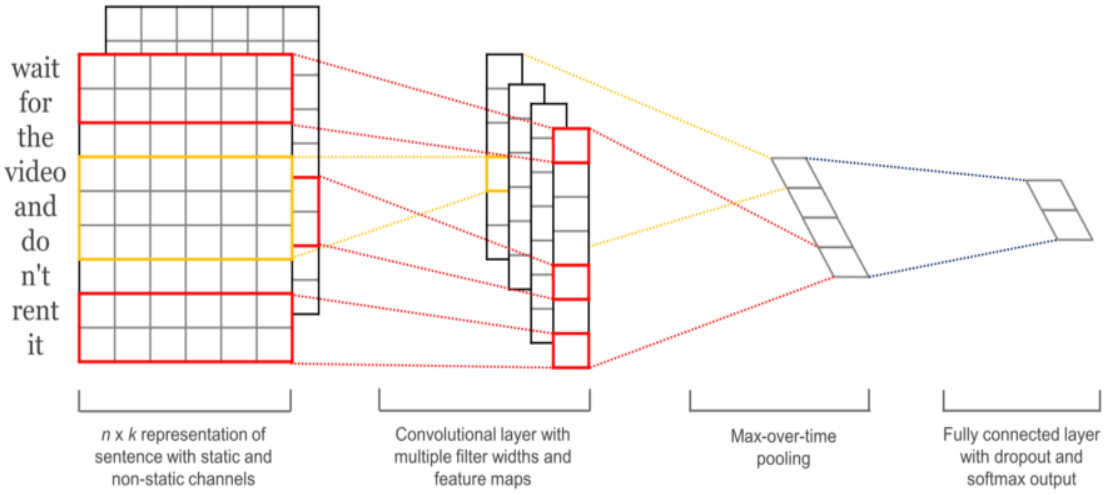


Fig: Kim Y(2014) ,Convolutional Neural Networks for Sentence Classification

The intuition behind using CNNs on text relies on the fact that text is structured and organized; As such, we can expect a CNN model to discover and learn patterns that would otherwise be lost in a feed-forward network. For example, it might be able to distinguish that using “down” in the context of “down to earth” is actually of positive sentiment as opposed to other phrases such as “feeling down”. Furthermore, it will be able to extract these features regardless of where they occur in the sentence.

Convolutional neural networks (CNN) utilize layers with convolving filters that are applied to local features (LeCun et al., 1998). Originally invented for computer vision, CNN models have subsequently been shown to be effective for NLP and have achieved excellent results in semantic parsing (Yih et al., 2014), search query retrieval (Shen et al., 2014), sentence modeling (Kalchbrenner et al., 2014), and other traditional NLP tasks (Collobert et al., 2011).

Long Short Term Neural Networks:

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture[1] used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDS's (intrusion detection systems).
A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

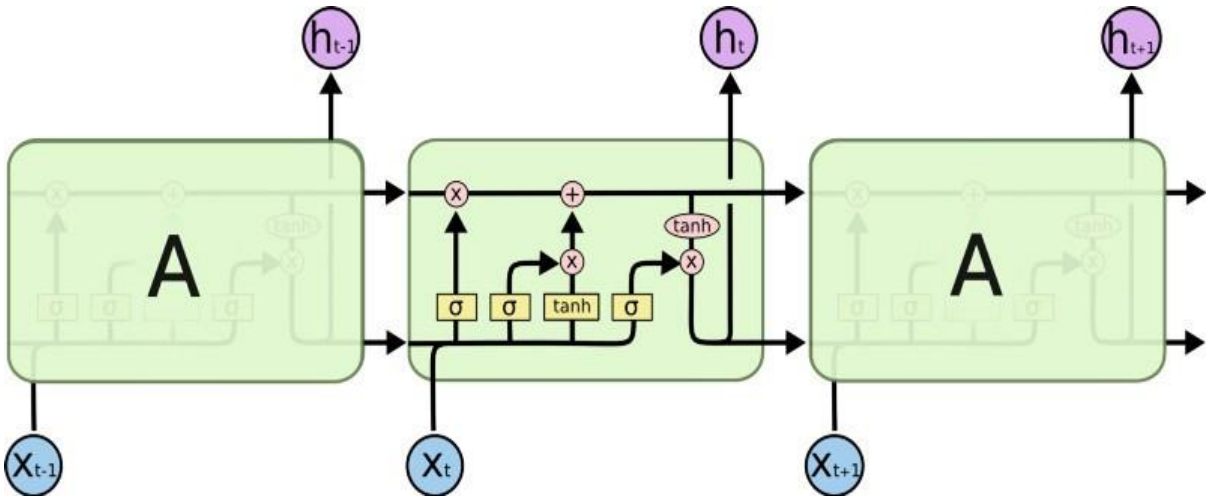


Fig:LSTM Network(Source:colah.github.io/Understanding LSTMs)

Long-Short Term Memory (LSTM) networks are a type of Recurrent Neural Network architecture that is designed to “remember” previously read values for any given period of time. LSTMs usually contain three gates that control the flow to and from their memories. The “input gate” controls the input of new information to the memory. The “forget gate” controls how long certain values are held in memory. Finally, the “output gate” controls how much the value stored in memory affects the output activation of the block.

In our particular case, it is possible that an LSTM could allow us to capture changing sentiment in a tweet. For example, a sentence such as: *At first I loved it, but then I ended up hating it.* has words with conflicting sentiments that would end-up confusing a simple Feed-Forward network. The LSTM, on the other hand, could learn that sentiments expressed towards the end of a sentence mean more than those expressed at the start.

LSTM-CNN Model:

This CNN-LSTM model consists of an initial LSTM layer which will receive word embeddings for each token in the tweet as inputs. The intuition is that its output tokens will store information not only of the initial token, but also any previous tokens; In other words, the LSTM layer is generating a new encoding for the original input. The output of the LSTM layer is then fed into a convolution layer which we expect will extract local features. Finally the convolution layer's output will be pooled to a smaller dimension and ultimately outputted as either a positive or negative label.

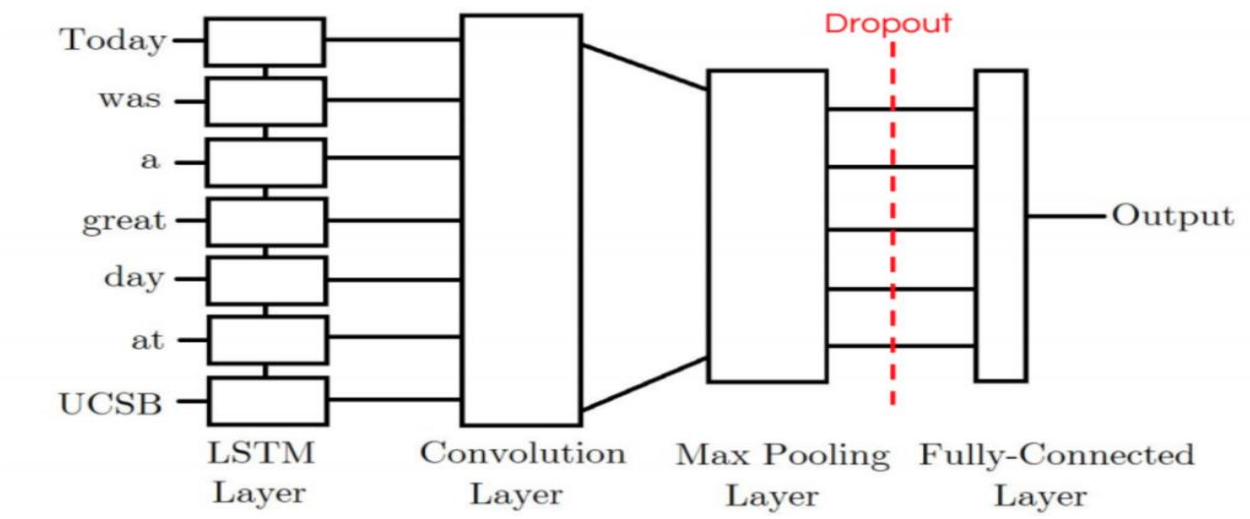


Fig: LSTM-CNN Model

➢ *4.2. Packages*:

