Sentiment Analysis of Product Reviews

Shree Rama Kamal Kumar Vegu, vegu, B00872272, sh907407@dal.ca
Harsh Gawai, gawai, B00876768, hr912357@dal.ca
Mani Teja Varma Kucherlapati, mtvk, B00859456, mn947410@dal.ca
Gautam Swaminath Ganesh , gganesh, B00862496, gt728571@dal.ca

Natural Language Processing Report
Dalhousie University
Faculty of Computer Science

1. ABSTRACT

With the rise in popularity of online shopping and the number of different products available in the market. The time and effort takes for determining products are good or bad is significant. This is done by the user on visiting sites such as amazon , blogs , YouTube videos etc. The feedback obtained from these purchases is vital for the website to improve the experience for future customers. Finding the best product from multiple sources based on the user or customer reviews is an important problem in this regard. This is not only important for customers but also for owners. Customers find this information interesting because this allows them to know the best company and get an idea about the product that they plan to purchase and similarly it is important for the product or company owners too because it allows them to know the view of the common man about their products and could aid them elevate their enterprise. This project tries to provide a solution to it by analyzing user reviews[1].

For that, we will build a model that will let the user know how much each product is worthy. The products are suggested by the user which he or she wants to know and compare.

The application gets the data from Twitter, other social networking sites or E-Commerce platforms. It goes through all the reviews and calculates the aggregate sentiment for the specific product. Users provide multiple products in the input that will let the application calculate the sentiment and compare between products. We will calculate the percentages based on the distance between optimistic/positive and damaging/negative words.

The research includes determining the mood/trend of the customers on the products. For example, if the user was frustrated with the product, the mood is "ANGRY" and if the user is super excited after using the product, the mood is “HAPPY”[2], so by applying these natural language processing techniques we can solve the analyzing of user review problems quickly and in an efficient manner.

2. INTRODUCTION

Online purchasing has become a core part of our society . Households prefer to shop online than visit stores in person. Thus, it became imperative for better online shopping sites. Hence large amounts of work were done to improve the UI and into research for human computer interaction. Through this customer found a very pleasant experience working with flashy and comfortable websites. However there also was also a large influx in products in such websites. Thus, this caused users to spend a considerable amount of time in the research for the good products among the others. Once the tedious search and the product has been purchased the customer is required to give feedback. This is very useful to the company is it can use this feedback to filter the bad products from the good ones[3].

These reviews are vital for both the user and the companies to improve their product and know what users want regarding such products. From the reviews one can determine if the user is happy or upset with the purchase. This is done with the help of sentiment analysis. Sentiment analysis is the process of finding emotional sentiment or opinions of the user. This method is also known as opinion mining. The aim of the project is to determine the top 3 products according to genre (Washing machine , Television, and earphones) by doing sentiment analysis of the user reviews. The motivation behind this is the importance of user reviews for the website on a business level. The solution to this problem statement can provide the restaurant with crucial information from the user point of view. The objective of this project is to experiment with various natural language processing algorithms that can be used to perform sentimental analysis and to evaluate these models. With the result being to procure the 3 best products of their unique genre.

First trial is done through matching the words in a review with the already collected list of positive and negative words and the sentiment of the review is predicted based on this. This is a simple approach and the model designed is very basic in this case. Data is split into training and test data with 70--30 percentages. Random Forest algorithm is applied on the training data. This model is evaluated with various parameters such as accuracy, confusion matrix, f score, precision, recall. Then further experimentation is done. Reviews are split into word trigrams and a Random forest algorithm is applied. Similarly, Naive Bayes algorithm is applied on both character trigrams and word trigrams and evaluates all these models based on various factors. In each of the models , we determine the top 3 products[4].

In addition to this , the project aims on improving the quality of life for the user by adding speech analysis to the model. This would enable the user to use voice commands to further improve his search. This is done by utilizing the python package speech recognition. This will take the user's voice as an input to filter the key words that would assist the system in finding the desired product.

Once the classification of the top 3 products is obtained along with their mapped reviews ,the system then applies entity recognition to determine the common reasons as to why the different customers like and dislike different products. This allows for a comprehensive understanding of the users likes and dislikes along with a basic understanding of what the users want in their products.

3. RELATED WORK

The first research work is from paper- “Thumbs up? Sentiment Classification using Machine Learning Techniques” by Bo Pang and Lillian Lee, Department of Computer Science, Cornell University & Sivakumar Vaithyanathan, IBM Almaden Research Center. This paper discusses the problem of classification based on the overall sentiment of various documents. The classes being considered are positive and negative. The data on which classification is done is the movie review data. During experimentation, it is also found that the classifiers Naive Bayes, maximum entropy classification and support vector machines do not perform well on the traditional topic-based classification. This paper also discusses the strength of sentiment of different words and does experimentation with the unigrams, bigrams, and parts of speech as a feature set. It considers these three and various combinations of these as a feature set[5].

4. PROBLEM DEFINITION AND METHODOLOGY

Our research problem here is to perform sentiment analysis of scraped data reviews and the amazon phone dataset to find the top 3 products in each genre and the reasoning behind the purchase.

