



Analysis of Sentiment in Text Utilizing the Twitter Dataset

Bunty Prasad Nayak, Raman Chadha and Devansh Tiwari

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

February 20, 2023

Analysis of Sentiment in Text Utilizing the Twitter Dataset

¹Bunty Prasad Nayak, ²Dr. Raman Chadha, ³Devansh Tiwari

¹3rd Year, Computer Science & Engineering Department
UIE, Chandigarh University, Mohali, India
buntyprasadnayak60@gmail.com

²Professor, Computer Science & Engineering Department
UIE, Chandigarh University-Punjab
dr.ramanchadha@gmail.com

³3rd Year, Computer Science & Engineering Department
UIE, Chandigarh University, Mohali, India
devanshtiwari0012@gmail.com

Abstract-The computational examination of people's opinions, sentiments, attitudes, and emotions as they are represented in written language is known as sentiment analysis or opinion mining. Two factors are the key causes of its appeal. The fact that opinions are fundamental to practically all human endeavors and are significant determinants of our behavior means that it has a wide range of applications. We seek out other people's perspectives whenever we need to make a decision. Second, it poses a number of difficult research issues that had never been addressed before to the year 2000. Prior until now, there wasn't much opinionated text available in digital forms, which contributed to the lack of research. Thus, it should come as no surprise that the emergence and explosive expansion of the area parallel those of online social media. Due to its significance for industry and society at large, research on this topic has really moved beyond computer science to include management sciences and social sciences. I will introduce mainstream sentiment analysis research at the outset of this session before moving on to detail some recent work on modeling comments, discussions, and debates, which is another type of sentiment analysis and opinion analysis.

Keywords: *Machine Learning, Text Sentiment Analysis, LSTM, NLP, Tensor-flow,*

1. INTRODUCTION

In this study, we examine the widely used microblog Twitter and develop models for categorizing "tweets" into positive and negative emotion. For two classification tasks, we develop models: Categorizing sentiment into positive and negative categories is a binary task. The goal of this project is to classify tweets to determine whether they are expressing personal opinions or not. One way to measure this is by looking at the words used in the tweet- if there are a great deal of adjectives, adverbs, and nouns it's likely an opinion. If the word "I" is used more than once then it's most likely an opinion. The classification can be determined by a machine learning algorithm that's built off these rules. Create a function that can classify Tweets by the topic it is about. For example, if someone tweets about sports, then it is classified as sports. Build a function that can classify Tweets by its mood. For example, if someone tweets happy or sad, then it is classified as happy or sad.

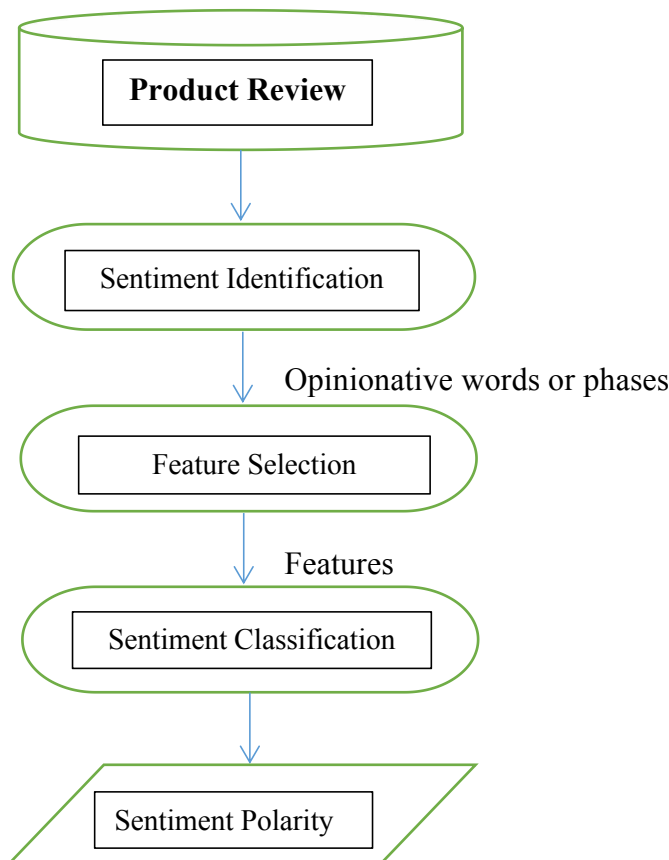
Data Mining is an analysis process used remove hidden patterns from available data sources. Using the contrary, a big data problem can be solved better by advanced data analysis process. Data mining is not individual process but is an important part of Knowledge Data acquisition process (KDD).

