SENTIMENT ANALYSIS ON LIVE TWEETS BASED ON GEO-LOCATION.

12.02.2021

—

CHAITANYA KUMAR POTHURAJU
DINESH CHANDRA SAYANA
KIRANKUMAR DANTHALA

ACKNOWLEDGEMENT

Dr. SAYED SHAH of the COMPUTER SCIENCE AND ENGINEERING Department, whose work as project guide was critical to the project's success, deserves our deepest gratitude. We appreciate his genuine concern for our well-being, the books and reference materials he gave, and the spiritual guidance he offered.

Last but not least, we appreciate all of our instructors for passing on technical information that will always be useful to us.

TABLE OF CONTENTS

INTRODUCTION

In recent years, microblogging services such as Twitter have grown in popularity. As a result of this growth, companies and media organizations are increasingly seeking to mine Twitter for information about what people think and feel about their products and services.

Sentiment analysis is becoming a vital tool for modern business in order to understand public emotions and thereby taking business decisions based on sentiment analysis. It also helps in finding brand reputation,product success rate, product dissatisfaction in the public.

Another drawback of microblogging is the wide range of subjects that may be discussed. It is not an exaggeration to say that people tweet about everything and everything. As a result, we'll need a way to quickly identify data that can be used for training in order to build algorithms that can mine Twitter sentiment on any topic. Using Twitter hashtags (for example, #bestfeeling, #epicfail, #news) to discover positive, negative, and neutral tweets to train different sentiment classifiers is one technique we look at in this work.

People are increasingly using the internet to express themselves, and social media has a lot of opinion content. Sentiment analysis may be used to identify the polarity of opinions, such as whether they are favorable, negative, or neutral. Sentiment analysis has been shown to be efficient in gaining customer input on items, projecting election outcomes, and obtaining feedback from movie reviews for businesses. Companies may use the information obtained through sentiment analysis to make better decisions in the future.Many traditional sentiment analysis approaches employ the bag of words method. Because the bag of words method ignores morphology, two sentences containing the same bag of words may be mistakenly identified as having the same meaning. The relationship between a set of words is explored rather than the interaction between individual words.

As text data is enormously increasing in the modern era and with the advent of social media, data is increasing with rapid pace. This data contains public feelings, Opinions, reviews about products and many more emotions. So, we wanted to process this text data and find out the sentiments from the text. This application also helps businesses to make decisions. which will help us learn machine learning concepts as well as NLP.

Our sentiment analysis model is based on LSTM and also tweets were sorted according to the geographical locations, our tool allows for easy deployment without disrupting the workflow. Aspect based sentiment analysis is also one of the

features of our tool, this implies that for each written content, you may acquire various sentiment scores ranging from positive to negative. It's also incredibly configurable, so it'll be tailored just to your company's needs.

BACKGROUND

Many NLP problems like word, phrase, and document representation can be handled well by deep learning-based models. Researchers have begun working on neural networks for Tweet categorization, focusing on cutting-edge approaches. The average accuracy of neural network topologies is greater than that of classical classifiers. To improve sentiment categorization accuracy, manually collected features are combined with deep learning models. The deep learning model is built on a linear machine learning model and a word embedding model. In terms of F1-score, the suggested model outperformed baseline models. Collobert and Kuksa used convolutional filters to extract significant characteristics for sentiment categorization.

RNN- In typical neural networks, all inputs and outputs are independent of one another. However, in some circumstances, such as when predicting the next word of a phrase, the prior words are necessary, and so the previous words must be remembered (as the next word will depend on your previous input)

OBJECTIVES

The main objective of our project is to calculate the sentiments of tweets. We are considering extraction of the following sentiments which are SADNESS, ANGER, JOY, LOVE, SURPRISE, FEAR.

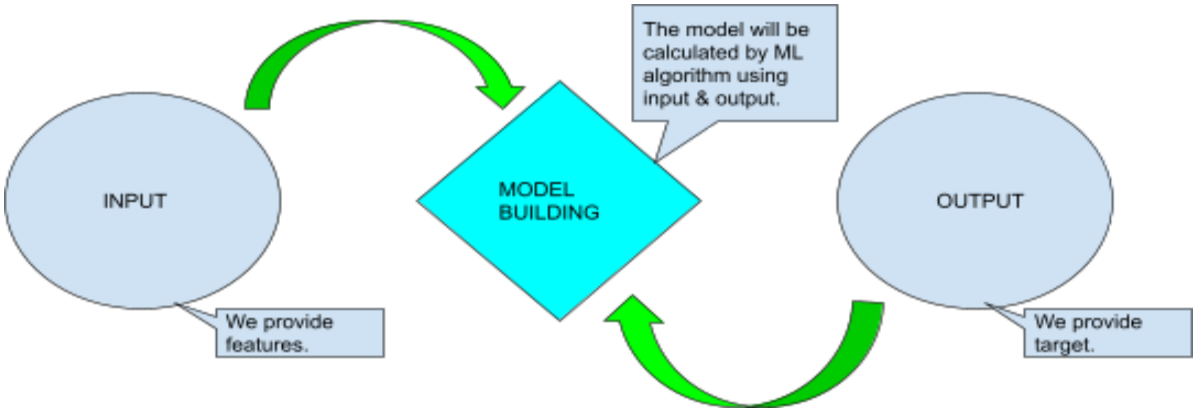
REQUIREMENTS AND TOOLS

- Numpy
- Scikit-learn
- Scipy
- Nltk
- Keras
- TensorFlow for Logistic Regression, RNN and LSTM
- Tweepy

