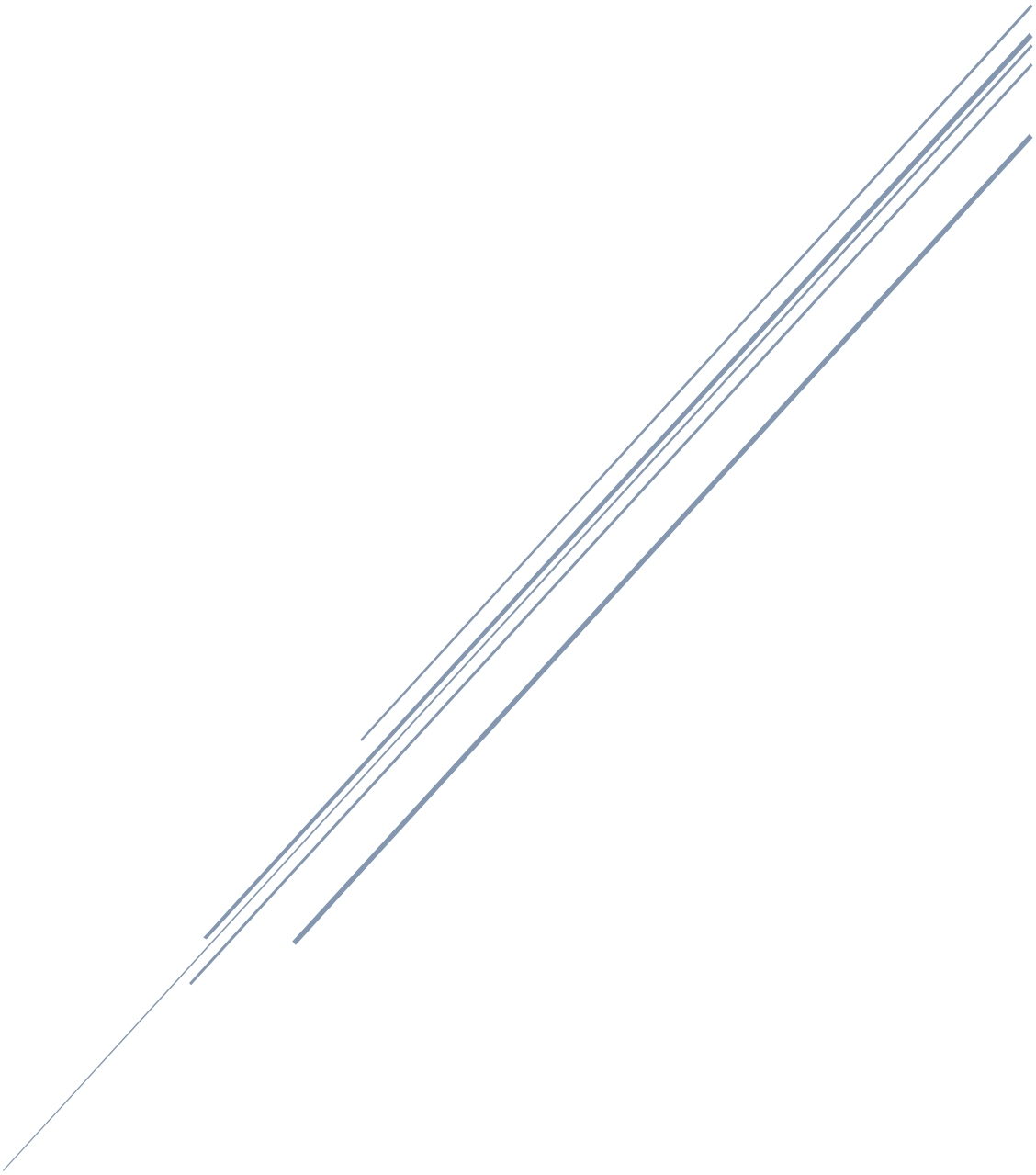
# MODEL DEVELOPMENT FOR SENTIMENT ANALYSIS

## Citizens' Voice Platform



Author: Michael Dubem Igbomezie

## Introduction

This section of the Election Campaign platform is centred on implementing various data transformation methods in combination with various types of models in order to answer the research questions posed in the first phase of this project and focused on in the second phase of the project.