- Pandas
- Numpy
- Seaborn
- Matplotlib
- tkinter

Pandas:

pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

- A fast and efficient DataFrame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format.
- Intelligent data alignment and integrated handling of missing data: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form.
- Flexible reshaping and pivoting of data sets.
- Columns can be inserted and deleted from data structures for size mutability.
- Aggregating or transforming data with a powerful group by engine allowing split-apply-combine operations on data sets.
- High performance merging and joining of data sets.

Numpy:

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Using NumPy, a developer can perform the following operations −

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.

Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation

Seaborn:

Seaborn is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures. Here is some of the functionality that seaborn offers:

- A dataset-oriented API for examining relationships between multiple variables.
- Specialized support for using categorical variables to show observations or aggregate statistics.
- Options for visualizing univariate or bivariate distributions and for comparing them between subsets of data.
- Automatic estimation and plotting of linear regression models for different kinds dependent variables.
- Convenient views onto the overall structure of complex datasets.
- High-level abstractions for structuring multi-plot grids that let you easily build complex visualizations.

- Concise control over matplotlib figure styling with several built-in themes.
- Tools for choosing color palettes that faithfully reveal patterns in your data.

Matplotlib:

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code.

Tkinter:

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. It commonly comes bundled with Python, using Tk and is Python's standard GUI framework. It is famous for its simplicity and graphical user interface. It is open-source and available under the Python License.

➢ *4.3. Platforms*:

- **Jupyter Notebook:**

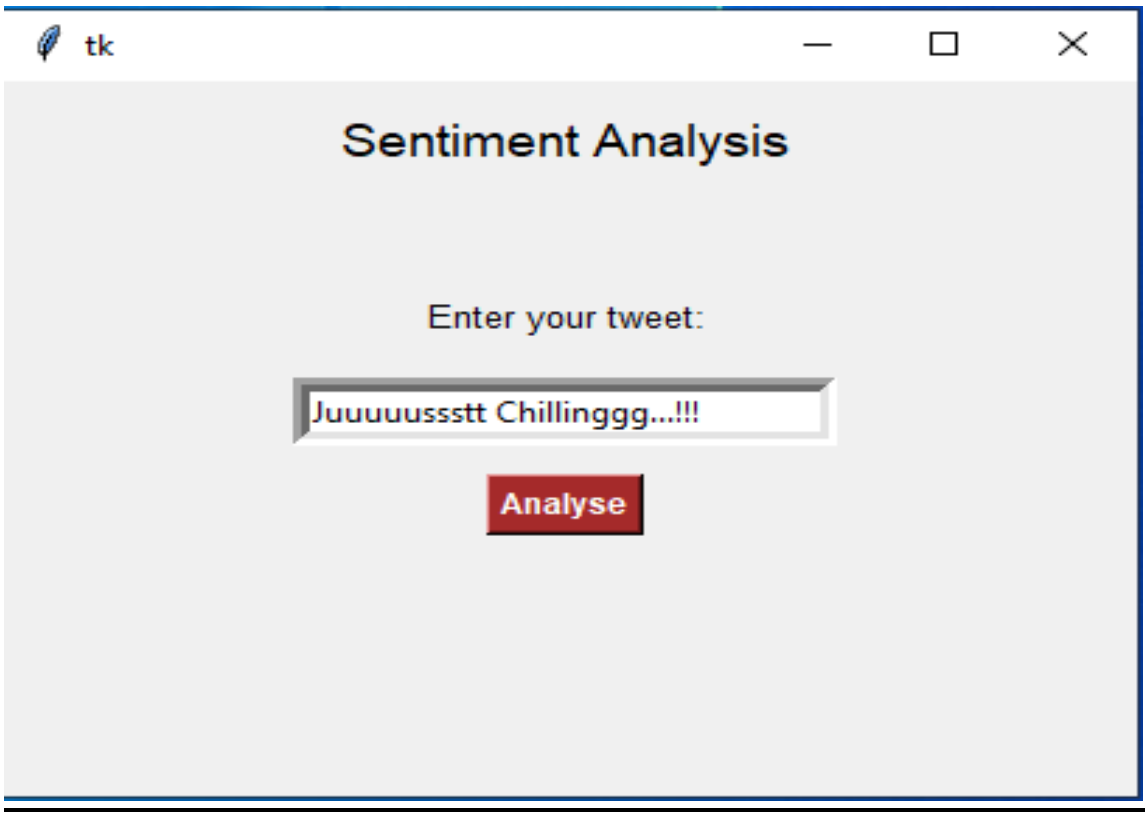
  Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

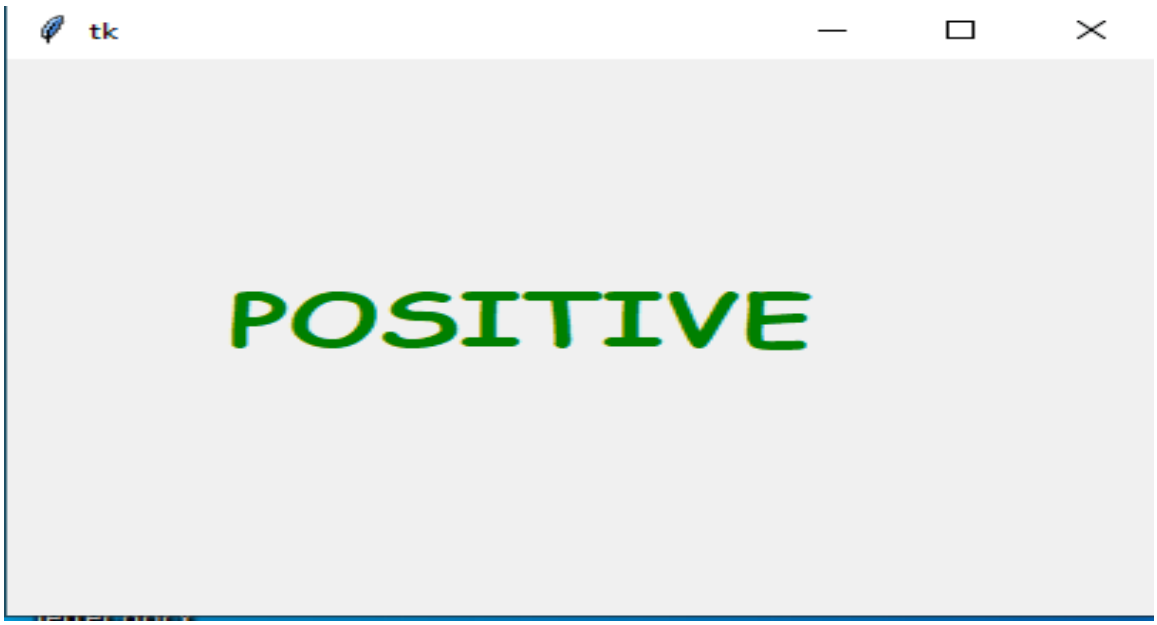
- **Tensorflow**:

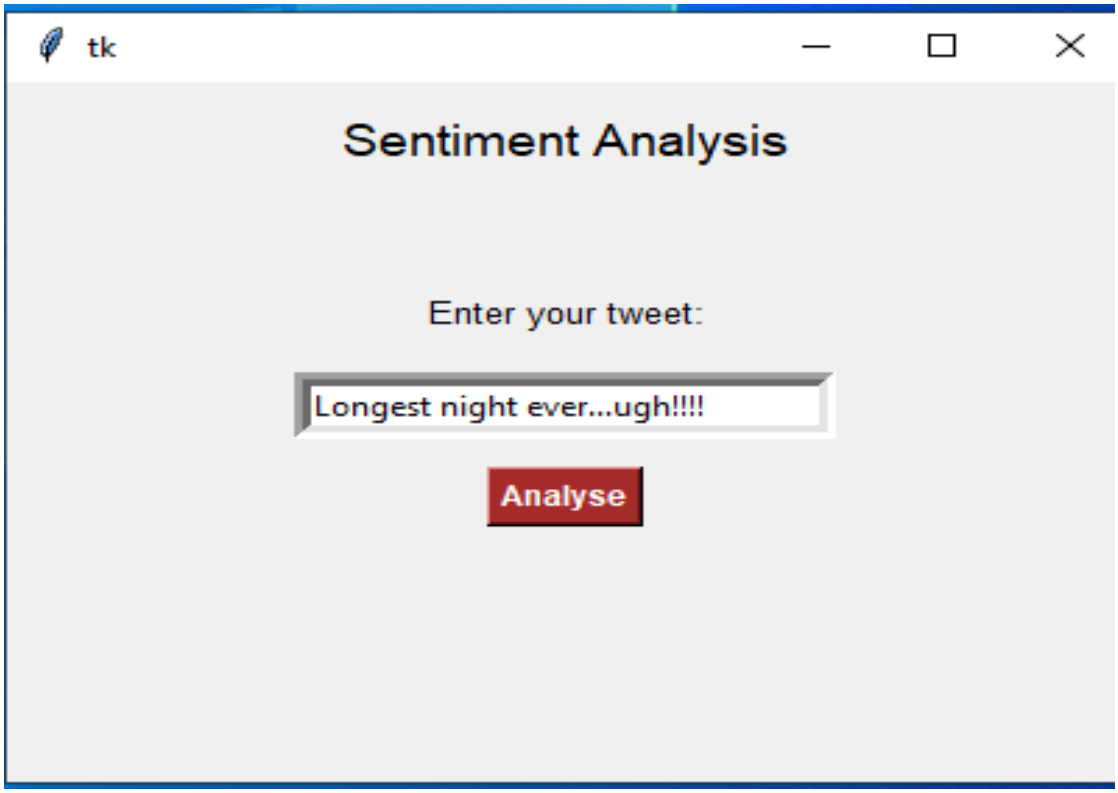
  TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. It is used for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. TensorFlow offers multiple levels of abstraction. It is used to build and train state-of-the-art models without sacrificing speed or performance. TensorFlow gives the flexibility and control with features like the Keras Functional API and Model Subclassing API for creation of complex topologies.

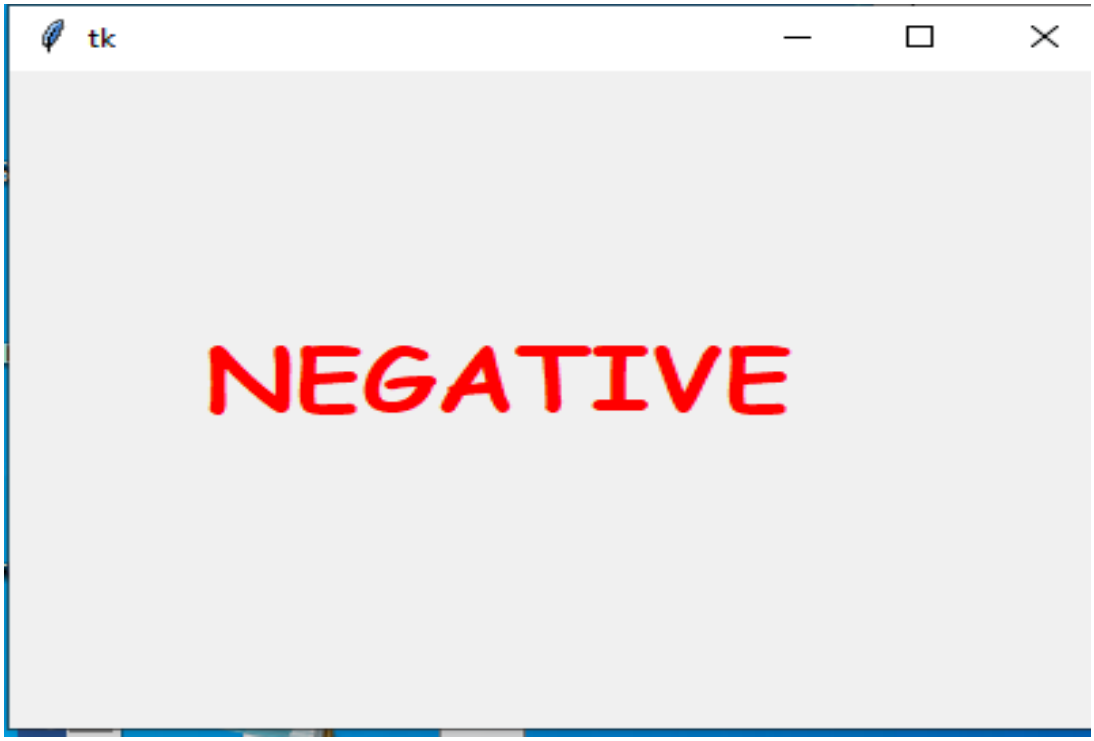
5. RESULTS

The model built is integrated with tkinter GUI interface to predict the tweet entered by the user as "***positive***" or "***negative***".






tk
Sentiment Analysis
Enter your tweet:
Longest night ever...ugh!!!!
Analyse


tk
NEGATIVE

6. CODE:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import csv
import re
import random
import numpy as np

from IPython import embed
```

Data Analysis:

```
data = pd.read_csv('/kaggle/input/twitter-sentiment/Sentiment Analysis Dataset 2.csv',error_bad_lines=False)
```

```
b'Skipping line 8836: expected 4 fields, saw 5\n'
b'Skipping line 535882: expected 4 fields, saw 7\n'
```

```
print(data.shape)
```

```
(1578612, 4)
```

```
data.head()
```

	ItemID	Sentiment	SentimentSource	SentimentText
0	1	0	Sentiment140	is so sad for my APL frie...
1	2	0	Sentiment140	I missed the New Moon trail...
2	3	1	Sentiment140	omg its already 7:30 :O
3	4	0	Sentiment140	.. Omgaga. Im sooo im gunna CRy. I'...
4	5	0	Sentiment140	i think mi bf is cheating on me!!! ...