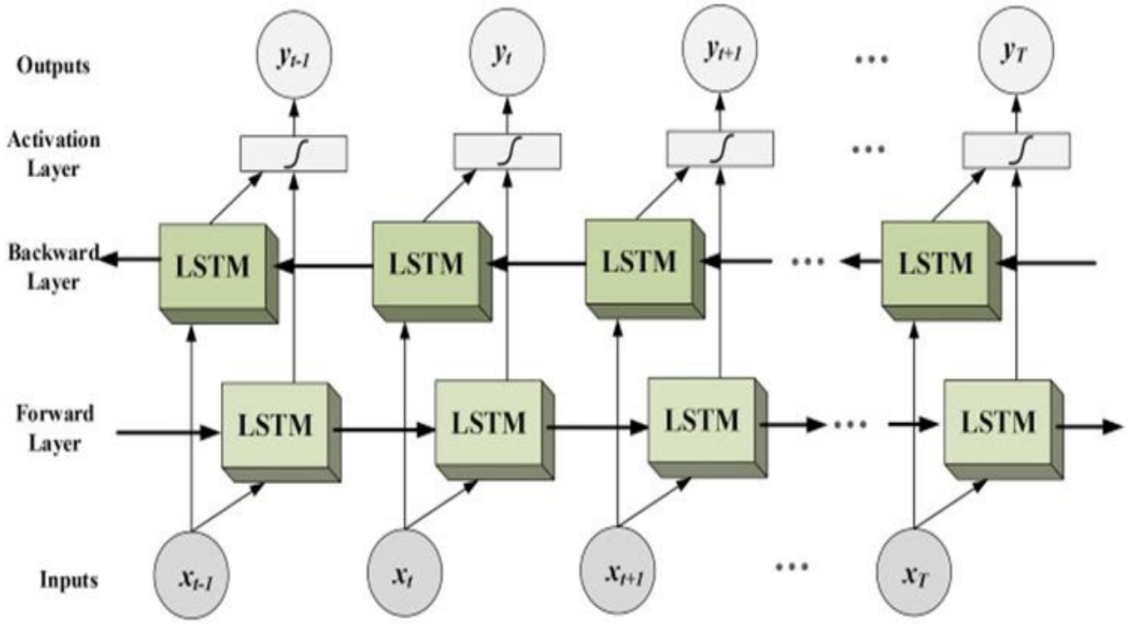
OUR MODEL

We wanted to build a classifier using bi-directional LSTM which predicts emotions of tweets(text). It is a supervised learning model.

The below provided diagram showcases how the model is built in machine learning.it takes features and targets as an input and process calculations and comes out with an equation which is nothing but a model which we use for classification.

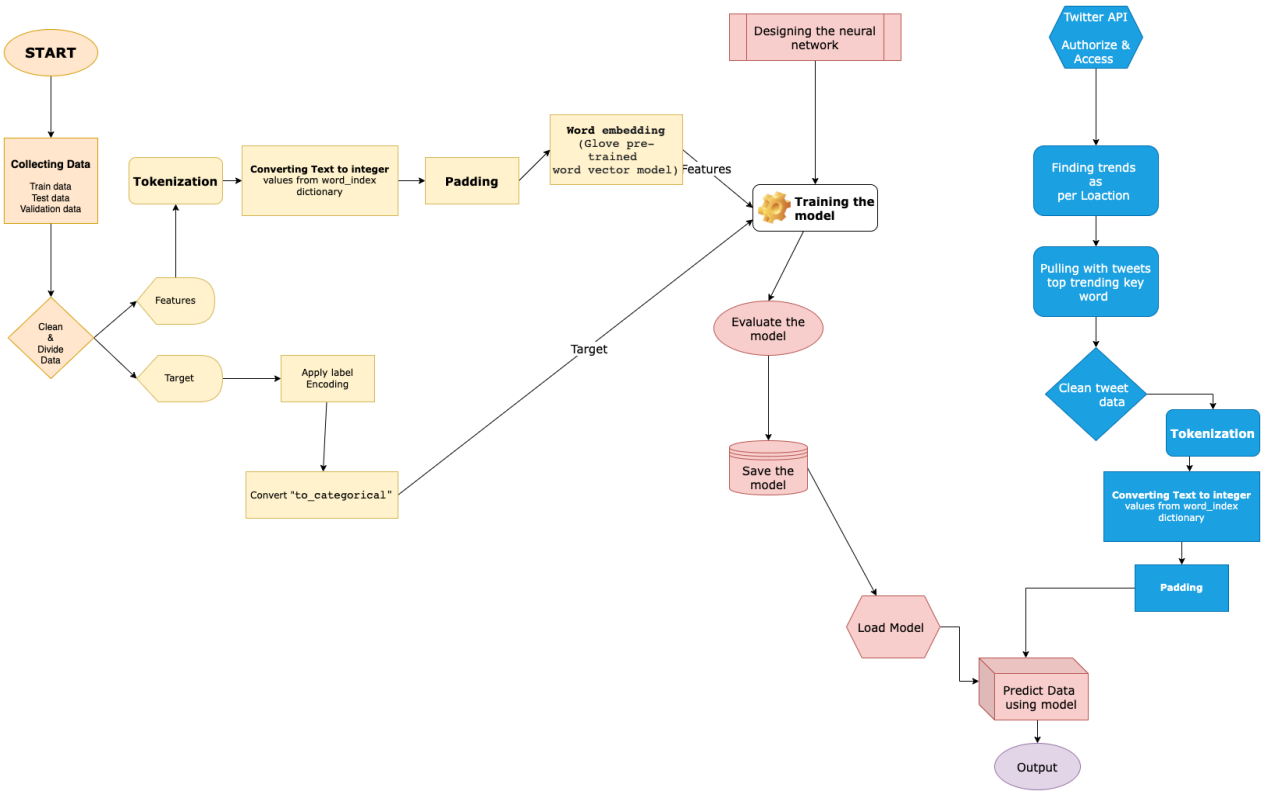


Bidirectional long-short term memory technique of allowing any neural network to store sequence information in both backwards and forwards orientations. Our input runs in two directions in a bidirectional LSTM, which distinguishes it from conventional LSTM. We can make input flow in one way, either backwards or forwards, with a normal LSTM. However, with bi-directional input, we can have the information flow in both directions, preserving both the future and the past.



Source - https://analyticsindiamag.com

PROJECT WORKFLOW DIAGRAM



First, we collected a dataset containing text as a feature and associated emotion as a label, then we cleaned the dataset to minimize errors and divided data into features and target as shown above in the flow chart.