The different data transformation methods used in this project were the Bag of Words (BOW), a customized character encoding method developed specifically for this project, combination of the Gensim Word2Vec word encoder and PCA data reduction method, a combination of the Gensim Word2Vec word encoder and the average function, combination of the Gensim Word2Vec word encoder and Tensorflow's tensor transformation, and the Keras word encoder. The different models that were tested in this project were the Multinomial Naïve Bayes Classifier (MNB), Multilayer Perceptron (MLP), Simple Recurrent Neural Network (SimpleRNN) and Long and Short Term Memory Recurrent Neural Network (LSTM). As it can already be observed, there's quite a long list of data transformation methods in combination with another long list of different types of classifier algorithms. The purpose of this is not just random combinations but specifically to check which combinations best retain two type of pattern present in text data.

There are two major kinds of context (pattern/value) embedded in sentences, and a good language analyser should be able to account for both contexts.

1. Context in the sequence of words (Relational Context): this refers to grammar (language structure). If two sentences are made of the same words, but in a different sequence entirely resulting to a different meaning, the difference should be notable.

2. Context in the meaning of words (Internal Context): this refer to meaning of words, therefore if the same sentence is constructed using synonyms, the meaning can be understood as the same.

The only dataset used in this project was a dataset comprised of tweets from the Nigerian Twitter online space, which was extracted using the Twitter API and stored in an online database.

Nigerian Twitter Dataset

```
In [3]: pd.set_option("display.max_colwidth", None)
        pd.set_option("display.max_rows", None)

        tweets = pd.read_csv("citizensvoice_dataset.csv", index_col=0, encoding='latin-1')
```

```
In [4]: tweets.head()
```

Out[4]:

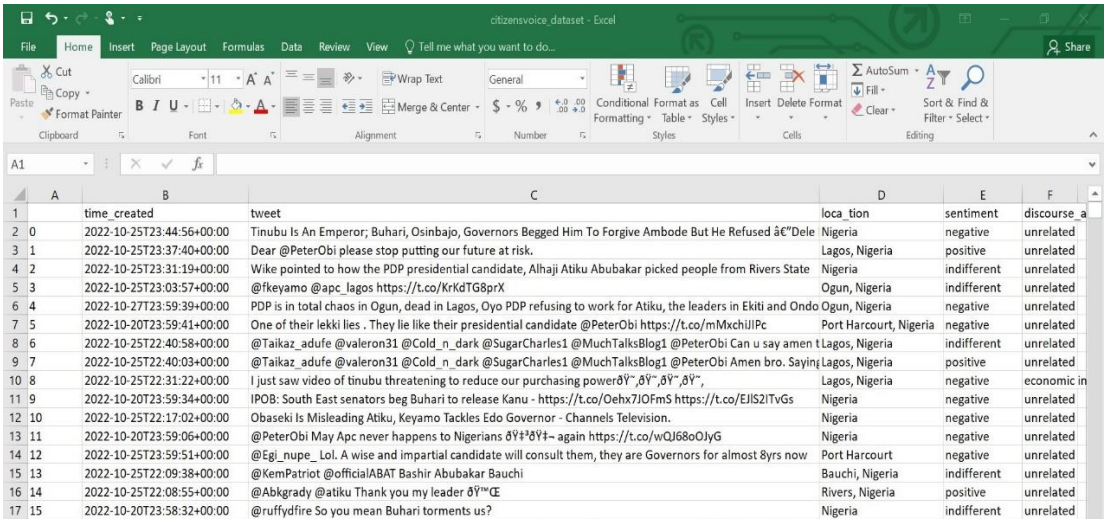
| | time_created | tweet | loca_tion | sentiment | discourse_area |
|---|---|---|---|---|---|
| 0 | 2022-10-25T23:44:56+00:00 | Tinubu Is An Emperor; Buhari, Osinbajo, Governors Begged Him To Forgive Ambode But He Refused â□□Dele Momodu \| Sahara Reporters https://t.co/mO1zWgXQxn | Nigeria | negative | unrelated |
| 1 | 2022-10-25T23:37:40+00:00 | Dear @PeterObi please stop putting our future at risk. \r\nYou are the only reason I still believe in Nigeria. \r\nMy vote is for you https://t.co/nKhLhzrV8H | Lagos, Nigeria | positive | unrelated |
| 2 | 2022-10-25T23:31:19+00:00 | Wike pointed to how the PDP presidential candidate, Alhaji Atiku Abubakar picked people from Rivers State as members of the presidential campaign council without any input from him.\r\n\r\nhttps://t.co/H2cicBJlu1 | Nigeria | indifferent | unrelated |
| 3 | 2022-10-25T23:03:57+00:00 | @fkeyamo @apc_lagos https://t.co/KrKdTG8prX | Ogun, Nigeria | indifferent | unrelated |
| 4 | 2022-10-27T23:59:39+00:00 | PDP is in total chaos in Ogun, dead in Lagos, Oyo PDP refusing to work for Atiku, the leaders in Ekiti and Ondo are refusing to mount a challenge, only in Osun does the party have a deem hope. https://t.co/w3r3dmSa0i | Ogun, Nigeria | negative | unrelated |

1. Methodology

This section of the project covers the data labelling, cleaning and various transformation methods that were carried out on the dataset before it was used to train the various selected classification models.