```
data.isnull().any()
```

```
ItemID           False
Sentiment        False
SentimentSource  False
SentimentText    False
dtype: bool
```

```
#checking out the negative comments from the train set
data[data['Sentiment'] == 0].head(10)
```

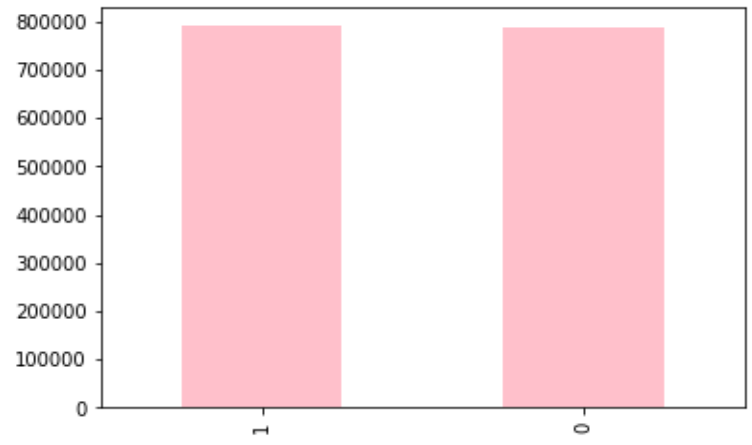
	ItemID	Sentiment	SentimentSource	SentimentText
0	1	0	Sentiment140	is so sad for my APL frie...
1	2	0	Sentiment140	I missed the New Moon trail...
3	4	0	Sentiment140	.. Omgaga. Im sooo im gunna CRy. I'...
4	5	0	Sentiment140	i think mi bf is cheating on me!!! ...
5	6	0	Sentiment140	or i just worry too much?
7	8	0	Sentiment140	Sunny Again Work Tomorrow :-| ...
10	11	0	Sentiment140	I must think about positive..
12	13	0	Sentiment140	this weekend has sucked so far
13	14	0	Sentiment140	jb isnt showing in australia any more!
14	15	0	Sentiment140	ok thats it you win.

```
#checking out the postive comments from the train set
data[data['Sentiment'] == 1].head(10)
```

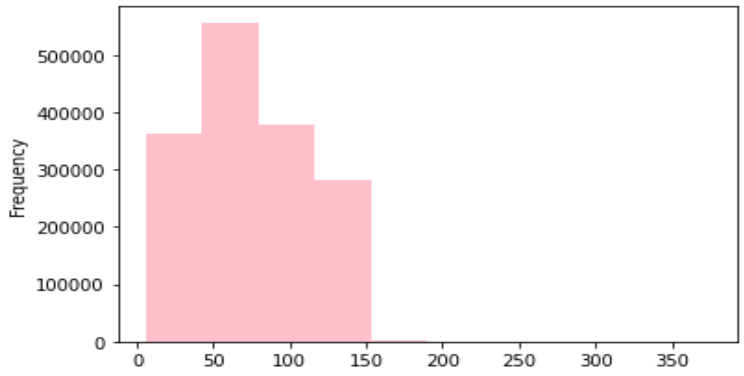
	ItemID	Sentiment	SentimentSource	SentimentText
2	3	1	Sentiment140	omg its already 7:30 :O
6	7	1	Sentiment140	Juuuuuuuuuuuuuuuussssst Chillin!!
8	9	1	Sentiment140	handed in my uniform today . i miss you ...
9	10	1	Sentiment140	hmmmm.... i wonder how she my number @-)
11	12	1	Sentiment140	thanks to all the haters up in my face a...
17	18	1	Sentiment140	Feeling strangely fine. Now I'm gonna go l...
22	23	1	Sentiment140	You're the only one who can see this cause...
28	29	1	Sentiment140	goodbye exams, HELLO ALCOHOL TONIGHT
38	39	1	Sentiment140	uploading pictures on friendster
41	42	1	Sentiment140	(: !!!!!! - so i wrote something last week. ...

```
data['Sentiment'].value_counts().plot.bar(color = 'pink', figsize = (6, 4))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f03074e2518>
```



```
# checking the distribution of tweets in the data
length_train = data['SentimentText'].str.len().plot.hist(color = 'pink', figsize = (6, 4))
```




```
data.groupby('Sentiment').describe()
```

|:

	ItemID							
	count	mean	std	min	25%	50%	75%	max
Sentiment								
0	788435.0	840458.653997	449231.963498	1.0	474930.0	868240.0	1223861.5	1578627.0
1	790177.0	738294.923096	456402.871802	3.0	339362.0	706525.0	1135595.0	1578624.0

Data Cleaning:

```
def clean_str(string):


    #EMOJIS
    string = re.sub(r":\)","emojihappy1",string)
    string = re.sub(r":P","emojihappy2",string)
    string = re.sub(r":p","emojihappy3",string)
    string = re.sub(r":>","emojihappy4",string)
    string = re.sub(r":3","emojihappy5",string)
    string = re.sub(r":D","emojihappy6",string)
    string = re.sub(r" XD ","emojihappy7",string)
    string = re.sub(r" <3 ","emojihappy8",string)

    string = re.sub(r":\(","emojisad9",string)
    string = re.sub(r":<","emojisad10",string)
    string = re.sub(r":<","emojisad11",string)
    string = re.sub(r">:\(","emojisad12",string)

    #MENTIONS "(@)\w+"
    string = re.sub(r"(@)\w+","",string)

    #WEBSITES
    string = re.sub(r"http(s)*:(\S)*","linktoken",string)

    #STRANGE UNICODE \x...
    string = re.sub(r"\\x(\S)*","",string)

    #General Cleanup and Symbols
    string = re.sub(r"[^A-Za-z0-9(),!?\'\`]", " ", string)
    string = re.sub(r"\'s", " \'s", string)
    string = re.sub(r"\'ve", " \'ve", string)
    string = re.sub(r"n\'t", " n\'t", string)
    string = re.sub(r"\'re", " \'re", string)
    string = re.sub(r"\'d", " \'d", string)
    string = re.sub(r"\'ll", " \'ll", string)
    string = re.sub(r",", " , ", string)
    string = re.sub(r"!", " ! ", string)
    string = re.sub(r"\(", " \( ", string)
    string = re.sub(r"\)", " \) ", string)
    string = re.sub(r"\?", " \? ", string)
    string = re.sub(r"\s{2,}", " ", string)