Applied tokenization on the text sentence to break down into each word.

Tokenization

It is the process of breaking down a long string or sentence into smaller chunks. Phrases, keywords, symbols, and other tokens are examples of smaller bits. Tokens can be any amount of characters, or they can even represent a full sentence. During this operation, punctuation marks are deleted in particular.

Texts to sequences

Each text in a collection is converted into an integer sequence. As a result, it substitutes each word in the text with an integer value from the word index dictionary.

Padding

The inputs to all neural networks must be of the same form and size. However, not all phrases are the same length when we pre-process and use the texts as inputs for our model. To put it another way, some of the phrases are inherently longer or shorter. But we need the inputs to be the same size, which is where the padding comes in.

Word embedding

Word Embedding is a term used to represent the words for text analysis, which is often in the form of a real-valued vector that encodes the meaning of a word and predicts the meaning of words that are close in the vector space. We have used "glove pre trained word to vector model"

For target data as it is in text format we need to convert it into numeric. So, we have applied label encoder

It is an Encoding technique which is very popular in dealing with categorical variables. In this Alphabetical ordering will be given to the labels for converting them into numeric data so that machines can understand what the data is about and what machine needs to do with the data.

Applied to_categorical function on the target data. So, it created a numpy array with integers representing all the classes.

Designed a neural network with hidden layers

- Embedding layer
- Bi directional LSTM * 3
- Dense layer with softmax activation

Train the model by providing input sentences as a feature and an Emotion as a target/label.

Evaluate the model

save the model for the later use

Now we have to collect the twitter tweets and run the model on it to predict the emotions

1) Accessing twitter api finding out the trends as per location
2) Cleaning the tweets
3) Applying tokenization, text to sequence, padding
4) Load the stored model
5) Predict the tweet data with the model
6) output.

DATASET

We have taken a dataset which contains a sentence and associated emotion of it. We have taken 3 Datasets which are train, test & validation. The training data set contains 16000 records, the test dataset contains 2000 records.

Our dataset contains 6 emotions which are sadness, anger, love, surprise, fear and joy.

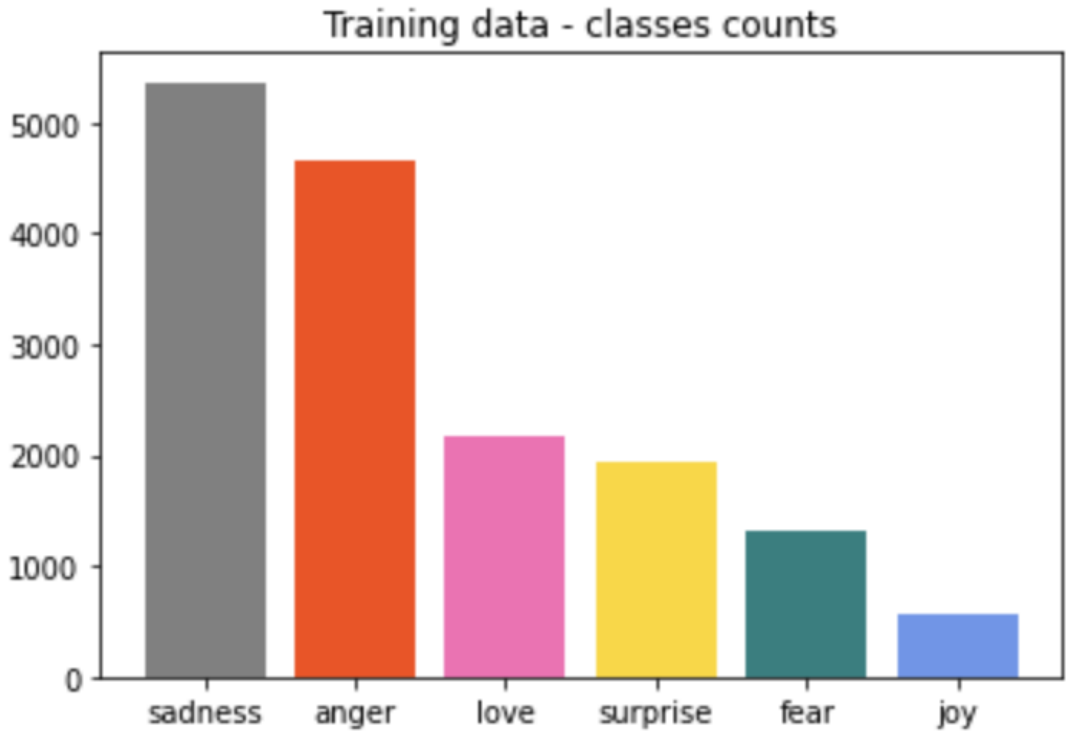
TRAINING DATASET

```
df_train.sample(5)
```

	Text	Emotion
5698	i have trouble in early afternoon and in the e...	sadness
11714	i feel i must write you owls until i am fearle...	joy
11591	i feel restless otherwise known as useless or ...	fear
11929	i know i dont live in new york anymore but i f...	anger
14434	i can offer you that feels loving to you	love

```
df_train.describe()
```