A. Data Labelling and Cleaning

The data was loaded in an excel file and was then labelled manually using a basic knowledge of human sentiment, and domain specific knowledge of the nature of the lingua and slangs used in Nigerian conversations. The three sentiment categories that were considered in this labelling were, Positive, Negative and Indifferent. The total number of labelled rows were over 2000 rows.



| | A | time_created | tweet | loca_tion | sentiment | discourse_a |
|---|---|---|---|---|---|---|
| 2 | 0 | 2022-10-25T23:44:56+00:00 | Tinubu Is An Emperor; Buhari, Osinbajo, Governors Begged Him To Forgive Ambode But He Refused â€"Dele | Nigeria | negative | unrelated |
| 3 | 1 | 2022-10-25T23:37:40+00:00 | Dear @PeterObi please stop putting our future at risk. | Lagos, Nigeria | positive | unrelated |
| 4 | 2 | 2022-10-25T23:31:19+00:00 | Wike pointed to how the PDP presidential candidate, Alhaji Atiku Abubakar picked people from Rivers State | Nigeria | indifferent | unrelated |
| 5 | 3 | 2022-10-25T23:03:57+00:00 | @fkeyamo @apc_lagos https://t.co/KrKdTG8prX | Ogun, Nigeria | indifferent | unrelated |
| 6 | 4 | 2022-10-27T23:59:39+00:00 | PDP is in total chaos in Ogun, dead in Lagos, Oyo PDP refusing to work for Atiku, the leaders in Ekiti and Ondo | Ogun, Nigeria | negative | unrelated |
| 7 | 5 | 2022-10-20T23:59:41+00:00 | One of their lekki lies . They lie like their presidential candidate @PeterObi https://t.co/mMxchiJIPc | Port Harcourt, Nigeria | negative | unrelated |
| 8 | 6 | 2022-10-25T22:40:58+00:00 | @Taikaz_adufe @valeron31 @Cold_n_dark @SugarCharles1 @MuchTalksBlog1 @PeterObi Can u say amen t | Lagos, Nigeria | indifferent | unrelated |
| 9 | 7 | 2022-10-25T22:40:03+00:00 | @Taikaz_adufe @valeron31 @Cold_n_dark @SugarCharles1 @MuchTalksBlog1 @PeterObi Amen bro. Sayin | Lagos, Nigeria | positive | unrelated |
| 10 | 8 | 2022-10-25T22:31:22+00:00 | I just saw video of tinubu threatening to reduce our purchasing powerðŸ˜,ðŸ˜,ðŸ˜, | Lagos, Nigeria | negative | economic in |
| 11 | 9 | 2022-10-20T23:59:34+00:00 | IPOB: South East senators beg Buhari to release Kanu - https://t.co/Oehx7JOFmS https://t.co/EJlS2ITvGs | Nigeria | negative | unrelated |
| 12 | 10 | 2022-10-25T22:17:02+00:00 | Obaseki Is Misleading Atiku, Keyamo Tackles Edo Governor - Channels Television. | Nigeria | negative | unrelated |
| 13 | 11 | 2022-10-20T23:59:06+00:00 | @PeterObi May Apc never happens to Nigerians ðŸ‡³ðŸ‡¬ again https://t.co/wQJ68oOJyG | Nigeria | negative | unrelated |
| 14 | 12 | 2022-10-25T23:59:51+00:00 | @Egi_nupe_ Lol. A wise and impartial candidate will consult them, they are Governors for almost 8yrs now | Port Harcourt | negative | unrelated |
| 15 | 13 | 2022-10-25T22:09:38+00:00 | @KemPatriot @officialABAT Bashir Abubakar Bauchi | Bauchi, Nigeria | indifferent | unrelated |
| 16 | 14 | 2022-10-25T22:08:55+00:00 | @Abkgrady @atiku Thank you my leader ðŸ™Œ | Rivers, Nigeria | positive | unrelated |
| 17 | 15 | 2022-10-20T23:58:32+00:00 | @ruffydfire So you mean Buhari torments us? | Nigeria | indifferent | unrelated |

After the labelling was done the dataset was then imported into the Jupyter environment and was cleaned. The cleaning process involved the use of regular expression to remove various unwanted elements in the texts such as, @ symbols, numbers, emails, web links, and emoji amongst others.