    return string.strip().lower()
```

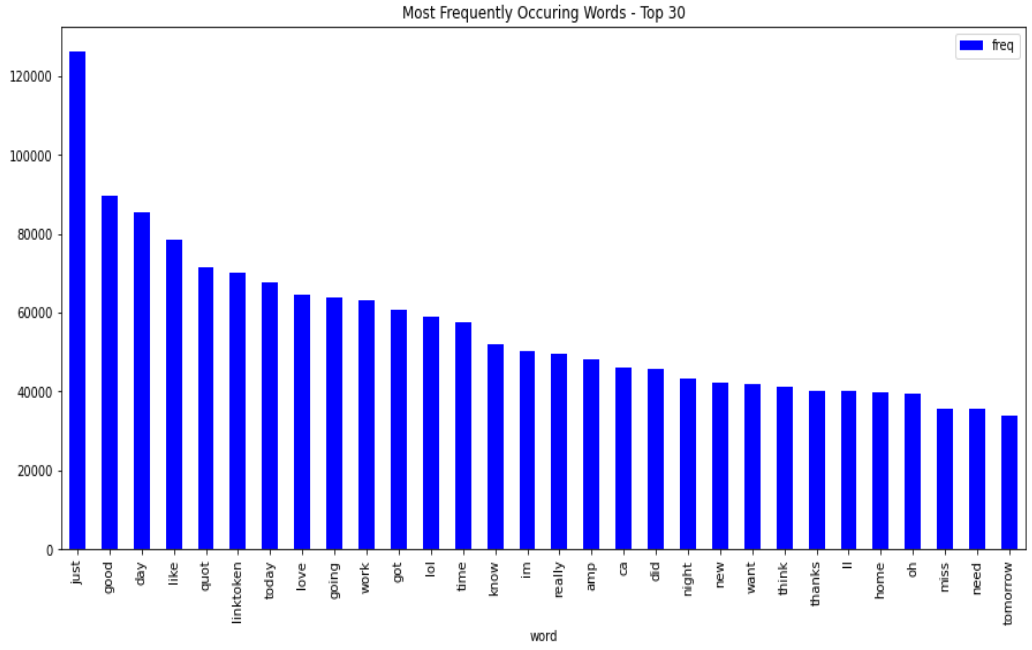
```
data['clean_text'] = data['SentimentText'].apply(clean_str)
```

```
data.head()
```

	ItemID	Sentiment	SentimentSource	SentimentText	clean_text
0	1	0	Sentiment140	is so sad for my APL frie...	is so sad for my apl friend
1	2	0	Sentiment140	I missed the New Moon trail...	i missed the new moon trailer
2	3	1	Sentiment140	omg its already 7:30 :O	omg its already 7emojihappy50 o
3	4	0	Sentiment140	.. Omgaga. Im sooo im gunna CRy. I'...	omgaga im sooo im gunna cry i 've been at this...
4	5	0	Sentiment140	i think mi bf is cheating on me!!! ...	i think mi bf is cheating on me ! ! ! t t

```
from sklearn.feature_extraction.text import CountVectorizer


cv = CountVectorizer(stop_words = 'english')
words = cv.fit_transform(data.clean_text)

sum_words = words.sum(axis=0)

words_freq = [(word, sum_words[0, i]) for word, i in cv.vocabulary_.items()]
words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)

frequency = pd.DataFrame(words_freq, columns=['word', 'freq'])

frequency.head(30).plot(x='word', y='freq', kind='bar', figsize=(15, 7), color = 'blue')
plt.title("Most Frequently Occuring Words - Top 30")
```

```
Text(0.5, 1.0, 'Most Frequently Occuring Words - Top 30')
```



```
pip install wordcloud
```

```
from wordcloud import WordCloud

wordcloud = WordCloud(background_color = 'white', width = 1000, height = 1000).generate_from_frequencies(dict(words_freq))

plt.figure(figsize=(10,8))
plt.imshow(wordcloud)
plt.title("WordCloud - Vocabulary from Dataset", fontsize = 22)
```

```
Text(0.5, 1.0, 'WordCloud - Vocabulary from Dataset')
```

```
positive_words =' '.join([text for text in data['clean_text'][data['Sentiment'] == 1]])

wordcloud = WordCloud(width=800, height=500, random_state = 0, max_font_size = 110).generate(positive_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title('The Positive Words')
plt.show()
```


```
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from textblob import TextBlob
```

```
def form_sentence(tweet):
    tweet_blob = TextBlob(tweet)
    return ' '.join(tweet_blob.words)

print(form_sentence(data['SentimentText'].iloc[0]))
print(data['SentimentText'].iloc[0])
text=form_sentence(data['SentimentText'].iloc[0])
```

```
is so sad for my APL friend
                    is so sad for my APL friend.............
```

```
def no_user_alpha(tweet):
    tweet_list = [ele for ele in tweet.split() if ele != 'user']
    clean_tokens = [ele for ele in tweet.split() if re.match(r'[^\W\d]*$', ele)]
    clean_s = ' '.join(clean_tokens)
    clean_mess = [word for word in clean_s.split() if word.lower() not in stopwords.words('english')]
    return ' '.join(clean_mess)
print(no_user_alpha(text))
print(data['SentimentText'].iloc[0])
text=no_user_alpha(text)
```

```
sad APL friend
                    is so sad for my APL friend.............
```

```
def normalization(tweet_list):
        lem = WordNetLemmatizer()
        normalized_tweet = []
        for word in tweet_list:
            normalized_text = lem.lemmatize(word,'v')
            normalized_tweet.append(normalized_text)
        return ' '.join(normalized_tweet)
print(normalization(text.split()))
print(data['SentimentText'].iloc[0])
```

```
sad APL friend
                    is so sad for my APL friend.............
```

```
data['clean_text'] = data['clean_text'].apply(form_sentence)
```

```
data.head()
```

	ItemID	Sentiment	SentimentSource	SentimentText	clean_text
0	1	0	Sentiment140	is so sad for my APL frie...	i s s o s a d f o r m y a p l f r ...
1	2	0	Sentiment140	I missed the New Moon trail...	i m i s s e d t h e n e w m o o n t ...
2	3	1	Sentiment140	omg its already 7:30 :O	o m g i t s a l r e a d y 7 e m o j i h ...
3	4	0	Sentiment140	.. Omgaga. Im sooo im gunna CRy I'...	o m g a g a i m s o o o i m g u n n a ...
4	5	0	Sentiment140	i think mi bf is cheating on me!!! ...	i t h i n k m i b f i s c h e a t i ...

```
!pip install tensorflow==1.14
import tensorflow as tf
```

```
#Separates a file with mixed positive and negative examples into two.
def separate_dataset(filename):
    good_out = open("good_file","w+",encoding="utf8");
    bad_out  = open("bad_file","w+",encoding="utf8");

    seen = 1;
    with open(filename,'r',encoding="utf8") as f:
        reader = csv.reader(f)
        next(reader)

        for line in reader:
            seen +=1
            sentiment = line[1]
            sentence = line[4]

            if (sentiment == "0"):
                bad_out.write(sentence+"\n")
            else:
                good_out.write(sentence+"\n")

            if (seen%10000==0):
                print (seen);

    good_out.close();
    bad_out.close();
```

```
separate_dataset("/kaggle/input/twitter-sentiment/Sentiment Analysis Dataset 2.csv");
```

```
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
110000
120000
130000
140000
150000
160000
170000
```

```
#Load Datafiles
def get_dataset(goodfile,badfile,limit,randomize=True):
    good_x = list(open(goodfile,"r",encoding="utf8").readlines())
    good_x = [s.strip() for s in good_x]

    bad_x  = list(open(badfile,"r",encoding="utf8").readlines())
    bad_x  = [s.strip() for s in bad_x]

    if (randomize):
        random.shuffle(bad_x)
        random.shuffle(good_x)

    good_x = good_x[:limit]
    bad_x = bad_x[:limit]

    x = good_x + bad_x
    x = [clean_str(s) for s in x]


    positive_labels = [[0, 1] for _ in good_x]
    negative_labels = [[1, 0] for _ in bad_x]
    y = np.concatenate([positive_labels, negative_labels], 0)
    return [x,y]
```