Machine learning- The data available is increasing day by day and such a huge amount of unprocessed data

is needed to be analyzed precisely, as it can give very informative and finely pure gradient results as per current standard requirements. It is not wrong to say as with the evolution of Artificial Intelligence (AI) over the past two decades, Machine Learning (ML) is also on a fast pace for its evolution. ML is an important mainstay of IT sector and with that, a rather central, albeit usually hidden, part of our life [1]. As the technology progresses, the analysis and understanding of data to give good results will also increase as the data is very useful in current aspects

NLP- NLP makes it possible for computers to comprehend natural language just like people do. Natural language processing use artificial intelligence to take real-world input, process it, and make sense of it in a way that a computer can comprehend, regardless of whether the language is spoken or written. Computers have reading programs and microphones to collect audio, much as people have various sensors like ears to hear and eyes to see. Computers have a programmed to process their various inputs, just as humans have a brain to do so. The input is eventually translated into computer-readable code during processing.

TensorFlow - A complete open-source machine learning platform is called TensorFlow. Researchers can advance the state-of-the-art in machine learning thanks to its extensive, adaptable ecosystem of tools, libraries, and community resources, while developers can simply create and deploy ML-powered applications.



2. METHODOLOGY



Fig.1

Our methodology in this project is to analyze a piece of text to discover the sentiment hidden. It accomplishes this by combining machine learning and natural language processing (NLP). Sentiment analysis allows you to examine the feelings expressed in a piece of text.

we build a binary text classifier to classify the sentiment behind the text. We use the various NLP preprocessing techniques to clean the data and utilize the LSTM layers to build the text classifier. We transform our text data into something that our machine learning model understands. Basically, we need to convert the text into an array of vector embedding. Word embedding are a beautiful way of representing the relationship between the words in the text.

In this machine learning project, we built a binary text classifier that classifies the sentiment of the tweets into positive and negative. We obtained more than 83% accuracy.

Tools and Library used

- Python – 3.x
- Pandas – 1.2.4
- Matplotlib – 3.3.4
- TensorFlow – 2.4.1

3. LITERATURE SURVEY

Research on sentiment analysis or opinion learning began at the turn of the 20th century. With the passage of time, many new innovations and ideas were proposed in this field. Today this topic is a high-priority topic because most e-cart websites and many social media platforms are using this for their betterment. some of the early recent works on sentiment analysis (wala Medhat a, Ahmed Hassan b , Hoda Korashy)[1] use various techniques like supervised learning under which they used probabilistic classifiers which use mixtures for the classification. The Naive Bayes classifier, which is the most basic and widely used classifier for the classification process, was the major tool they utilized. This model works with BOW features extraction and ignores the location of the words. And uses the Bayes theorem for the prediction purpose. Kang and Yoo [2] suggested an enhanced Naive Bayes classifier to address the tendency problem, where positive classification accuracy was 10% more than negative classification accuracy. And due to this, the problem was coming that the average accuracy was coming to be in decreased manner. They demonstrated how the difference between positive and negative accuracy might be reduced by combining their algorithm with customer reviews of local eateries. The overall average accuracy of the Model was slightly increased.

At various granularities, sentiment analysis has been treated as a task in natural language processing. Beginning as a classification task at the document level (Turney, 2002; Pang and Lee, 2004)[3], it has been handled at the phrase level more recently (Hu and Liu, 2004; Kim and Hovy, 2004)[4] than at the sentence level (Wilson et al., 2005; Agarwal et al,2009)[5]. Microblog data provides fresh and distinct issues

because users can post real-time comments and views about "anything" on sites like Twitter. Pak and Paroubek[6] are some of the authors of early and current findings on sentiment analysis of Twitter data (2010) collect sentiment data using remote learning. They create models using Support Vector Machines (SVM), Naive Bayes, and MaxEnt, and they claim SVM performs better than other classifiers. They experiment with a Unigram, Bigram model in conjunction with parts-of-speech (POS) characteristics in the feature space. They point out that the unigram model performs better than every other model. Bigrams and POS characteristics, in particular, are ineffective. Data collection is done using a similar distance learning approach as used by Pak and Paroubek (2010)[6]. However, they carry out a distinct kind of classification task: subjective as opposed to objective. In order to gather subjective data, they follow the same procedure as and collect tweets that finish in emoticons .