In [8]:
```
def clean_tweet(tweet):
    """
    Function to clean tweet by:
    - Changing '&' sign to and
    - Removing newlines, carriage returns, links, emojis, handles, hashtags and punctuations

    Parameters:
        tweet (string): The tweet
    """

    tweet = re.sub("@[^\s]+", "",tweet) # removes handles
    tweet = re.sub("\n", " ", tweet) # remove newlines
    tweet = re.sub("\r", "", tweet) # remove carriage returns
    tweet = re.sub(r"http\S+", "", tweet) # removes links
    tweet = re.sub(emojis, "", tweet) # remove emojis
    tweet = re.sub(r"#(\w+)", "", tweet) # remove hashtags
    tweet = re.sub("&", "and", tweet) # changes & sign to and
    tweet = re.sub(r"[^\w\s\@]","",tweet) # removes punctuation
    tweet = tweet.strip()

    return tweet
```

In [9]:
```
tweets["clean_tweet"] = tweets.tweet.apply(clean_tweet)
```

In [10]:
```
tweets[["tweet", "clean_tweet"]].sample(5)
```

Out[10]:

| | tweet | clean_tweet |
|---|---|---|
| **16854** | apc inaugurates presidential campaign council in canada https://t.co/wmrqdmnw7h | apc inaugurates presidential campaign council in canada |
| **36557** | @aidelomo_happy @mysteriousdoct3 @shehusani that is ur id card for sure. you can keep piling till next year december no wahala, but one thing you shud know is that your principal can't win the election, simple as apc | that is ur id card for sure you can keep piling till next year december no wahala but one thing you shud know is that your principal cant win the election simple as apc |
| **693** | @mahdeeus @rbleipzig_en see what vini cause us | see what vini cause us |

B. Data Transformation

There are six (6) different types of transformation methods used in this projects. Some of these transformation method retain the sequential position of each word in a text while some don't, some retain the internal meaning of each word in the text while others don't. Each of these methods are explained as follows.

- Bag of Words Method: This transformation method removes the stop words in the dataset, conducts a frequency count of all the remaining words in the dataset, and selects the first 1500 words as the feature set. To encode a sentence or text with this feature set, the text is checked against the entire feature set to see which words in the feature set it has and which words are absent. This implies that a single data point/instance will be a binary vector that contains ones for the words in the feature set it has and zeros for those it doesn't have.

- Custom Character Encoder: This was a personal transformation method developed to simply check if simply remembering the sequence of characters in a sentence was enough to pick up on the sentiment of the said sentence. The linspace function was used to assign a numeric value (between zero and one) to each letter in the alphabet, and for a given sentence, all the character were simply replaced with their assigned numeric value. The longest sentence was used to determine the length of the input vector and all other sentences with shorter length were padded with zeros to make up for the missing length. This is different from one-hot encoder. This method has no use for the removal of stop words.

- Combination of Gensim Word2Vec word encoder with PCA: The "word2vec-google-news-300" dictionary from the Gensim Corpus was used to encode the words of each sentence. The Gensim dictionary basically contains the vector representation of a vast number of word, which retains their respective inherent meaning, and reflects similarity in words. This is because each number of a word's vector representation is its score in a certain feature. These features are generated using Neural Network model. This method also has no use for stop words removal, however due to length of each vector being up to 300 elements, the entire array of vector representation of words in a given sentence is then reduced using the PCA algorithm, for faster computation time.

- Combination of Gensim Word2Vec encoder with Average Function: This transformation is identical to the immediate previous method, with just one difference. In this method, instead of using PCA to reduce the vector to a smaller size, the average value of each vector is simply taken to represent the entire vector.

- Combination of Gensim Word2Vec encoder with Tensorflow's Tensor converter: This transformation method is also very similar to the two immediate previous methods. The difference in this case is that no data reduction was carried out. After all the words in the sentences are encoded using the Gensim dictionary, the complete dataset is converted to a tensor, which is later fed directly into the Neural Network, so as to retain all the information originally contained in the dataset.