```
#Generate random batches

def gen_batch(data, batch_size, num_epochs, shuffle=True):
    """
    Generates a batch iterator for a dataset.
    """
    data = np.array(data)
    data_size = len(data)
    num_batches_per_epoch = int((len(data)-1)/batch_size) + 1
    for epoch in range(num_epochs):
        # Shuffle the data at each epoch
        if shuffle:
            shuffle_indices = np.random.permutation(np.arange(data_size))
            shuffled_data = data[shuffle_indices]
        else:
            shuffled_data = data
        for batch_num in range(num_batches_per_epoch):
            start_index = batch_num * batch_size
            end_index = min((batch_num + 1) * batch_size, data_size)
            yield shuffled_data[start_index:end_index]
```

```
# Data Preparation
filename = "/kaggle/input/twitter-sentiment/Sentiment Analysis Dataset 2.csv"
goodfile = "/kaggle/input/extracted-data/good_file"
badfile = "/kaggle/input/extracted-data/bad_file"
```

+ Code + Markdown

```
x_text, y = get_dataset(goodfile, badfile, 5000)
```

```
good_tweets = pd.read_csv('good_file',error_bad_lines=False)
```

```
print(good_tweets.shape)
```

```
(594585, 1)
```

```
good_tweets.head(10)
```

	omg its already 7:30 :O
0	Juuuuuuuuuuuuuuuuussssst Chillin!!
1	handed in my uniform today . i miss you ...
2	hmmmm.... i wonder how she my number @-)
3	thanks to all the haters up in my face a...
4	Feeling strangely fine. Now I'm gonna go l...
5	You're the only one who can see this cause...
6	uploading pictures on friendster
7	(: !!!!!! - so i wrote something last week. ...
8	... health class (what a joke!)
9	@ginaaa <3 GO TO THE SHOW TONIGHT

```
bad_tweets = pd.read_csv('bad_file',error_bad_lines=False)
```

```
print(bad_tweets.shape)
```

```
(449373, 1)
```

+ Code + Markdown

```
bad_tweets.head(10)
```

	is so sad for my APL friend.....
0	I missed the New Moon trail...
1	.. Omgaga. Im sooo im gunna CRy. I'...
2	i think mi bf is cheating on me!!! ...
3	or i just worry too much?
4	Sunny Again Work Tomorrow :-| ...
5	I must think about positive..
6	this weekend has sucked so far
7	jb isnt showing in australia any more!
8	ok thats it you win.
9	<-------- This is the way i feel right ...

Model Building:

```
import tensorflow as tf
import numpy as np
from IPython import embed

class LSTM_CNN(object):
    def __init__(self, sequence_length, num_classes, vocab_size, embedding_size, filter_sizes, num_filters, l2_reg_lambda=0.0,num_hidden=100):

        # PLACEHOLDERS
        self.input_x = tf.placeholder(tf.int32, [None, sequence_length], name="input_x")    # X - The Data
        self.input_y = tf.placeholder(tf.float32, [None, num_classes], name="input_y")      # Y - The Lables
        self.dropout_keep_prob = tf.placeholder(tf.float32, name="dropout_keep_prob")       # Dropout



        l2_loss = tf.constant(0.0) # Keeping track of l2 regularization loss

        #1. EMBEDDING LAYER ################################################################
        with tf.device('/cpu:0'), tf.name_scope("embedding"):
            self.W = tf.Variable(tf.random_uniform([vocab_size, embedding_size], -1.0, 1.0),name="W")
            self.embedded_chars = tf.nn.embedding_lookup(self.W, self.input_x)
            #self.embedded_chars_expanded = tf.expand_dims(self.embedded_chars, -1)
```

```
        #2. LSTM LAYER ######################################################################
        self.lstm_cell = tf.contrib.rnn.LSTMCell(32,state_is_tuple=True)
        #self.h_drop_exp = tf.expand_dims(self.h_drop,-1)
        self.lstm_out,self.lstm_state = tf.nn.dynamic_rnn(self.lstm_cell,self.embedded_chars,dtype=tf.float32)
        #embed()

        self.lstm_out_expanded = tf.expand_dims(self.lstm_out, -1)

        #2. CONVOLUTION LAYER + MAXPOOLING LAYER (per filter) ###############################
        pooled_outputs = []
        for i, filter_size in enumerate(filter_sizes):
            with tf.name_scope("conv-maxpool-%s" % filter_size):
                # CONVOLUTION LAYER
                filter_shape = [filter_size, embedding_size, 1, num_filters]
                W = tf.Variable(tf.truncated_normal(filter_shape, stddev=0.1), name="W")
                b = tf.Variable(tf.constant(0.1, shape=[num_filters]), name="b")
                conv = tf.nn.conv2d(self.lstm_out_expanded, W,strides=[1, 1, 1, 1],padding="VALID",name="conv")
                # NON-LINEARITY
                h = tf.nn.relu(tf.nn.bias_add(conv, b), name="relu")
                # MAXPOOLING
                pooled = tf.nn.max_pool(h, ksize=[1, sequence_length - filter_size + 1, 1, 1], strides=[1, 1, 1, 1], padding='VALID', name="pool")
                pooled_outputs.append(pooled)
```

```
        # COMBINING POOLED FEATURES
        num_filters_total = num_filters * len(filter_sizes)
        self.h_pool = tf.concat(pooled_outputs, 3)
        self.h_pool_flat = tf.reshape(self.h_pool, [-1, num_filters_total])

        # #3. DROPOUT LAYER ###################################################################
        with tf.name_scope("dropout"):
            self.h_drop = tf.nn.dropout(self.h_pool_flat, self.dropout_keep_prob)

        # Final (unnormalized) scores and predictions
        with tf.name_scope("output"):
            W = tf.get_variable(
                "W",
                shape=[num_filters_total, num_classes],
                initializer=tf.contrib.layers.xavier_initializer())
            b = tf.Variable(tf.constant(0.1, shape=[num_classes]), name="b")
            l2_loss += tf.nn.l2_loss(W)
            l2_loss += tf.nn.l2_loss(b)
            self.scores = tf.nn.xw_plus_b(self.h_drop, W, b, name="scores")
            self.predictions = tf.argmax(self.scores, 1, name="predictions")
```

```
        # CalculateMean cross-entropy loss
        with tf.name_scope("loss"):
            losses = tf.nn.softmax_cross_entropy_with_logits(logits=self.scores, labels=self.input_y)
            self.loss = tf.reduce_mean(losses) + l2_reg_lambda * l2_loss