4.1 Implementation Part 1

For the beginning of our work , data was gathered using web scraping through the website, flipkart.com. The data extracted were products and reviews of various genre’s namely vacuum cleaner, Televisions, and earphones.

4.1.1 Web Scraping

This process of web scraping is done using the function Beautiful soup , we input the website and the required tags to fetch the data. So, in this case we input the website flipkart and the tags being the different genre intended (vacuum cleaner , Televisions, and earphones) and the corresponding reviews of the products. Once the data has been obtained the corresponding products are mapped along with their reviews and is stored as a CSV file.

4.1.2 Amazon Phone Reviews

In addition to this web scraped data, we also use an Amazon phone review dataset that contains 400 thousand phone reviews and 2000 unique brands of phones. A basic preprocessing is done on this dataset i.e., removal of null values , basic encoding of the data and splitting data into test and train[6].

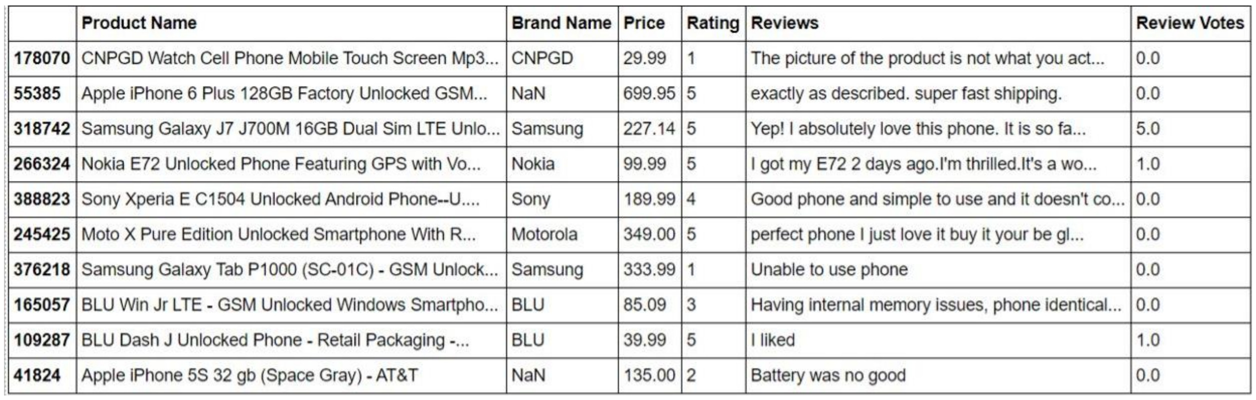
4.1.3 Vader Sentiment Analysis

The aim is to perform a basic sentiment analysis using Vader sentiment analysis. Vader Sentiment is a low resource consuming model, it uses mathematical rules instead of hardcoding the information into the required format.

The data can also be encoded and decoded easily. Vader sentiment is especially used to quantify the emotions present in the data from multiple types of data namely video , audio, and text.

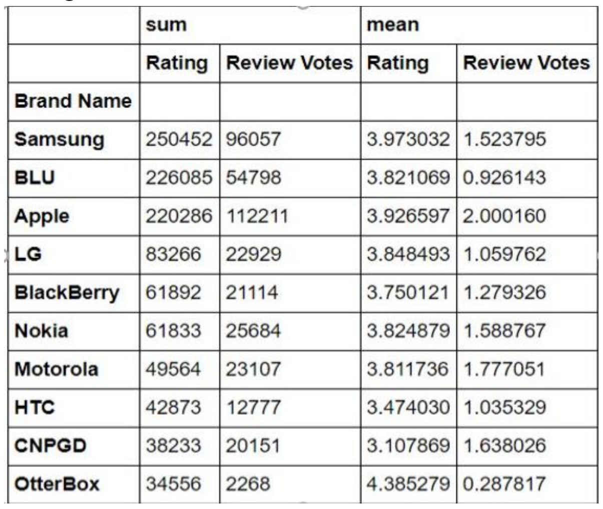
Vader sentiments are also good if not better at matching the sentiments of sentences when compared with humans who were assigned to the same task.
Here we will create a dictionary that would contain the mapping of the words and the lexical features that indicate a particular sentiment.

Once the basic preprocessing is complete the data is shown in the figure below:

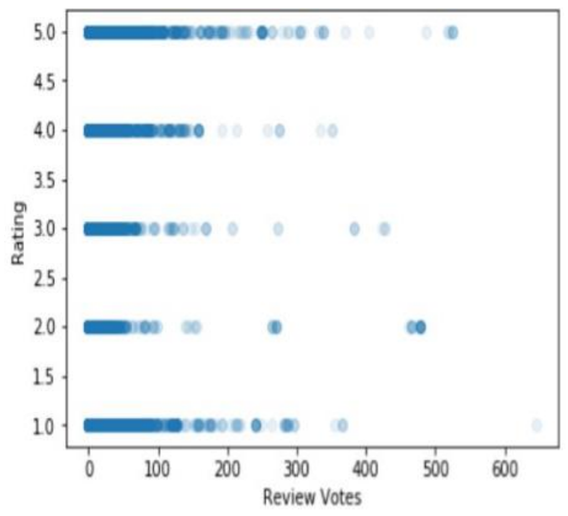


	Product Name	Brand Name	Price	Rating	Reviews	Review Votes
178070	CNPGD Watch Cell Phone Mobile Touch Screen Mp3...	CNPGD	29.99	1	The picture of the product is not what you act...	0.0
55385	Apple iPhone 6 Plus 128GB Factory Unlocked GSM...	NaN	699.95	5	exactly as described. super fast shipping.	0.0
318742	Samsung Galaxy J7 J700M 16GB Dual Sim LTE Unlo...	Samsung	227.14	5	Yep! I absolutely love this phone. It is so fa...	5.0
266324	Nokia E72 Unlocked Phone Featuring GPS with Vo...	Nokia	99.99	5	I got my E72 2 days ago.I'm thrilled.It's a wo...	1.0
388823	Sony Xperia E C1504 Unlocked Android Phone--U....	Sony	189.99	4	Good phone and simple to use and it doesn't co...	0.0
245425	Moto X Pure Edition Unlocked Smartphone With R...	Motorola	349.00	5	perfect phone I just love it buy it your be gl...	0.0
376218	Samsung Galaxy Tab P1000 (SC-01C) - GSM Unlock...	Samsung	333.99	1	Unable to use phone	0.0
165057	BLU Win Jr LTE - GSM Unlocked Windows Smartpho...	BLU	85.09	3	Having internal memory issues, phone identical...	0.0
109287	BLU Dash J Unlocked Phone - Retail Packaging -...	BLU	39.99	5	I liked	1.0
41824	Apple iPhone 5S 32 gb (Space Gray) - AT&T	NaN	135.00	2	Battery was no good	0.0

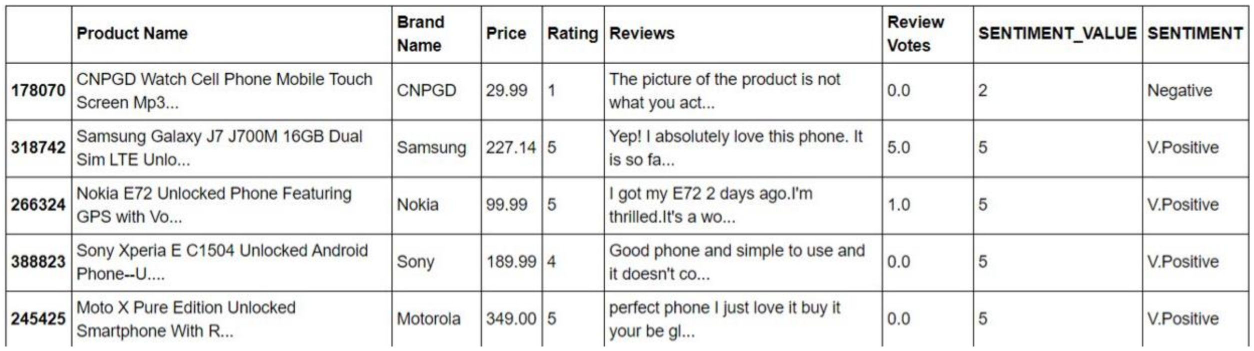
To summarize and for better understanding , we average the data per brand with respect to the ratings and the review votes.



	sum		mean	
	Rating	Review Votes	Rating	Review Votes
Brand Name				
Samsung	250452	96057	3.973032	1.523795
BLU	226085	54798	3.821069	0.926143
Apple	220286	112211	3.926597	2.000160
LG	83266	22929	3.848493	1.059762
BlackBerry	61892	21114	3.750121	1.279326
Nokia	61833	25684	3.824879	1.588767
Motorola	49564	23107	3.811736	1.777051
HTC	42873	12777	3.474030	1.035329
CNPGD	38233	20151	3.107869	1.638026
OtterBox	34556	2268	4.385279	0.287817



On performing Vader sentiment analysis on the data, we obtained the following output.



	Product Name	Brand Name	Price	Rating	Reviews	Review Votes	SENTIMENT_VALUE	SENTIMENT
178070	CNPGD Watch Cell Phone Mobile Touch Screen Mp3...	CNPGD	29.99	1	The picture of the product is not what you act...	0.0	2	Negative
318742	Samsung Galaxy J7 J700M 16GB Dual Sim LTE Unlo...	Samsung	227.14	5	Yep! I absolutely love this phone. It is so fa...	5.0	5	V.Positive
266324	Nokia E72 Unlocked Phone Featuring GPS with Vo...	Nokia	99.99	5	I got my E72 2 days ago.I'm thrilled.It's a wo...	1.0	5	V.Positive
388823	Sony Xperia E C1504 Unlocked Android Phone--U....	Sony	189.99	4	Good phone and simple to use and it doesn't co...	0.0	5	V.Positive
245425	Moto X Pure Edition Unlocked Smartphone With R...	Motorola	349.00	5	perfect phone I just love it buy it your be gl...	0.0	5	V.Positive