- Keras Word Encoder: In this transformation method, the Keras tokenizer is used to create a vocabulary with respect to all the words in the dataset. This vocabulary is then used to assign a numerical value to each existing word in the dataset. This method as well has no use for stop words removal.

C. Model Training

- Multinomial Naïve Bayes: The Multinomial NB was trained on the encoded dataset gotten from the Bag of Words methods. The Multinomial NB was used because the target variable had more than two categories to predict

- Dense Multilayer Perceptron: The MLP was trained on the encoded dataset gotten from the custom character encoder, the combination of the Gensim Word2Vec encoder with PCA, and the combination of the Gensim Word2Vec encoder with the average function. The purpose of including this model was to check if simply remembering which words were included in the sentence without any regards for their sequential position would be enough to make a good prediction.

- Simple Recurrent Neural Networks: The Simple RNN was trained on the dataset gotten from all the transformation methods used except the Bag of Words method. This was to check if retention of the relation context (which refers to the inherent meaning contained in a sentence due to it sequential structure) played any significant role in accurately predicting the sentiment of the data.

- Long and Short Term Memory Recurrent Network: The LSTM wasn't initially included in the models trained and was only added towards the end of the experiment to see if it will cause any significant improvement in the result given by the best model.

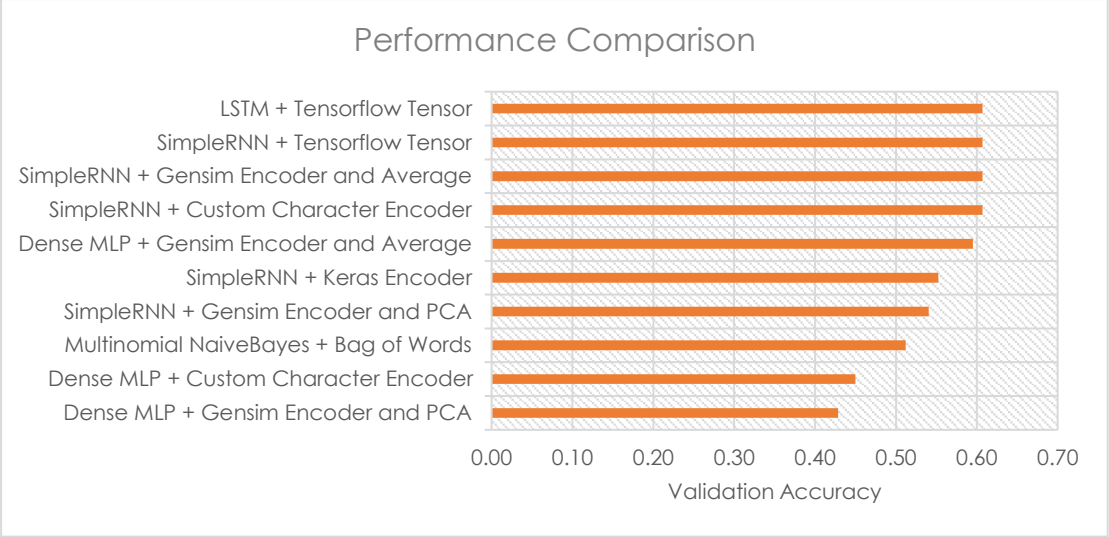
## 2. Result and Discussion

The results were organized into tabular format, with the algorithm types with their respective data transformation being the index and their validation accuracy being the columns. The table is ordered in ascending order.

### A. Results:

Table 1. Model Performance

| Data Transformation Method | Validation Accuracy |
|---|---|
| **Dense MLP + Gensim Encoder and PCA** | 0.43 |
| **Dense MLP + Custom Character Encoder** | 0.45 |
| **Multinomial Naive Bayes + Bag of Words** | 0.51 |
| **SimpleRNN + Gensim Encoder and PCA** | 0.54 |
| **SimpleRNN + Keras Encoder** | 0.55 |
| **Dense MLP + Gensim Encoder and Average** | 0.60 |
| **SimpleRNN + Custom Character Encoder** | 0.61 |
| **SimpleRNN + Gensim Encoder and Average** | 0.61 |
| **SimpleRNN + Tensorflow Tensor** | 0.61 |
| **SimpleRNN + Tensorflow Tensor** | 0.61 |
| **LSTM + Tensorflow Tensor** | 0.61 |




The first thing to notice about the results is that all the models performed fairly well with some performing much better than others, considering that a random guess with three categories in a target variable would usually amount to 30% performance.

However the interesting part is in the embedded meaning in the results, which would be broken down in the next section.