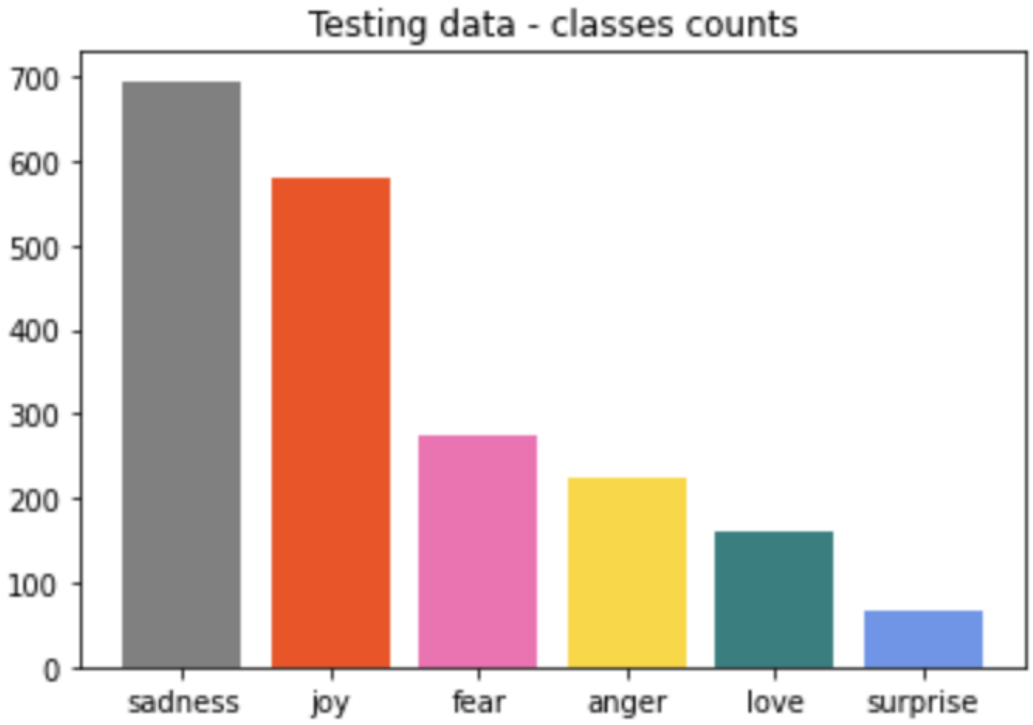
	Text	Emotion
count	16000	16000
unique	15969	6



TEST DATASET

```
df_test.describe()
```

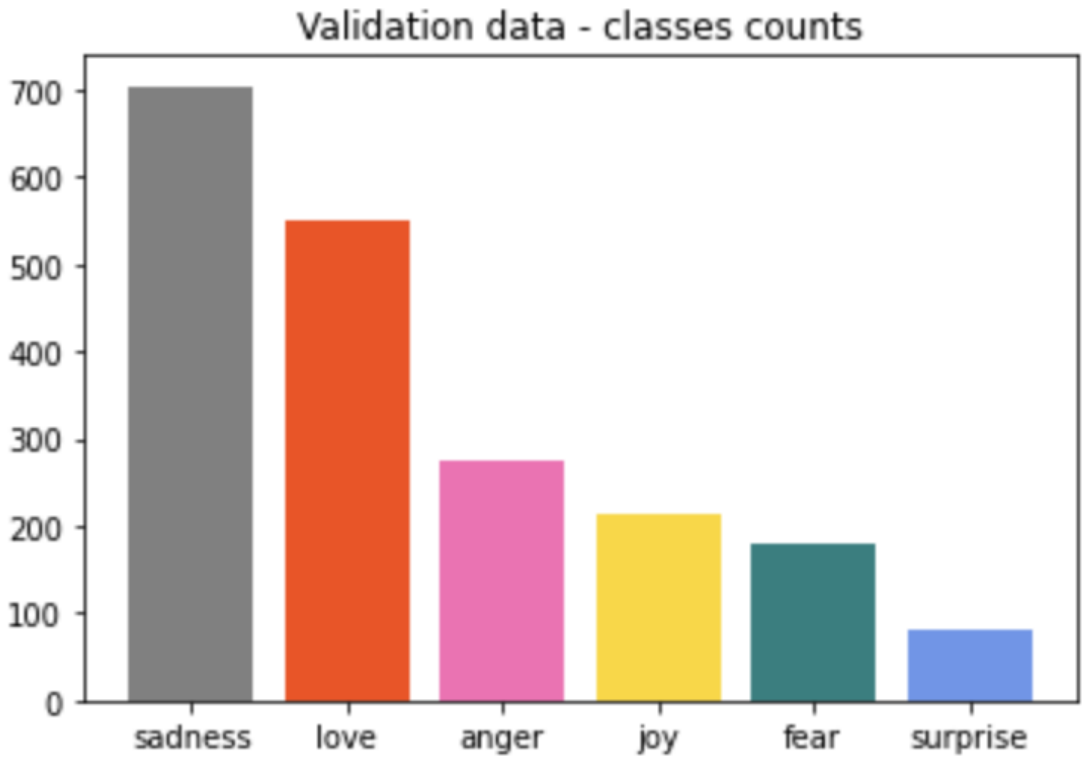
	Text	Emotion
count	2000	2000
unique	2000	6



VALIDATION DATASET

```
df_val.describe()
```

	Text	Emotion
count	2000	2000
unique	1998	6



The following dataset is obtained from kaggle and the link is provided below.

SOURCE - https://www.kaggle.com/praveengovi/emotions-dataset-for-nlp

ANALYSIS OF DATA

DATA PRE-PROCESSING

Data Preprocessing contains Cleaning of text, Transformation, Extracting the features and selection.

- Reading dataset into dataframes

```
df_train = pd.read_csv('/content/drive/MyDrive/nlp_project/train.txt', names=['Text', 'Emotion'], sep=';')
df_val = pd.read_csv('/content/drive/MyDrive/nlp_project/val.txt', names=['Text', 'Emotion'], sep=';')
df_test = pd.read_csv('/content/drive/MyDrive/nlp_project/test.txt', names=['Text', 'Emotion'], sep=';')
```

- Data Cleaning - Removing Punctuations

```
str_punc = string.punctuation.replace(',', '').replace("'",'')

def clean(text):
    global str_punc
    text = re.sub(r'[^a-zA-Z ]', '', text)
    text = text.lower()
    return text
```

- Dividing Data into features and target

```
X_train = df_train['Text'].apply(clean)
y_train = df_train['Emotion']

X_test = df_test['Text'].apply(clean)
y_test = df_test['Emotion']

X_val = df_val['Text'].apply(clean)
y_val = df_val['Emotion']
```

- Encoding Targets (Label Encoder)

```
le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_test = le.transform(y_test)
y_val = le.transform(y_val)

y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
y_val = to_categorical(y_val)
```