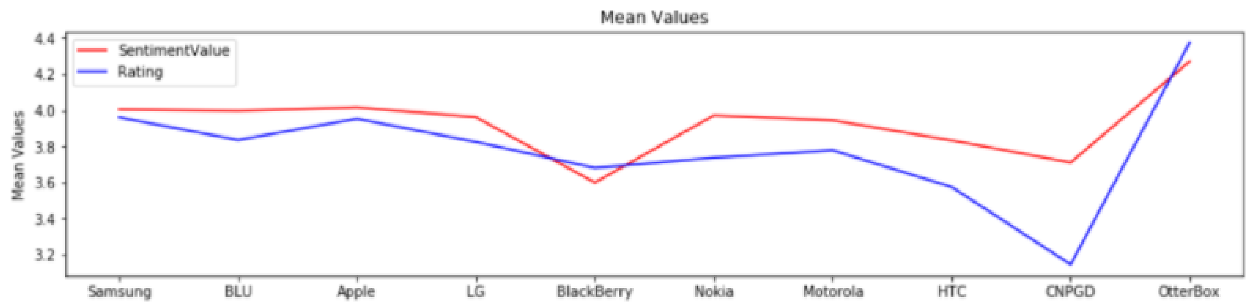
This is a general example of one of the products reviewed. On comparing the review with the dictionary that was initially created. The sentence is then analyzed and determined if it is positive or negative . The total score is then rated between 1 to -1.

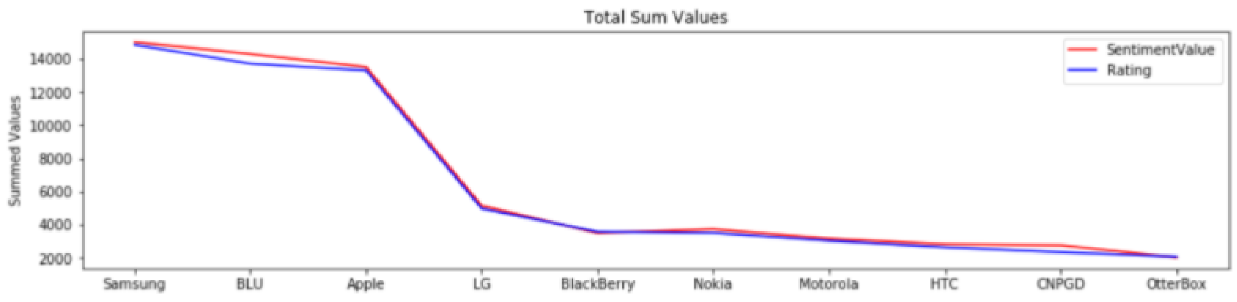
In the given example the review is very positive with the only negative being the camera quality which is considered as a minor setback. Hence an overall score of 0.9 is awarded to it.

In this fashion the data is analyzed and stored in the form of a table. The table then contains an addition of two features , sentiment that would indicate the overall sentiment of the sentence which could be Very Positive , Positive , Neutral , Negative and Very Negative. And sentiment value that is a numerical representation of the sentiment of the sentence.

4.1.4 Conclusion

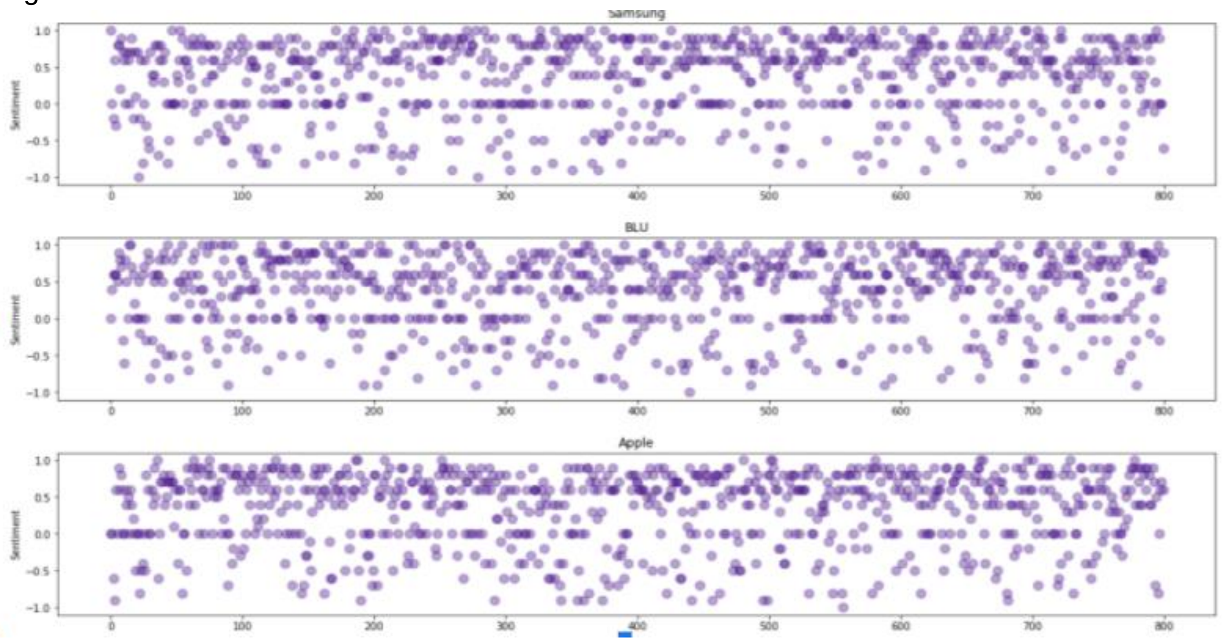
Next, we compare the result derived from the sentiment value against the ratings in the dataset.