B. Observation and Hindrances:

In this section of the report, the models' comparative performance and the data transformation type comparative performance will be first scrutinized before the general performance and ways to improve it is expanded on.

- Models', Transformation Types' Comparative Performance:

Considering the model performance, it can be observed that the SimpleRNN outperformed all the MLP models (but one), even in cases where the SimpleRNN was trained with just half the number of nodes used for the MLP. All the highest accuracies were produced by SimpleRNN which implies that sequential training played a very significant role in the performance

In terms of encoder types, it can be observed that the Gensim Encoder combined with Tensorflow's Tensor, Gensim Encoder combined with Average, and the Custom Character Encoder were the top performing encoders. However, it would only be logical to prefer the Gensim Encoder to the Custom Character Encoder because checking the model where the Custom Character Encoder was used on MLP which pays no mind to sequence in data, there was a significant drop in performance, which implies that just knowing the character that appear in a sentence is not nearly enough to capture all the embedded meaning in that sentence. Although knowing the sequence of characters could improve this performance significantly.

At this point it starts to look pretty obvious that the highest performing models are those where Gensim Encoder was combined with SimpleRNN. One possible explanation for this is that this is the method which retains both the internal meaning of each word in the sentence, and the relational meaning of the word sequence in each sentence, which are two very crucial types of meaning which form pattern that can be captured through proper analysis.

Looking at this, one could ask "If Gensim Encoder and SimpleRNN was the best combination, why then was Gensim Encoder with PCA and SimpleRNN not amongst the best?". One explanation for this is that the internal meaning of words that would have been retained by Gensim Encoder was removed due to the reduction method enforced by PCA, hence only one type of meaning (relational meaning) was retained, hence a lower performance. Taking a look at the combination of Gensim Encoder with PCA and MLP where no type of meaning (internal nor relational) was retained, it can be seen that this was the model that gave the lowest performance.

The LSTM model was implemented on best Gensim Encoder method and the performance although amongst the best in the project, didn't add any further improvement to the already achieved best score.

- Overall Performance:

Taking a look at the overall fair performance of the model, there are a series of possible causes which could have led to this.

- Error Induced by incorrectly labelled data: because many sample in the extracted data had mixed sentient which led to difficulty in correct labelling, the existence of any proper distinct pattern for each sentiment will be hard, which ultimately leads to either model not converging properly or models that perform bad even if they converge.

- Not Enough Data: because of the variety of ways the exact same human emotion can be expressed using the English language, 2000 rows generally might not capture this variety efficiently.

- Small Neural Network Models: due to the fact that resources are limited (none of the models were trained using GPUs or even parallel computing), and some of the transformation types producing tensors with up to 300 elements for a single input cell, the models would be quite limited in the intensity of the pattern they would be able to capture in the data.

Any of these three factors is enough to hinder a model from performing well, but if all three are present, then you get models that performing fair or even worse than random guesses, even though the model trained in this project performed much better than random guesses.

## C. Possible Improvements:

There are three possible ways amongst other that could lead to model performance improvement, and they are as follows.

- BERT Sentence Encoder + LSTM: BERT sentence encoder could be an easier way to encode entire sentence while retaining both internal word meanings and relational word meanings in sentence, which could allow for faster encoding of larger datasets, for more training, testing and experimenting.

- Parallel + Cluster Computing: Using more processors at a time for the training process could allow for training of much larger models that could begin to capture much more pattern in the dataset.

- More Useful Data: It is not a new fact that in the data world, the more useful data, the better. This could very well be crucial in this case as there is a large variety of ways the same sentiment can be linguistically expressed, hence the more useful data gotten, the more pattern can be recognized by good algorithms.

- Properly Splitting Mixed Sentiments, and Properly Capturing the Different types of Sentiments: types of sentiment can vary but often oversimplified as either positive or negative. An angry sentiment is worded differently from sad sentiment, which is worded differently from pity sentiment, but they all have the same aura, which is simplistically classified under negative sentiment. Properly separating these different major existing sentiments could lead to better pattern capture with good algorithms. Also sentences with mixed sentiments could be split where one sentiment ends and the other begins.

## 3. Conclusion

This project showed the importance of Recurrent Neural Networks in text data type. This is not out of the ordinary as text data type in this particular context is in sequential form, and Recurrent Neural Networks were developed specifically for this type of data.

The project also showed the importance of any good sentence classification model and encoder combination having the ability to capture the internal meaning and relational meaning in any sentence, as these two types of embedded meaning form the entire message (information) being passed in the said sentence.