- Tokenizing and Converting Text to Sequence

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(pd.concat([X_train], axis=0))


sequences_train = tokenizer.texts_to_sequences(X_train)
sequences_test = tokenizer.texts_to_sequences(X_test)
sequences_val = tokenizer.texts_to_sequences(X_val)
```

- Padding the input to keep them in the same size

```
X_train = pad_sequences(sequences_train, maxlen=256, truncating='pre')
X_test = pad_sequences(sequences_test, maxlen=256, truncating='pre')
X_val = pad_sequences(sequences_val, maxlen=256, truncating='pre')
```

- Word Embedding - Converting Word to Vectors

  For which, we are using Glove pre-trained word to vector model

```
path_to_glove_file = '/content/drive/MyDrive/nlp_project/glove.6B.200d.txt'
num_tokens = vocabSize
embedding_dim = 200
hits = 0
misses = 0
embeddings_index = {}

# Read word vectors
with open(path_to_glove_file) as f:
    for line in f:
        word, coefs = line.split(maxsplit=1)
        coefs = np.fromstring(coefs, "f", sep=" ")
        embeddings_index[word] = coefs
print("Found %s word vectors." % len(embeddings_index))


# Assign word vectors to our dictionary/vocabulary
embedding_matrix = np.zeros((num_tokens, embedding_dim))
for word, i in tokenizer.word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        # Words not found in embedding index will be all-zeros.
        # This includes the representation for "padding" and "OOV"
        embedding_matrix[i] = embedding_vector
        hits += 1
    else:
        misses += 1
print("Converted %d words (%d misses)" % (hits, misses))
```

```
Found 400000 word vectors.
Converted 14212 words (1000 misses)
```

IMPLEMENTATION OF NEURAL NETWORKS

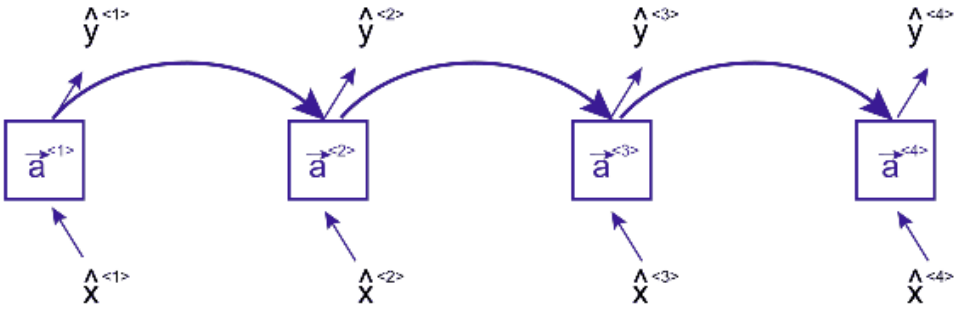
DESIGN OF NEURAL NETWORK

We designed the sequential neural network containing embedding layer as input, Bidirectional LSTM as hidden layers, Dense Layer as output and using softmax as activation function because we are classifying multi-label with 6 outputs.

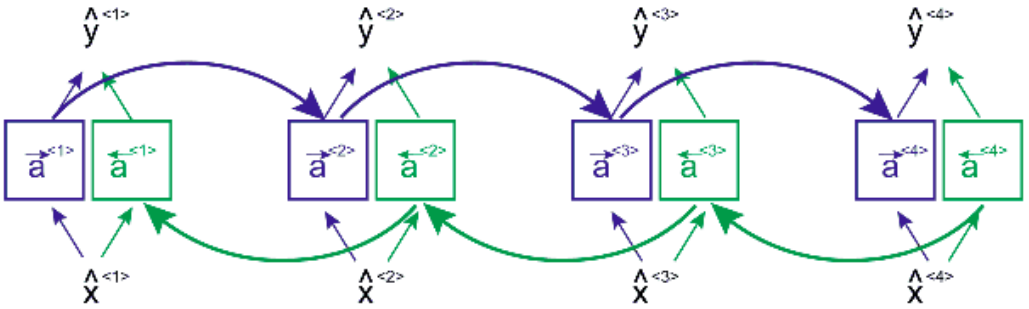
```
storedModel.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 256, 200)          3042600

 bidirectional (Bidirectiona  (None, 256, 512)         935936
 l)

 bidirectional_1 (Bidirectio  (None, 256, 256)         656384
 nal)

 bidirectional_2 (Bidirectio  (None, 256)              394240
 nal)

 dense (Dense)               (None, 6)                 1542

=================================================================
Total params: 5,030,702
Trainable params: 1,988,102
Non-trainable params: 3,042,600
```

Bidirectional LSTMs allow us to send the original data to the learning algorithm twice, once from beginning to finish and once from end to beginning. Bidirectional LSTM gives better results as it reads/learns sequence of data from both front and back.



Forward RNN (LSTM or GRU) network



Connecting the backward cells

$$\hat{y}^{<t>} = g(W_y [\overrightarrow{a}^{<t>}, \overleftarrow{a}^{<t>}] + b_y)$$

We are using cross entropy as a loss function to compute loss between labels and predication because we have multiple label classes so using cross.

```
model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])
```

TRAINING THE MODEL

Provided input (Features - X_train, Target - y_train)

Validating the model with testing data (X_test, y_test) and processing input batch size of '256' and given '10' epoch for model to be trained.