A further addition to this was carried out by Doaa Mohey El-Din Mohamed Hussein [7]. They used the Lexicon Based Approach which relies on finding the opinion Lexical, that is used to analyse the text. They divide the challenges and focus on the degree of accurate meaning. They use the data set where several lexicons were available as sentiments. The used lexicon was having the sentiment word and polarity. This classification was divided into several sub-classes like 1.) two levels (positive one and the negative one), 2.) Three levels like in the hierarchical level, or 3.) four levels (negative, positive, neutral, and mixed). Further, this lexicon model follows two approaches- 1.) Dictionary Based approach in which a small number of words of opinion with established opinions are collected manually (Medhat , 2014)[8], this collection will gradually decrease until no new terms are introduced in it. This method has the inconvenience of depending on the scale of the dictionary scale and as the size of the dictionary increase, the approach goes wrong. (Jain & Dandannavar, 2016)[9]. And the other approach is 2.) Corpus-Based Approach This approach relies on massive syntactic and semantic patterns. Words that are created using this are context-specific and it needs a large data set labelled (Jain & Dandannavar, 2016) [9].

4. RESOURCE AND DATA PREPROCESSING

As we are dealing with the text data, we need to preprocess it using word embeddings. We need to import the necessary libraries.

4.1 Data Exploration-

Firstly, we imported the data set from the google drive and then we store it in dataframe(tweet_data) and start exploring it.

We don't really need neutral reviews in our dataset for this binary classification problem. So, drop those rows from the dataset.

```
tweet_data.drop(tweet_data.head(551424).index, inplace=True)
usless_columns = ["ids", "date", 'user', 'flag']
tweet_data = drop_columns( tweet_data, usless_columns)

#this will show the number of categories in the target value
tweet_data['target'].value_counts()
```

| | |
|---|--------|
| 0 | 248576 |
| 1 | 248576 |

Name: target, dtype: int64

There are more than 10,00,000 data samples in the sentiment analysis Dataset and the function shape give the no. of columns and rows present in the dataset.

```
✓ [6]
0s tweet_data.shape

(1048576, 6)
```

In dtypes function result is a Series with each column's data type listed in it. The columns of the initial DataFrame serve as the result's index.

```
✓ [7]
0s tweet_data.dtypes

target    int64
ids       int64
date      object
flag      object
user      object
text      object
dtype: object
```

The DataFrame's information is printed via the info() method. The data includes the total number of columns, their labels, data kinds, memory use, range index, and the number of cells in each column (non-null values).

```
✓ [5] tweet_data.info()
0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048576 entries, 0 to 1048575
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  -
 0  target  1048576 non-null  int64
 1  ids     1048576 non-null  int64
 2  date    1048576 non-null  object
 3  flag    1048576 non-null  object
 4  user    1048576 non-null  object
 5  text    1048576 non-null  object
dtypes: int64(2), object(4)
memory usage: 48.0+ MB
```

The DataFrame's data is described via the describe() method. If the DataFrame includes numerical data, each column's description will provide the following details:-count is the total amount of non-empty values. mean - A measure of average value. the standard deviation, or std min is the lowest value.

25% – It is the 25 percentiles*.

50% – It is the 50 percentiles*.