This indicates how close the prediction is against the actual ratings given by the user. The model has an 82% accuracy when matched with the ratings[7].

Next, we consider the top 3 products namely Apple , Samsung, and BLU. We can see that the sentiments ranging from 1 to -1 indicate that there are a larger number of positive reviews over negative reviews.



4.1.5 Inference

From this sentiment analysis we can conclude the following

- Sentiment concentration towards positivity decreases as we move from top to lower brands.
- Population towards negativity and neutrality keeps on increasing as we move downwards.
- Sentiment analysis that was derived matches when we sort the data in accordance with the mean rating.

4.2 Implementation Part 2

In the second part of the implementation , the aim is to evaluate and analyze supervised models (namely Logistic Regression , Naive bayes and Random Forest Classifier) for sentiment analysis on product reviews.

4.2.1 Bag of Words

In this model we use the concept known as bag of words. The bag-of-words model is a simplifying representation used in natural language processing and information retrieval.
In this model , a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. The bag-of-words

model is commonly used in methods of document classification where the (frequency of) occurrence of each word is used as a feature for training a classifier.

The Goal of this part of the implementation is the following:

- Classify the reviews into positive and negative sentiment.
- Two main steps involved
    - Convert words in document into a numerical representation.
    - Input the numerical representations of text to machine learning algorithms.

The most common approach of input representation is the Frequency based embedding such as Bag of Words (BOW) model. BOW model learns a vocabulary list from a given corpus and represents each document based on some counting methods of words. From these we can explore the model performance of using BOW with supervised learning algorithms.

4.2.2 Workflow BOW

In this experiment the procedure is the following

- Process the raw reviews into cleaned reviews.
- Create BOW using Count Vectorizer / Tfidf vectorizer with the help of sklearn
- Transform the review text to numerical representation (i.e., feature vectors)
- Fit the feature vectors to supervised learning algorithm (e.g., Naive Bayes , Logistic regression etc.)
- Evaluate the supervised learning models.

However, before the commencement of the experiment we require the data to be processed. Hence a text processing is implemented to convert the raw reviews into cleaned ones. This is done to make it easier for us to do feature extraction in the next step.

Text Processing is done by :

- Removing tags with the use of Beautiful soup.
- Remove non-character such as digits and symbols.
- Stop words are removed if required.

Once the preparation of the data is complete , it is then converted into numerical representations for the machine learning algorithm. With the use of Count Vectorizer that creates a document -term frequency for all the reviews . This model's output is then represented with the use of a sparse matrix representation.

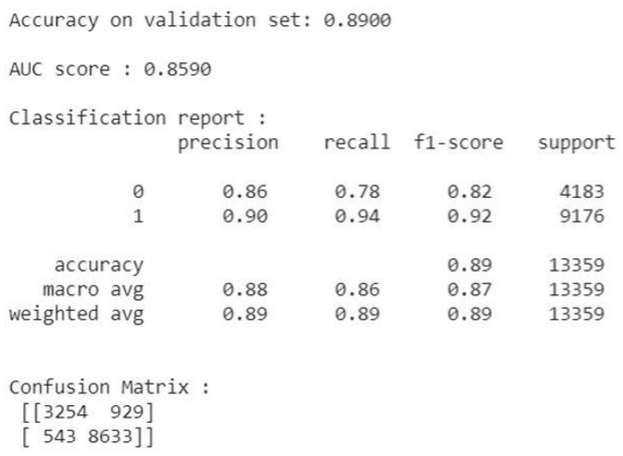
The model is then evaluated with the use of various evaluation metrics i.e., accuracy , precision and recall , AUC-ROC curve and finally a confusion matrix.

4.2.3 Sentiment analysis of product reviews using Supervised models

4.2.3.1 Naive Bayes Classifier:

The words are converted into a document term matrix , this is then fit and transformed to the dataset. We train the document with the use of a multinomial naive bayes classifier.