        # Accuracy
        with tf.name_scope("accuracy"):
            correct_predictions = tf.equal(self.predictions, tf.argmax(self.input_y, 1))
            self.accuracy = tf.reduce_mean(tf.cast(correct_predictions, "float"), name="accuracy")


        print ("(!!) LOADED LSTM-CNN! :)")
```

```
import random
import sys
import os

file_name = "Sentiment Analysis Dataset.csv"
count = 1000

subscript = 1

while os.path.isfile('./good' + str(count) + '_' + str(subscript)):
    subscript += 1

t_file = list(open(file_name, 'r',encoding="utf8"))
good_file = open("good" + str(count) + '_' + str(subscript), 'a',encoding="utf8")
bad_file = open("bad" + str(count) + '_' +  str(subscript), 'a',encoding="utf8")

print("Opened file")

good_count = 0
bad_count = 0

while True:
    line = random.choice(t_file)
    line_split = line.split(',', 2)
    label = int(line_split[1])
    if label and good_count < count:
        good_file.write(line)
        good_count += 1
    elif not label and bad_count < count:
        bad_file.write(line)
        bad_count += 1
    elif bad_count >= count and good_count >= count:
        break
```

```
Opened file
```

Tuning the model:

```
import numpy as np
import time
import datetime
from tensorflow.contrib import learn

from IPython import embed

# Parameters
dev_size = .10

# Model Hyperparameters
embedding_dim  = 32     #128
max_seq_legth = 70
filter_sizes = [3,4,5]  #3
num_filters = 32
dropout_prob = 0.5 #0.5
l2_reg_lambda = 0.0
use_glove = True #Do we use glove

# Training parameters
batch_size = 128
num_epochs = 10 #200
evaluate_every = 100 #100
checkpoint_every = 100000 #100
num_checkpoints = 1 #Checkpoints to store

# Misc Parameters
allow_soft_placement = True
log_device_placement = False


# Data Preparation
filename = "Sentiment Analysis Dataset.csv"
goodfile = "good_file"
badfile = "bad_file"

# Load data
print("Loading data...")
x_text, y = get_dataset(goodfile, badfile, 5000)

#TODO: MAX LENGTH
# Build vocabulary
max_document_length = max([len(x.split(" ")) for x in x_text])
vocab_processor = learn.preprocessing.VocabularyProcessor(max_document_length)
x = np.array(list(vocab_processor.fit_transform(x_text)))

# Randomly shuffle data
np.random.seed(42)
shuffle_indices = np.random.permutation(np.arange(len(y)))
x_shuffled = x[shuffle_indices]
y_shuffled = y[shuffle_indices]

# Split train/test set
# TODO: This is very crude, should use cross-validation
dev_sample_index = -1 * int(dev_size * float(len(y)))
x_train, x_test = x_shuffled[:dev_sample_index], x_shuffled[dev_sample_index:]
y_train, y_test = y_shuffled[:dev_sample_index], y_shuffled[dev_sample_index:]
print("Vocabulary Size: {:d}".format(len(vocab_processor.vocabulary_)))
print("Train/Test split: {:d}/{:d}".format(len(y_train), len(y_test)))

#embed()
```

Training and Evaluation

```
with tf.Graph().as_default():
    session_conf = tf.ConfigProto(allow_soft_placement=True, log_device_placement=False)
    sess = tf.Session(config=session_conf)
    with sess.as_default():
        #embed()
        cnn = CNN_LSTM(x_train.shape[1],y_train.shape[1],len(vocab_processor.vocabulary_),embedding_dim,filter_sizes,num_filters,

        # Define Training procedure
        global_step = tf.Variable(0, name="global_step", trainable=False)
        optimizer = tf.train.AdamOptimizer(1e-3)
        grads_and_vars = optimizer.compute_gradients(cnn.loss)
        train_op = optimizer.apply_gradients(grads_and_vars, global_step=global_step)


        # Output directory for models and summaries
        timestamp = str(int(time.time()))
        out_dir = os.path.abspath(os.path.join(os.path.curdir, "runs", timestamp))
        print("Writing to {}\n".format(out_dir))

        # Summaries for loss and accuracy
        loss_summary = tf.summary.scalar("loss", cnn.loss)
        acc_summary = tf.summary.scalar("accuracy", cnn.accuracy)

        # Train Summaries
        train_summary_op = tf.summary.merge([loss_summary, acc_summary, grad_summaries_merged])
        train_summary_dir = os.path.join(out_dir, "summaries", "train")
        train_summary_writer = tf.summary.FileWriter(train_summary_dir, sess.graph)

        # Dev summaries
        dev_summary_op = tf.summary.merge([loss_summary, acc_summary])
        dev_summary_dir = os.path.join(out_dir, "summaries", "dev")
        dev_summary_writer = tf.summary.FileWriter(dev_summary_dir, sess.graph)

        # Checkpoint directory. Tensorflow assumes this directory already exists so we need to create it
        checkpoint_dir = os.path.abspath(os.path.join(out_dir, "checkpoints"))
        checkpoint_prefix = os.path.join(checkpoint_dir, "model")
        if not os.path.exists(checkpoint_dir):
            os.makedirs(checkpoint_dir)
        saver = tf.train.Saver(tf.global_variables(), max_to_keep=num_checkpoints)

        # Write vocabulary
        vocab_processor.save(os.path.join(out_dir, "vocab"))

        # Initialize all variables
        sess.run(tf.global_variables_initializer())

        #TRAINING STEP
        def train_step(x_batch, y_batch,save=False):
            feed_dict = {
              cnn.input_x: x_batch,
              cnn.input_y: y_batch,
              cnn.dropout_keep_prob: dropout_prob
            }
            _, step, summaries, loss, accuracy = sess.run(
                [train_op, global_step, train_summary_op, cnn.loss, cnn.accuracy],
                feed_dict)
            time_str = datetime.datetime.now().isoformat()
            print("{}: step {}, loss {:g}, acc {:g}".format(time_str, step, loss, accuracy))
            if save:
                train_summary_writer.add_summary(summaries, step)
```

```
#EVALUATE MODEL
def test_step(x_batch, y_batch, writer=None,save=False):
    feed_dict = {
      cnn.input_x: x_batch,
      cnn.input_y: y_batch,
      cnn.dropout_keep_prob: 0.5
    }
    step, summaries, loss, accuracy = sess.run(
        [global_step, dev_summary_op, cnn.loss, cnn.accuracy],
        feed_dict)
    time_str = datetime.datetime.now().isoformat()
    print("{}: step {}, loss {:g}, acc {:g}".format(time_str, step, loss, accuracy))
    if save:
        if writer:
            writer.add_summary(summaries, step)
```

```
#CREATE THE BATCHES GENERATOR
batches = gen_batch(list(zip(x_train, y_train)), batch_size, num_epochs)

#TRAIN FOR EACH BATCH
for batch in batches:
    x_batch, y_batch = zip(*batch)
    train_step(x_batch, y_batch)
    current_step = tf.train.global_step(sess, global_step)
    if current_step % evaluate_every == 0:
        print("\nEvaluation:")
        test_step(x_test, y_test, writer=dev_summary_writer)
        print("")
    if current_step % checkpoint_every == 0:
        path = saver.save(sess, checkpoint_prefix, global_step=current_step)
        print("Saved model checkpoint to {}\n".format(path))
```