75% – It is the 75 percentiles*.

max is the highest value.

```
✓ [10] tweet_data.isnull().sum()
0s

target    0
ids       0
date      0
flag      0
user      0
text      0
dtype: int64
```

isnull() function is used to check Null values present in the dataset.

```
tweet_data.head()
```

| | target | ids | date | flag | user | text |
|---|--------|------------|------------------------------|----------|-----------------|---|
| 0 | 0 | 1467810369 | Mon Apr 06 22:19:45 PDT 2009 | NO_QUERY | _TheSpecialOne_ | @switchfoot http://twitpic.com/2y1zl - Awww, t... |
| 1 | 0 | 1467810672 | Mon Apr 06 22:19:49 PDT 2009 | NO_QUERY | scotthamilton | is upset that he can't update his Facebook by ... |
| 2 | 0 | 1467810917 | Mon Apr 06 22:19:53 PDT 2009 | NO_QUERY | mattycus | @Kenichan I dived many times for the ball. Man... |
| 3 | 0 | 1467811184 | Mon Apr 06 22:19:57 PDT 2009 | NO_QUERY | ElleCTF | my whole body feels itchy and like its on fire |
| 4 | 0 | 1467811193 | Mon Apr 06 22:19:57 PDT 2009 | NO_QUERY | Karoli | @nationwideclass no, it's not behaving at all... |

The labels for this dataset are categorical. Machines understand only numeric data. So, convert the categorical values to numeric using the factorize() method. This returns an array of numeric values and an Index of categories.

```
[30] sentiment_label = tweet_data.target.factorize()

#machine only understand numerical value so by factorize function
# we convert the outcome/target value into numerical value
```

0-> Negative 1-> Positive

```
tweet = tweet_data.text.values
```

```
[32] # "word embedding"
# tokenize all the words in the text with the help of tokenizer
# in this we break down the word of a text into small part called tokens
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(tweet)
vocab_size = len(tokenizer.word_index) + 1
encoded_docs = tokenizer.texts_to_sequences(tweet)
padded_sequence = pad_sequences(encoded_docs, maxlen=200)
```

```
[33] print(tokenizer.word_index)
```

IOPub data rate exceeded.
The notebook server will temporarily stop sending output to the client in order to avoid crashing it.
To change this limit, set the config variable `--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)

5. OUR MODEL

Long Short-Term Memory Networks are also known as LSTM. Recurrent Neural Networks are a version of it. With sequential data, such as text and audio, recurrent neural networks are typically used. Typically, when calculating an embedding matrix, hidden states—which refer to the computations made for each word—are stored. RNNs are unable to store all these calculations in their memory if, for example, a word is referenced 100 words into a text. RNNs are therefore unable to learn these long-term dependencies. On the other hand, LSTMs perform well with this type of content. Time-series data are highly suited for LSTM networks.