The performance of the naive bayes model on the test set is shown in the following figure:

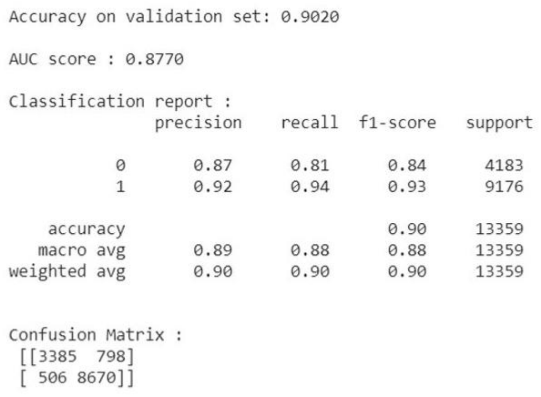


```
Accuracy on validation set: 0.8900

AUC score : 0.8590

Classification report :
              precision    recall  f1-score   support

           0       0.86      0.78      0.82      4183
           1       0.90      0.94      0.92      9176

    accuracy                           0.89     13359
   macro avg       0.88      0.86      0.87     13359
weighted avg       0.89      0.89      0.89     13359


Confusion Matrix :
 [[3254  929]
 [ 543 8633]]
```

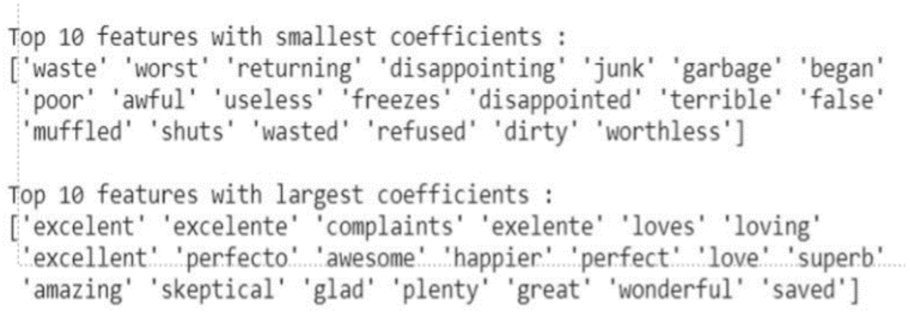
4.2.3.2 Logistic Regression with Count Vectorizer and TFIDF Vectorizer

In this case some words might frequently appear but have little meaningful information about the sentiment of a particular review. Thus, instead of using occurrence counting. We can use TFIDF vectorizer which implements both tokenization and tf-idf weighted counting in a single class.

The methodology of this part is to fit and transform the data to a document-term matrix using tfidf Vectorizer. Logistic regression is then performed on this dataset. Next, we look at the top 20 features with the smallest and the largest coefficients. The results of evaluating using logistic regression count vectorization is as follows.

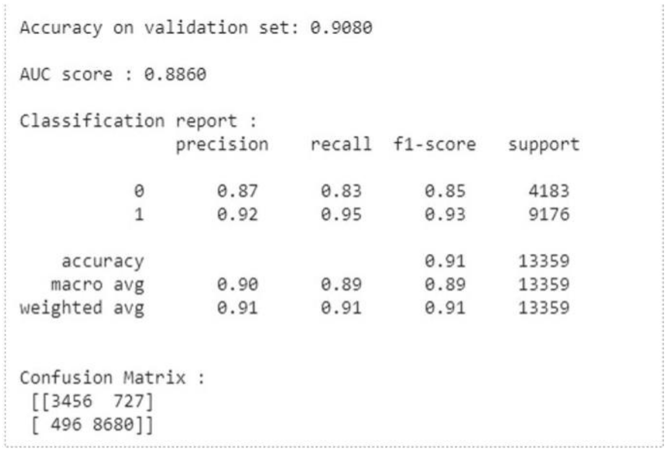


```
Accuracy on validation set: 0.9020

AUC score : 0.8770

Classification report :
              precision    recall  f1-score   support

           0       0.87      0.81      0.84      4183
           1       0.92      0.94      0.93      9176

    accuracy                           0.90     13359
   macro avg       0.89      0.88      0.88     13359
weighted avg       0.90      0.90      0.90     13359


Confusion Matrix :
 [[3385  798]
 [ 506 8670]]
```


```
Top 10 features with smallest coefficients :
['waste' 'worst' 'returning' 'disappointing' 'junk' 'garbage' 'began'
 'poor' 'awful' 'useless' 'freezes' 'disappointed' 'terrible' 'false'
 'muffled' 'shuts' 'wasted' 'refused' 'dirty' 'worthless']

Top 10 features with largest coefficients :
['excelent' 'excelente' 'complaints' 'exelente' 'loves' 'loving'
 'excellent' 'perfecto' 'awesome' 'happier' 'perfect' 'love' 'superb'
 'amazing' 'skeptical' 'glad' 'plenty' 'great' 'wonderful' 'saved']
```