```
2020-06-09T11:25:31.535280: step 6, loss 0.729867, acc 0.484375
2020-06-09T11:25:31.633950: step 7, loss 0.741072, acc 0.515625
2020-06-09T11:25:31.735301: step 8, loss 0.7322, acc 0.4375
2020-06-09T11:25:31.834570: step 9, loss 0.718511, acc 0.507812
2020-06-09T11:25:31.934737: step 10, loss 0.704298, acc 0.53125
2020-06-09T11:25:32.034172: step 11, loss 0.703349, acc 0.523438
2020-06-09T11:25:32.133007: step 12, loss 0.766637, acc 0.429688
2020-06-09T11:25:32.232775: step 13, loss 0.743313, acc 0.507812
2020-06-09T11:25:32.336906: step 14, loss 0.712604, acc 0.539062
2020-06-09T11:25:32.434848: step 15, loss 0.708759, acc 0.5
```

```
2020-06-09T11:30:23.514791: step 711, loss 0.312148, acc 0.867188
2020-06-09T11:30:23.609535: step 712, loss 0.254799, acc 0.890625
2020-06-09T11:30:23.706371: step 713, loss 0.273354, acc 0.875
2020-06-09T11:30:23.803893: step 714, loss 0.234794, acc 0.90625
2020-06-09T11:30:23.902675: step 715, loss 0.243684, acc 0.914062
2020-06-09T11:30:23.997558: step 716, loss 0.220353, acc 0.929688
2020-06-09T11:30:24.093292: step 717, loss 0.301725, acc 0.882812
2020-06-09T11:30:24.188089: step 718, loss 0.270537, acc 0.898438
2020-06-09T11:30:24.284098: step 719, loss 0.254042, acc 0.882812
2020-06-09T11:30:24.379178: step 720, loss 0.227556, acc 0.914062
2020-06-09T11:30:24.475443: step 721, loss 0.206239, acc 0.929688
2020-06-09T11:30:24.571039: step 722, loss 0.167103, acc 0.945312
2020-06-09T11:30:24.674397: step 723, loss 0.283056, acc 0.90625
2020-06-09T11:30:24.780246: step 724, loss 0.305147, acc 0.898438
2020-06-09T11:30:24.875837: step 725, loss 0.249959, acc 0.90625
2020-06-09T11:30:24.972689: step 726, loss 0.272652, acc 0.867188
2020-06-09T11:30:25.066789: step 727, loss 0.174683, acc 0.945312
```

Code for UI:

```
import tkinter as tk
import numpy as np
from keras.models import model_from_json
from keras import *
import os
from tkinter import *

root= tk.Tk()

canvas1 = tk.Canvas(root, width = 400, height = 300,  relief = 'raised')
canvas1.pack()

label1 = tk.Label(root, text='Sentiment Analysis')
label1.config(font=('helvetica', 14))
canvas1.create_window(200, 25, window=label1)

label2 = tk.Label(root, text='Enter your tweet:')
label2.config(font=('helvetica', 10))
canvas1.create_window(200, 100, window=label2)

entry1 = tk.Entry (root,bd=6,selectborderwidth=2,width=30)
canvas1.create_window(200, 140, window=entry1)
```

```
def analyse():
    global path2
    try:
        json_file=open('model.json','r')
        loaded_model_json=json_file.read()
        json_file.close()
        loaded_model=model_from_json(loaded_model_json)
        loaded_model.load_weights("model.h5")
        label=['POSITIVE','NEGATIVE']

        path2=filedialog.askopenfilename()
        x1 = entry1.get()
        result=loaded_model.predict(x1)

        new=tk.Toplevel(root)
        canvas1 = tk.Canvas(new, width = 400, height = 300,  relief = 'raised')
        canvas1.pack()

        lbl=label[result]
        label3.configure(text=lbl)

        label3 = tk.Label(new,font=('Comic Sans MS', 30,'bold'),fg='green')
        canvas1.create_window(180, 140, window=label3)

    except IOError:
        pass


button1 = tk.Button(text='Analyse', command=analyse, bg='brown', fg='white', font=('helvetica', 9, 'bold'))
canvas1.create_window(200, 180, window=button1)

root.mainloop()
```

7. CONCLUSION AND FUTURE WORK

In this project we have presented a model that aims to combine CNNs and LSTMs to achieve better performance in Sentiment Analysis tasks.The obtained results seem to indicate that by combining CNNs and LSTMs, we are able to harness both the CNN's ability in recognizing local patterns, and the LSTM's ability to harness the text's ordering.

The LSTM-CNN model seems to work because its initial LSTM layer seems to act as an encoder such that for every token in the input there is an output token that contains information not only of the original token, but all other previous tokens. Afterwards, the CNN layer will find local patterns using this richer representation of the original input, allowing for better accuracy.

The model that we present seem to perform better than regular CNN and LSTMs, and might be suitable for other tasks aside from sentiment analysis. It would be interesting to further apply these models for more complex tasks such as image analysis. Furthermore, it would be beneficial to test different types of Recursive Neural Networks aside from LSTMs for our models. For example, using bidirectional LSTMs on the LSTM-CNN model might yield an even better result as each token fed to the CNN layer would contain information from all the other tokens in the original input.

8. REFERENCES

- Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. In CS224N Project Report, Stanford, 2009.
- N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. In ACL, 2014.
- Y. Kim. Convolutional neural networks for sentence classification. In EMNLP, 2014.
- S. M. Mohammad, S. Kiritchenko, and X. Zhu. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In Semeval, 2013.
- Qurat Tul Ain , Mubashir Ali , Amna Riaz , Amna Noureen, Muhammad Kamran, Babar Hayat and A. Rehman.: Sentiment Analysis Using Deep Learning Techniques: A Review, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 8, No. 6, 2017.
- Alexander Pak, Patrick Paroubek. Twitter as a Corpus for Sentiment Analysis and Opinion Mining,2010.
- Saif, Hassan; He, Yulan and Alani, Harith (2012). Semantic sentiment analysis of twitter. In: The 11th International Semantic Web Conference (ISWC 2012), 11-15 Nov 2012, Boston, MA, USA.
- Vishal A. Kharde, S.S. Sonawane. Sentiment Analysis of Twitter Data: A Survey of Techniques, International Journal of Computer Applications (0975 – 8887) Volume 139 – No.11, April 2016.
- Shuichi Hashida, Keiichi Tamura, Tatsuhiro Sakai. Classifying Tweets using Convolutional Neural Networks with Multi-Channel Distributed Representation, IAENG International Journal of Computer Science, 46:1, IJCS_46_1_08 , February 2019.

- Fajar Ratnawati, Edi Winarko, Sentiment Analysis of Movie Opinion in Twitter Using Dynamic Convolutional Neural Network Algorithm, IJCCS (Indonesian Journal of Computing and Cybernetics Systems) Vol.12, No.1, January 2018.
- M. Iyyer, P. Enns, J. Boyd-Graber, P. Resnik 2014. Political Ideology Detection Using Recursive Neural Networks. In Proceedings of ACL 2014.
- Y. Shen, X. He, J. Gao, L. Deng, G. Mesnil. 2014. Learning Semantic Representations Using Convolutional Neural Networks for Web Search. In Proceedings of WWW 2014.
- S. Wang, C. Manning. 2012. Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. In Proceedings of ACL 2012.
- R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuglu, P. Kuksa. 2011. Natural Language Processing (Almost) from Scratch. Journal of Machine Learning Research 12:2493–2537.
- L. Dong, F. Wei, S. Liu, M. Zhou, K. Xu. 2014. A Statistical Parsing Framework for Sentiment Classification. CoRR, abs/1401.6330.