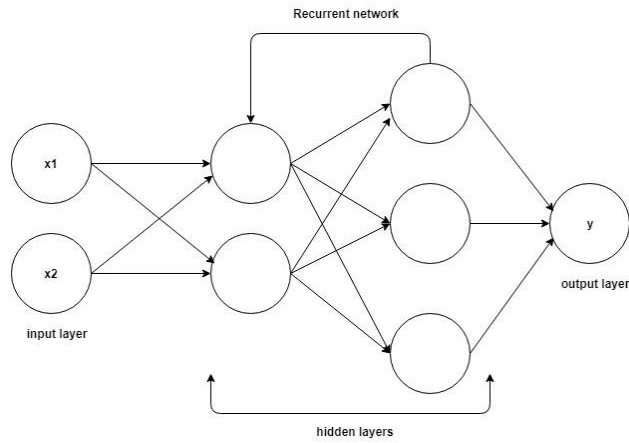


Fig.2 Recurrent Neural Network

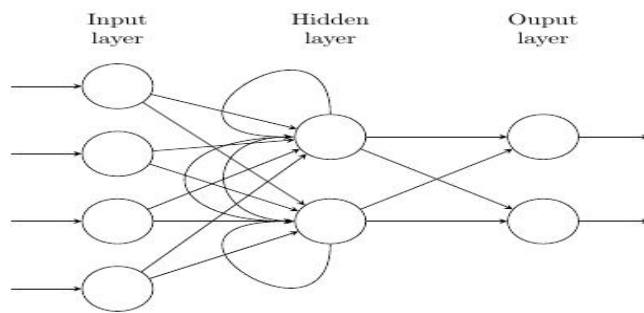


Fig.3 LSTM Neural Network

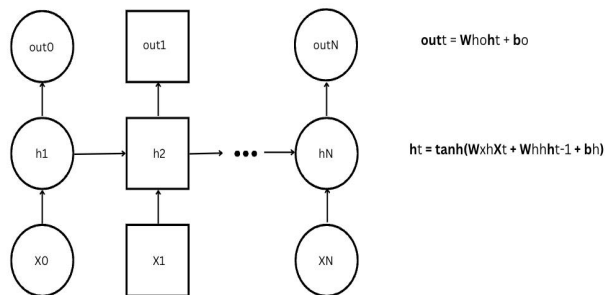


Fig.4 LSTM Recurrent Neural Network

Dropout is one of the regularization techniques. It is used to avoid overfitting. In the dropout mechanism, we drop some neurons randomly. The layer takes an argument, a number between 0 and 1 that represents the probability to drop the neurons. This creates a robust model avoiding overfitting.


```

2s ✓ ▶ embedding_vector_length = 32
model = Sequential()
model.add(Embedding(vocab_size, embedding_vector_length, input_length=200) )
model.add(SpatialDropout1D(0.25))
model.add(LSTM(50,
              kernel_regularizer=keras.regularizers.L1L2(l1=1e-5, l2=1e-4),
              activation="tanh"))

# model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(loss=keras.losses.BinaryCrossentropy(from_logits=False),optimizer='adam', metrics=['accuracy'])
print(model.summary())

```

Model: "sequential"

| Layer (type) | Output Shape | Param # |
|--|-----------------|---------|
| embedding (Embedding) | (None, 200, 32) | 9870400 |
| spatial_dropout1d (SpatialD ropout1D) | (None, 200, 32) | 0 |
| lstm (LSTM) | (None, 50) | 16600 |
| dense (Dense) | (None, 1) | 51 |

=====
 Total params: 9,887,051
 Trainable params: 9,887,051
 Non-trainable params: 0
 =====
 None

6. EXPERIMENT AND RESULT

Validation Accuracy- The term "test" or "testing" accuracy is frequently used to refer to the validation accuracy, which is the accuracy that was calculated on a dataset that was not used for training but was used during training to validate or "test" the generalizability of your model or to "early stop."

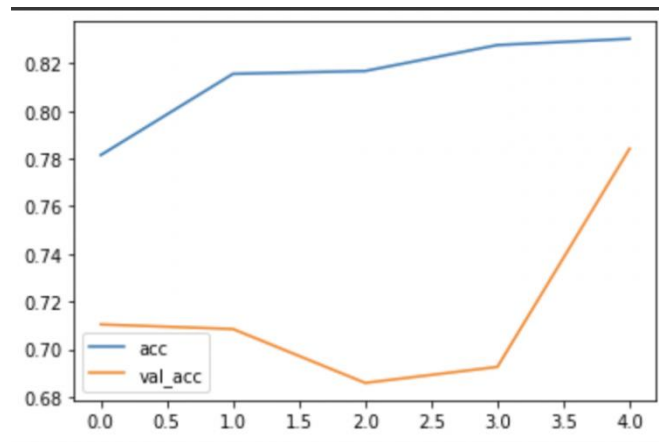


Fig.5 Validation Accuracy.

Validation loss - On the other hand, a deep learning model's performance on the validation set is evaluated using a statistic called validation loss.

The dataset's validation set is a section set aside to check the model's efficacy.

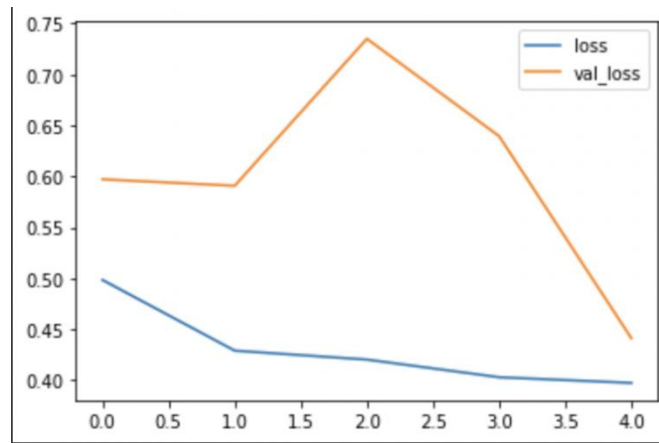


Fig.6 Validation Loss

6.1 Performance Metrics –

Performance metrics are a part of every machine learning pipeline. They quantify your progress and let you know if you're making any. Similar to performance measurements, every machine learning activity may be divided into two categories: regression and classification. There are other metrics for these issues, but we'll talk about the most common ones and the data they reveal regarding model performance. It's critical to understand how your model interprets your data.