```
# Building model(fit) and providing test data for validation
history = model.fit(X_train,
                    y_train,
                    validation_data=(X_test, y_test),
                    verbose=1,
                    batch_size=256,
                    epochs=10

                   )
```

```
Epoch 1/10
63/63 [==============================] - 562s 9s/step - loss: 1.5598 - accuracy: 0.3695 - val_loss: 1.2736 - val_accuracy: 0.5325
Epoch 2/10
63/63 [==============================] - 548s 9s/step - loss: 1.1521 - accuracy: 0.5696 - val_loss: 0.8588 - val_accuracy: 0.6825
Epoch 3/10
63/63 [==============================] - 546s 9s/step - loss: 0.7080 - accuracy: 0.7402 - val_loss: 0.4515 - val_accuracy: 0.8270
Epoch 4/10
63/63 [==============================] - 546s 9s/step - loss: 0.3467 - accuracy: 0.8721 - val_loss: 0.2603 - val_accuracy: 0.9020
Epoch 5/10
63/63 [==============================] - 544s 9s/step - loss: 0.2276 - accuracy: 0.9131 - val_loss: 0.1973 - val_accuracy: 0.9215
Epoch 6/10
63/63 [==============================] - 540s 9s/step - loss: 0.1598 - accuracy: 0.9300 - val_loss: 0.1649 - val_accuracy: 0.9190
Epoch 7/10
63/63 [==============================] - 539s 9s/step - loss: 0.1302 - accuracy: 0.9401 - val_loss: 0.1587 - val_accuracy: 0.9245
Epoch 8/10
63/63 [==============================] - 543s 9s/step - loss: 0.1132 - accuracy: 0.9459 - val_loss: 0.1603 - val_accuracy: 0.9270
Epoch 9/10
63/63 [==============================] - 538s 9s/step - loss: 0.1007 - accuracy: 0.9527 - val_loss: 0.1440 - val_accuracy: 0.9225
Epoch 10/10
63/63 [==============================] - 536s 9s/step - loss: 0.0944 - accuracy: 0.9530 - val_loss: 0.1589 - val_accuracy: 0.9295
```

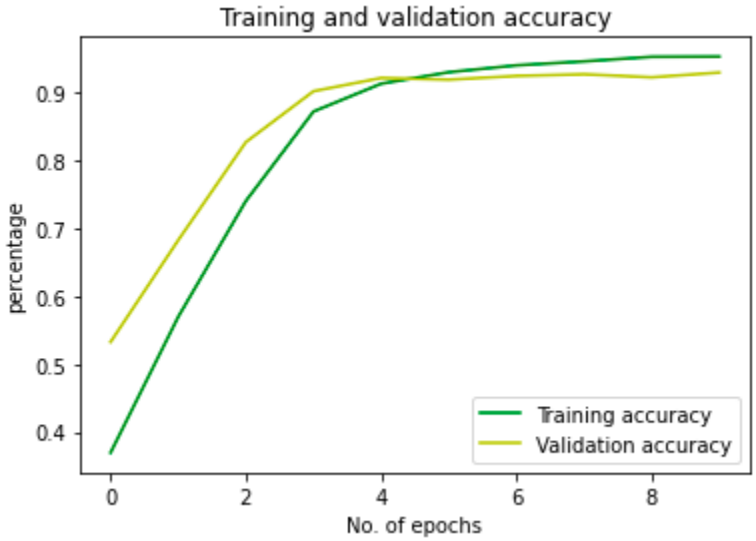
EVALUATING THE MODEL - PERFORMING EVALUATION OF MODEL WITH VALIDATION DATASET.

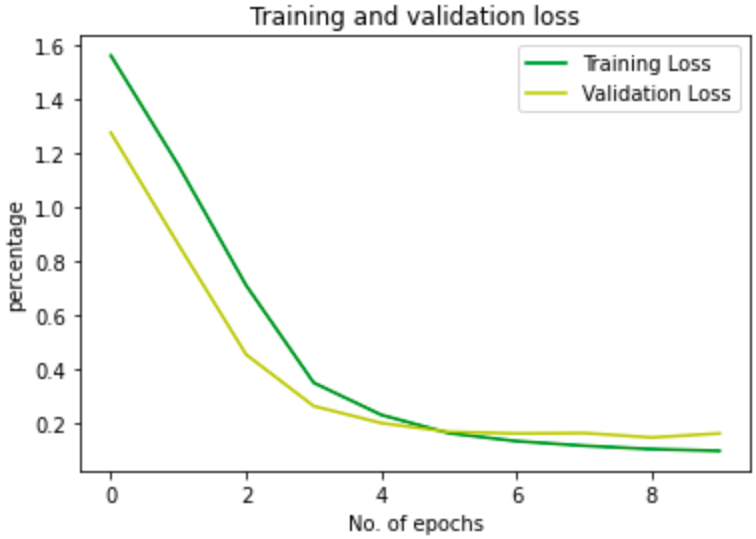
“ Accuracy : 93.30%, Loss : 14.17% “

```
# Evaluating the model with validation data set

model.evaluate(X_val, y_val, verbose=1)
```

```
63/63 [==============================] - 62s 989ms/step - loss: 0.1417 - accuracy: 0.9330
[0.14173798263072968, 0.9330000281333923]
```

PLOTTING THE ACCURACY and LOSS VALUES





Below section is related to fetching tweets and predicting emotion of tweets using the above built model.

```
import tweepy
import pandas as pd
```

Importing packages which are necessary to build framework and tweepy is used to access twitter API.

```
api_key = "LeRJxmWPxG0dBcPoDThRqmAoB"
api_key_secret = "OUpsJV4oMmcBxVZ7sSkXhrPyxhxo6PzEAZQhrnhkk7b9x0mKHt"

access_token = "3482308459-o1cw43udRKgLuU5hb1QJ8DnlMcDvPOYKxrKTCe5"
access_token_secret = "yIboNQP1gqljX6ZHnETmKwBKq3V5eIWVPUGnYwIoTT0IU"
```

Getting API keys from twitter developer account in order to access the data directly from twitter.

```
auth = tweepy.OAuthHandler(consumer_key=api_key, consumer_secret=api_key_secret)
auth.set_access_token(access_token,access_token_secret)

api = tweepy.API(auth)
```

Authorizing the keys that are acquired from the developer account in order to secure the data from twitter userhandle.

```
# For GeoLocation

woeid=23424748  # Taking Austrial as location
```

Every country holds a unique code, We are using it in order to find trends in the location.

```
# Accessing trends topic of particular geolocation

trend_result = api.trends_place(woeid)
```

Getting all the trend topics based on geolocation

```
for trend in trend_result[0]["trends"][:10]:
  print(trend['name'])
```

Displaying Top 10 trends in the location.