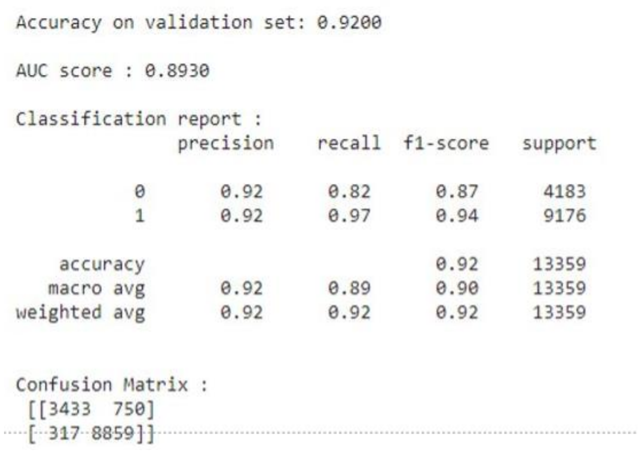
TF-IDF

The main problem with scoring word frequency is that highly frequent words start to dominate in the document though it might not contain much information or content. Thus, rescaling the frequency of words that are also frequent across all documents are penalized. On applying tfidf vectorization the output that was received is given bellow:


```
Accuracy on validation set: 0.9080

AUC score : 0.8860

Classification report :
              precision    recall  f1-score   support

           0       0.87      0.83      0.85      4183
           1       0.92      0.95      0.93      9176

    accuracy                           0.91     13359
   macro avg       0.90      0.89      0.89     13359
weighted avg       0.91      0.91      0.91     13359


Confusion Matrix :
 [[3456  727]
 [ 496 8680]]
```

4.2.3.3 Random Forest Classifier

We also implemented random forest classifier against product reviews dataset. The performance of this model seems to be better than previous supervised models. The training time is more compared to other models as many decision trees are created and ensembled in the process. The performance of this model is given below.

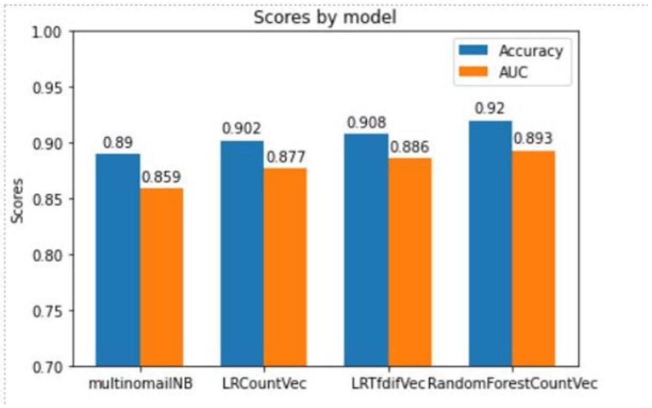

```
Accuracy on validation set: 0.9200

AUC score : 0.8930

Classification report :
              precision    recall  f1-score   support

           0       0.92      0.82      0.87      4183
           1       0.92      0.97      0.94      9176

    accuracy                           0.92     13359
   macro avg       0.92      0.89      0.90     13359
weighted avg       0.92      0.92      0.92     13359


Confusion Matrix :
 [[3433  750]
 [ 317 8859]]
```

4.2.4 Challenges

1. Class Imbalance: The dataset contains a greater number of positive sentiments than negative sentiments. This will bias the model to learn positive sentiment patterns more than negative sentiments. In turn outputting more false positives.
2. Sparsity : Representing the input document in the form of count vectors will create a sparse matrix that is fed into the model. These sparse vectors consume lot of space and will make the model take greater time to learn.
3. Context: Context/Meaning of the reviews is not taken into consideration in the bag of words model.
4. Time to train: Random forest takes huge amount to time to train compared to other models.

4.2.4 Inference

To conclude this part of the experiment we compared the scores of the supervised models used. The random forest classifier performed the best among the different models following close second is the Logistic regression with tf-idf. The performance of various models is shown in the below figure. As the complexity of the model increases the AUC and accuracy is increased.



4.3 Implementation Part 3

In the third part of this experiment the aim is to evaluate and Analyze Long Short-Term Memory (LSTM), Word2Vec, Gated Recurrent (GRU), Dense Layers and Glove Models.

4.3.1 Word2Vec

In this model the main features that were tested were word embedding and word2vec. Word embedding is one of the most popular representations of document vocabulary, it must be capable of capturing context of a word in a document , semantics and syntactic similarity , relation with other words.
Word2vec is a common approach of word embedding is prediction-based embedding, such as the word2vec model. Word2vec can be obtained using two methods :

- Skip Gram
- Continuous Bag of Words(CBOW)

4.3.2 CBOW Model

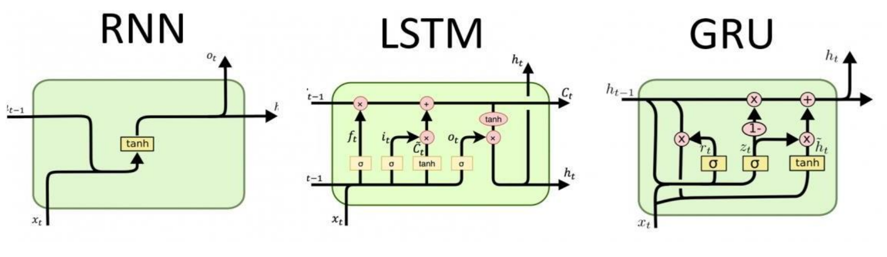
This model takes the context of each word as the input and it tries to predict the word corresponding to the context. Both the gram model and the CBOW map the words . They also learn weights which act as word vector representations.
The main problems that are faced are , the weights are getting updated in the back propagation in chain rule, the weights are becoming smaller and smaller due to which we are reducing Vanishing Gradient. To overcome these , the model considers LSTM and GRU models.