6.1.1 Accuracy Score-

Classification When we say something is accurate, we usually mean it to be accurate. It measures the proportion of accurate predictions to all input samples. Only when there are an equal number of samples from each class does it function properly.

```

0s ✓ ▶ # Accuracy = (TP+TN)/(TP+FP+FN+TN)
    from sklearn.metrics import accuracy_score
    print("Accuracy Score is ",accuracy_score(Y_train, Y_test))
    accuracy_score(Y_train, Y_test, normalize=False)

```

```

☞ Accuracy Score is  0.6376266479492188
  334300

```

6.1.2 Precision and Recall-

By dividing the genuine positives by any positive predictions, precision is determined.

By dividing the real positives by anything else that should have been projected as positive, recall (also known as the True Positive Rate) is obtained.

```

0s ✓ ▶ # Precision and Recall
    from sklearn.metrics import precision_score, recall_score
    # precision_score(Y_train, Y_test)

    # recall_score(Y_train, Y_test)
    print("Precision score is",precision_score(Y_train, Y_test, average='macro'))

```

```

☞ Precision score is 0.49910509190332936

```

```

1s ✓ [21] print("Recall score is",recall_score(Y_train, Y_test, average='macro'))

```

```

Recall score is 0.49910449614417

```

6.1.3 Confusion matrix-

It is a way to gauge how well a machine learning classification algorithm performs when the output can include two or more classes. There are four possible anticipated and actual value combinations in the table. Recall, precision, specificity, accuracy, and—most importantly—AUC-ROC curves may all be measured with great success with this tool.

```
[17] print(X.shape,X_train.shape,X_test.shape)
      print(Y.shape,Y_train.shape ,Y_test.shape)

(1048576, 5) (524288, 5) (524288, 5)
(1048576,) (524288,) (524288,)
```

```
[28] from sklearn.model_selection import cross_val_predict
      from sklearn.metrics import confusion_matrix
      confusion_matrix(Y_train, Y_test)

array([[305006,  95054],
       [ 94934, 29294]])
```

6.1.4 Correlation-

We can determine the link between two quantities with the use of Pearson's correlation coefficient. It provides us with a measurement of how strongly two variables are associated. The Pearson's Correlation Coefficient's value can range from -1 to +1.

1 denotes a strong link between them, while 0 denotes no association. - A correlation of -1 indicates a negative relationship. Consider it to be inverse proportional.

```
f, ax = plt.subplots(figsize=(10, 8))
corr = tweet_data.corr()
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool),
            cmap=sns.diverging_palette(220, 10, as_cmap=True),
            square=True, ax=ax)
```

`/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: DeprecationWarning: `np.bool` is a deprecated alias for the builtin `bool``
`Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations`
`This is separate from the ipykernel package so we can avoid doing imports until`
`<matplotlib.axes._subplots.AxesSubplot at 0x7f506770c490>`