```
#WorldAIDSDay
#JusticeForRailwayStudents
#wednesdaythought
#BSFRaisingDay
#FarmersFighting4Rights
श्री गणेश
सीमा सुरक्षा
Nagaland
Happy Anniversary
Border Security Force
```

From top trending we are taking the top most trend and filtering tweets with that keyword.

```
search_words = top + " -filter:retweets"
```

Now we filter the tweets like whichever the tweets related to the top trending topic. And also filtering retweets which may distract the emotions of the topic.

```
tweets = tweepy.Cursor(api.search,
                       q=search_words,
                       lang="en").items(1000)
```

Now we pull the tweets containing top trending topic as key words in the tweet. Here we are taking 1000 recent tweets.

Storing the retrieved data into pandas dataframe.

```
df
```

	Tweet
0	To my #1 podcast of 2021, ASMR Therapy.: thank...
1	So about #SpotifyWrapped - why can I not searc...
2	Thank you Moshiach Oi! for spending 261 minute...
3	tis the season to listen to my #SpotifyWrapped...
4	To be honest I don't know how Lil Wayne and As...
...	...
995	Thank you @yungblud for spending 6,686 minutes...
996	To my #1 podcast of 2021, Love Letters: thank ...
997	I spent a total of 20,720 minutes listening to...
998	Can't say this shocks me at all. Top 3 songs a...
999	Thank you @coldplay for spending 1,545 minutes...

1000 rows × 1 columns

These are some tweets that we pulled from top trending tweets.

As we can see the # tags are shown along with the tweet and a link to verify whether the tweet is real or not.

As shown in the above picture, we have gathered thousand tweets.

The obtained tweets contains emoticons , Hashtags, @mentions and punctuation. So, we need to clean and make data free from unnecessary data like emoticons,#hashtags ..etc. So, that text will be classified more accurately.

Removing emoticons from the text

```
def remove_emojis(data):
    emoj = re.compile("["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
        u"\U00002500-\U00002BEF"  # chinese char
        u"\U00002702-\U000027B0"
        u"\U00002702-\U000027B0"
        u"\U000024C2-\U0001F251"
        u"\U0001f926-\U0001f937"
        u"\U00010000-\U0010ffff"
        u"\u2640-\u2642"
        u"\u2600-\u2B55"
        u"\u200d"
        u"\u23cf"
        u"\u23e9"
        u"\u231a"
        u"\ufe0f"  # dingbats
        u"\u3030"
                      "]+", re.UNICODE)
    return re.sub(emoj, '', data)
```

```
] def text_cleaning(text):
    text = re.sub(r'@[A-Za-z0-9]+', '', text)      # removing @mentions
    text = re.sub(r'@[A-Za-zA-Z0-9]+', '', text)  # removing @mentions
    text = re.sub(r'@[A-Za-z]+', '', text)         # removing @mentions
    text = re.sub(r'@[-)]+', '', text)             # removing @mentions
```

Now we clean the tweets by removing mentions first.

```
    text = re.sub(r'#', '', text )                 # removing '#' sign
```

#tags are removed

```
text = re.sub(r'RT[\s]+', '', text)           # removing RT
text = re.sub(r'https\S+', '', text)          # removing https:
text = re.sub(r'https?\/\/\S+', '', text)     # removing the hyper link
text = re.sub(r'&[a-z;]+', '', text)          # removing '&gt;'
text = re.sub(r'\n', '',text )                # removing '\n'
text = re.sub(r',', '',text)                  # removing ","
text = re.sub(r'\"','',text)                   # removing "."
return text
```

From the tweets all hyperlinks and different punctuation marks are removed so that tweets will be clean and will be easy to detect emotion.

```
df
```

	Tweet	tweet_clean
0	To my #1 podcast of 2021, ASMR Therapy.: thank...	To my podcast of ASMR Therapy.: thank you fo...
1	So about #SpotifyWrapped - why can I not searc...	So about SpotifyWrapped - why can I not search...
2	Thank you Moshiach Oi! for spending 261 minute...	Thank you Moshiach Oi! for spending minutes w...
3	tis the season to listen to my #SpotifyWrapped...	tis the season to listen to my SpotifyWrapped ...
4	To be honest I don't know how Lil Wayne and As...	To be honest I don't know how Lil Wayne and As...
...	...	...
995	Thank you @yungblud for spending 6,686 minutes...	Thank you for spending minutes with me this ...
996	To my #1 podcast of 2021, Love Letters: thank ...	To my podcast of Love Letters: thank you for...
997	I spent a total of 20,720 minutes listening to...	I spent a total of minutes listening to Lemon...
998	Can't say this shocks me at all. Top 3 songs a...	Can't say this shocks me at all. Top songs ar...
999	Thank you @coldplay for spending 1,545 minutes...	Thank you for spending minutes with me this ...

1000 rows × 2 columns

The cleaned tweets were added to a new column and it is evident that clean tweets don't have #tags, @ mentions and no emoticons.

Predicting Tweets emotion with the above built model, First we need to tokenize and change texts to integer values and apply proper padding to tweet data. And provide data to model for prediction.

```
result_list = []
proba_list = []


for sentence in df['tweet_clean']:
  sentence = clean(sentence)
  sentence = tokenizer.texts_to_sequences([sentence])
  sentence = pad_sequences(sentence,maxlen=256, truncating='pre')
  result = le.inverse_transform(np.argmax(storedModel.predict(sentence),axis=-1))[0]
  proba = np.max(storedModel.predict(sentence))
  result_list.append(result)
  proba_list.append(proba)
```

Each Tweet has been predicted and classified with associated emotion and also the probability of prediction.