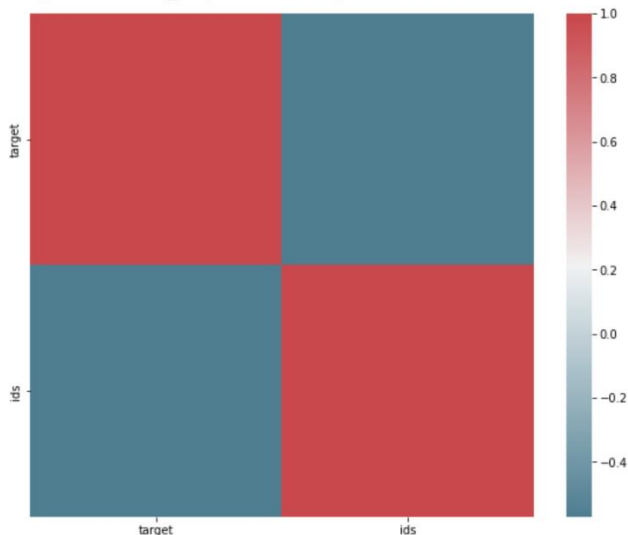
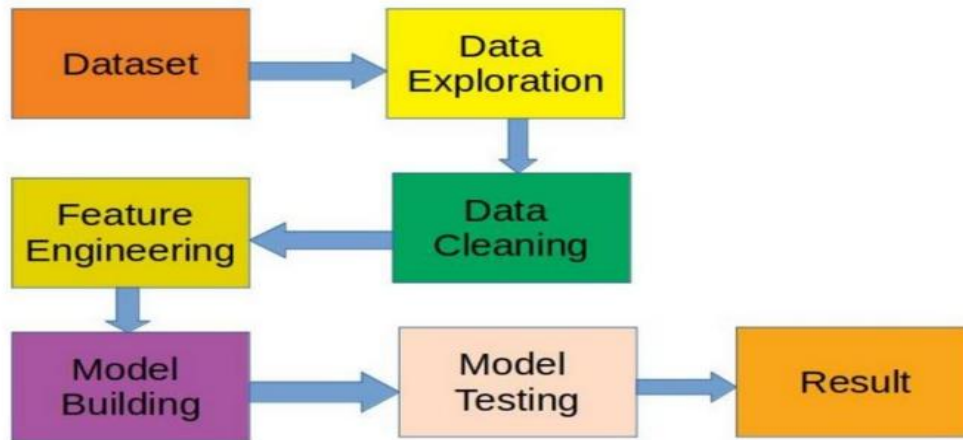


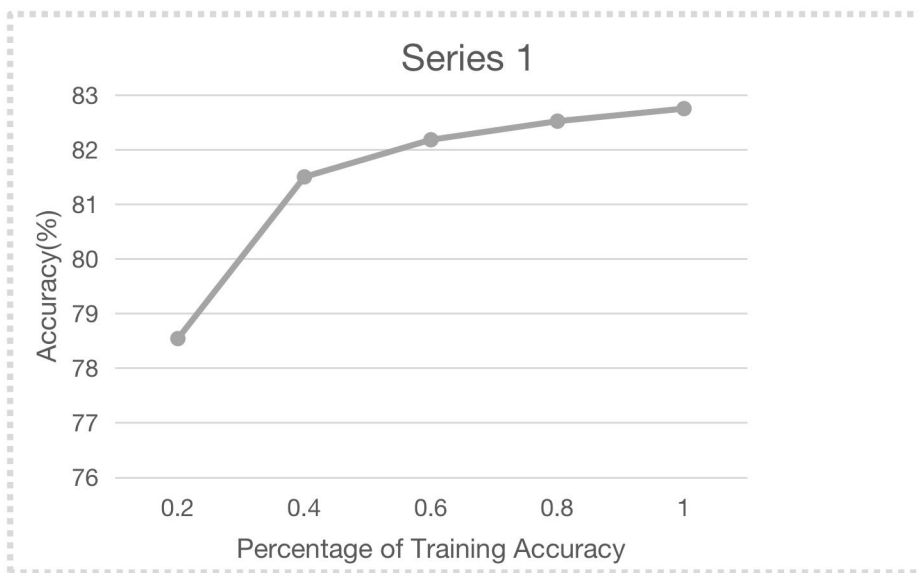
Fig. 7 Correlation

7. COMPARISION OF MODEL

Design Flow -



Learning Curve-



Model Comparison-

The our LSTM model outperforms more than **83% accuracy** for two classification tasks using a baseline of the most recent state-of-the-art than unigram model that was previously proposed contrasting the effectiveness of our three models. We present the 5-fold test accuracy's mean and standard deviation.