```
df
```

	Tweet	tweet_clean	predicted_emotion	probability
0	To my #1 podcast of 2021, ASMR Therapy.: thank...	To my podcast of ASMR Therapy.: thank you fo...	joy	0.395654
1	So about #SpotifyWrapped - why can I not searc...	So about SpotifyWrapped - why can I not search...	joy	0.523200
2	Thank you Moshiach Oi! for spending 261 minute...	Thank you Moshiach Oi! for spending minutes w...	anger	0.353740
3	tis the season to listen to my #SpotifyWrapped...	tis the season to listen to my SpotifyWrapped ...	joy	0.533431
4	To be honest I don't know how Lil Wayne and As...	To be honest I don't know how Lil Wayne and As...	fear	0.632899
...	...	...	...	...
995	Thank you @yungblud for spending 6,686 minutes...	Thank you for spending minutes with me this ...	anger	0.353740
996	To my #1 podcast of 2021, Love Letters: thank ...	To my podcast of Love Letters: thank you for...	joy	0.402875
997	I spent a total of 20,720 minutes listening to...	I spent a total of minutes listening to Lemon...	anger	0.324238
998	Can't say this shocks me at all. Top 3 songs a...	Can't say this shocks me at all. Top songs ar...	anger	0.396442
999	Thank you @coldplay for spending 1,545 minutes...	Thank you for spending minutes with me this ...	anger	0.353740

1000 rows × 4 columns

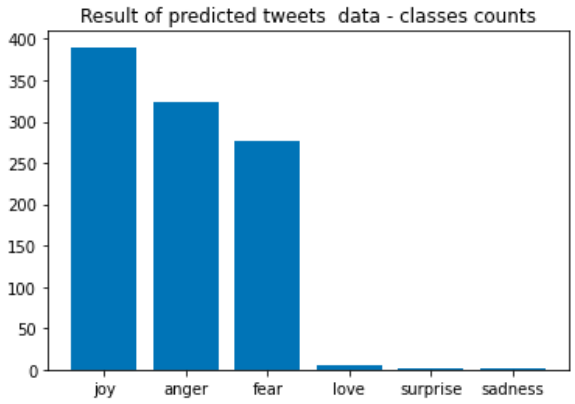
Analyzing the output - predicted emotions

```
df['predicted_emotion'].value_counts()
```

```
anger       390
joy         324
fear        276
surprise      6
love          2
sadness       2
Name: predicted_emotion, dtype: int64
```

Visualization of output

```
plt.bar(df['predicted_emotion'].unique(),height=df['predicted_emotion'].value_counts())
plt.title("Result of predicted tweets  data - classes counts")
plt.show()
```



IMPLEMENTATION STATUS REPORT

Work completed

SINCE OUR PROJECT IS DEPENDING ON EMOTION DETECTION, WE HAVE FULFILLED ALL THE REQUIRED WORK AND THE PREDICTION WORKS GOOD WITH ALMOST

DATASET COLLECTION, DATA PREPROCESSING, FEATURE EXTRACTION, MODEL BUILDING, MODEL EVALUATION.

Description

DATA HAS BEEN PREPROCESSED FIRST AFTER TAKING IT FROM KAGGLE LIKE REMOVING SYMBOLS, EMOJIS AND REPEATED CHARACTERS. LATER WE ARE PERFORMING THE MODEL BUILDING AND ITS EVALUATION.

Responsibility (Task,Person)

CHAITANYA POTHURAJU - DATA PRE-PROCESSING, MODEL BUILDING, MODEL EVALUATION

DINESH CHANDRA SAYANA - DATA PRE-PROCESSING, MODEL BUILDING, MODEL EVALUATION

KIRAN KUMAR DANTHALA - DATASET SELECTION, TWEET COLLECTION AND CLEANING.

Contributions (members/percentage)

CHAITANYA POTHURAJU - 35%

DINESH CHANDRA SAYANA - 35%

KIRANKUMAR DANTHALA - 30%

Issues/Concerns

CONFLICT OF INTEREST

All three members in the team have no conflict of interest. We believed in everything that we worked for and the model we developed is a product of all three members' hard work as we three fit perfectly in the development of this project.

CONCLUSION

The growing popularity of social media platforms created an environment suited to microblogging, in which individuals may communicate their ideas and opinions on any topic to a large audience. Twitter is a popular social media network that provides a wealth of data for sentiment analysis.

Although sentiment analysis of tweets is similar to sentiment analysis of general texts, the problem specific limits allow for innovative and imaginative solutions for this sort of writing. Several strategies have been presented as a result of the widespread use of extracting sentiments from tweets.

We found that 256 is the ideal number of hidden units based on the number of hidden layers and units per layer that we evaluated. The distribution of these units did not have to be predefined, as we discovered. A network with one layer of 256 units functions similarly to one with two levels of 128 units. Increasing the number of layers while maintaining the number of units per layer increased the model's performance on the training set, but not on the testing set. This revealed that extending the network size beyond the best-performing size we determined did not

result in improved performance outside of the data provided to the model during training.

We tried a few other architectural approaches, but ultimately settled on the recurrent neural network design and fundamental cells that had already been published. Additional recurrent cells or even various algorithms would be examined in a more extensive examination to determine how they compare.

We're working hard to improve the memory cell architecture in recurrent neural networks. We attained accuracy of 93.34 percent using the regular BI-LSTM, but we feel that a unit specifically created for our purpose would be able to extract the emotion more correctly. When compared to previous publications using this dataset as a corpus for sentiment analysis, our study shows a clear experimental design, creating a new benchmark for future developments.

REFERENCES

- A.Pak and P. Paroubek. „Twitter as a Corpus for Sentiment Analysis and Opinion Mining". In Proceedings of the Seventh Conference on International Language Resources and Evaluation, 2010, pp.1320-1326
- Agarwal, B. Xie, I. Vovsha, O. Rambow, R. Passonneau, "Sentiment Analysis of Twitter Data", In Proceedings of the ACL 2011Workshop on Languages in Social Media,2011 , pp. 30-38
- Neethu M,S and Rajashree R," Sentiment Analysis in Twitter using Machine Learning Techniques" 4th ICCCNT 2013,at Tiruchengode, India. IEEE – 31661

www.kaggle.com

www.tensorflow.com

www.keras.com

GithubLink:

https://github.com/chay2021/sentiment-analysis/blob/main/NLP%20Increment-1%20submission.docx