We find that the unigram and the Senti-features model that are outperformed by the tree kernels by 4.02% and 4.29% absolute, respectively. We note that this difference is much more pronounced comparing to the two way classification task.

| Model | Avg Accuracy(%) | Std Deviation (%) |
|----------------|-----------------|-------------------|
| Unigram | 56.58 | 1.52 |
| Senti-features | 56.31 | 0.69 |
| Kernel | 60.60 | 1.0 |

| Model | Avg Accuracy(%) | Std Deviation (%) |
|----------------------------|-----------------|-------------------|
| Unigram +Senti-features | 60.50 | 2.27 |
| Kernel+ Senti-features | 60.83 | 1.09 |

8. CONCLUSION AND FUTURE SCOPE

We displayed the findings from text sentiment analysis on Twitter Dataset. We present more than **83%** accuracy for two classification tasks using a baseline of the most recent state-of-the-art than unigram model that was previously proposed. On manually annotating the data, that represents a random sample of a stream of tweets, we provided an extensive collection of tests for both of these tasks.

```
In [23]: history = model.fit(padded_sequence, sentiment_label[0], validation_split=0.2, epochs=5, batch_size=32)

Epoch 1/5
12429/12429 [=====] - 2987s 240ms/step - loss: 0.4991 - accuracy: 0.7862 - val_loss: 0.6139 - val_accuracy: 0.6888
Epoch 2/5
12429/12429 [=====] - 2988s 240ms/step - loss: 0.4225 - accuracy: 0.8183 - val_loss: 0.5342 - val_accuracy: 0.7311
Epoch 3/5
12429/12429 [=====] - 29533s 2s/step - loss: 0.4097 - accuracy: 0.8244 - val_loss: 0.7384 - val_accuracy: 0.6546
Epoch 4/5
12429/12429 [=====] - 2221s 179ms/step - loss: 0.4030 - accuracy: 0.8272 - val_loss: 0.6522 - val_accuracy: 0.6660
Epoch 5/5
12429/12429 [=====] - 2139s 172ms/step - loss: 0.3973 - accuracy: 0.8314 - val_loss: 0.5259 - val_accuracy: 0.7203
```


We looked at two different types of models: Naive Bayes and Lexicon Based models, and we show that both of these models perform better than the unigram baseline. We do feature analysis for our feature-based approach, and the results show that the key features are those that integrate the prior polarity of words with their parts-of-speech tags. We speculate that Twitter data sentiment analysis is sentiment analysis for other genres is similar to this.

Future research will examine even more sophisticated linguistic analysis and e-commerce companies, such as topic modeling, semantic analysis, and parsing.

Final Output-

We give a statement (Bunty and Devansh are best friends and wrote this paper) to our model and then it recognize it as a **positive statement** which confirm that our model is correct.

0 -> Negative, 1 -> Positive

```
43s  tweet = input('Input tweet text : ')
predict_sentiment(tweet)

Input tweet text : Bunty and Devansh are best friends and wrote this paper
1/1 [=====] - 0s 17ms/step
Predicted label: 1
```

```
✓ [42] test_sentence1 = "I didn't enjoy my last flight"
      predict_sentiment(test_sentence1)

      test_sentence2 = "This is the best moment of my life "
      predict_sentiment(test_sentence2)

1/1 [=====] - 0s 375ms/step
Predicted label: 0
1/1 [=====] - 0s 30ms/step
Predicted label: 1
```

9. REFERENCES

- [1] walaa Medhata, Ahmed Hassan b, Hoda Korashy Sentiment analysis algorithms and applications. 2090-4479 2014
- [2] Kang Hanhoon, Yoo Seong Joon, Han Dongil. Senti-lexicon and improved Naive Bayes algorithms for sentiment analysis of restaurant reviews. Expert Syst Appl 2012;39:6000–10.
- [3] Turney, 2002; Pang and Lee, 2004 Sentiment analysis using support vector machines with diverse information sources.
- [4] Hu and Liu, 2004; Kim and Hovy, 2004 Opinion Extraction and Summarization on the Web.
- [5] Wilson et al., 2005; Agarwal et al.,2009 Contextual Phrase-Level Polarity Analysis using Lexical Affect Scoring and Syntactic N-grams.
- [6] Pak and Paroubek 2010. Twitter as a corpus for sentiment analysis and opinion mining.
- [7] Doaa Mohey El-Din Mohamed Hussein. A survey on sentiment analysis challenges.
- [8] Medhat- 2014 Sentiment analysis algorithms and applications: A survey.
- [9] Jain & Dandannavar, 2016 Application of machine learning techniques to sentiment analysis.