4.3.3 Long Short-Term Memory Networks (LSTM)

These are special kind of Recurrent Neural Networks (RNN) capable of learning long-term dependencies. It Involves dependencies in the sentences which can be caught in the "memory" of the LSTM.

4.3.4 Gated Recurrent Units (GRU) and difference from LSTM

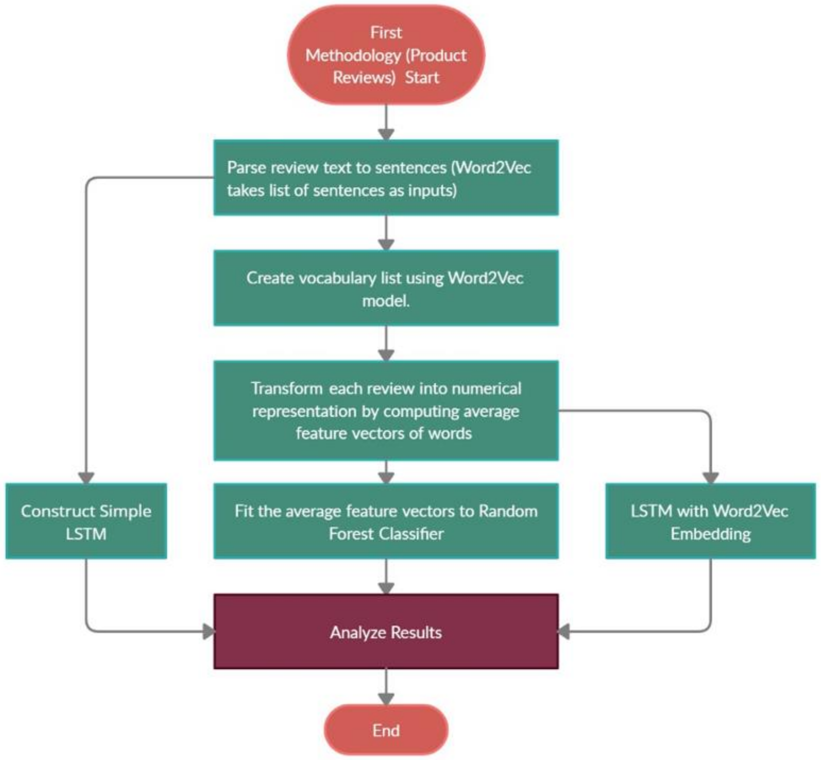
The GRU is like a long short-term memory (LSTM) with a forget gate, but has fewer parameters than LSTM, as it lacks an output gate. GRU is less complex than LSTM because it has a smaller number of gates. If the dataset is small, then GRU is preferred otherwise LSTM for the larger dataset. GRU exposes the complete memory and hidden layers but LSTM does not.

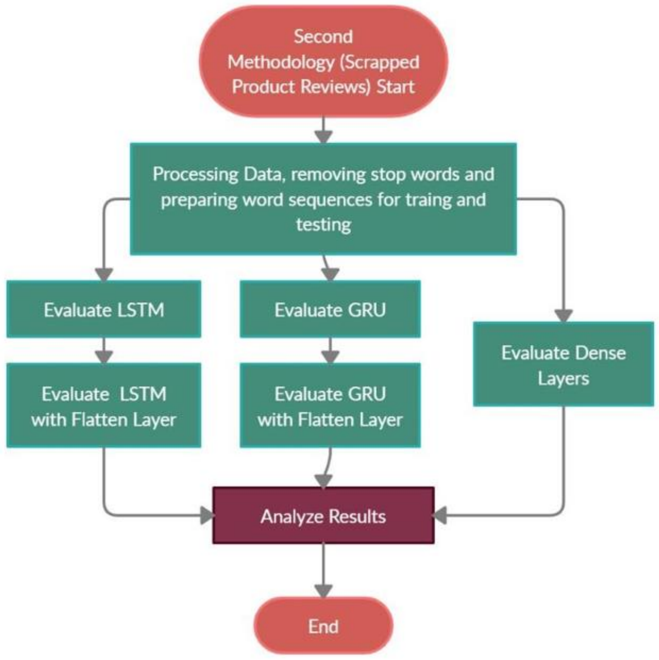


4.3.5 Global Vectors for Word Representation (GloVe)

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

4.3.6 Experimental Setups


First Methodology (Product Reviews) Start
Parse review text to sentences (Word2Vec takes list of sentences as inputs)
Create vocabulary list using Word2Vec model.
Transform each review into numerical representation by computing average feature vectors of words
Construct Simple LSTM
Fit the average feature vectors to Random Forest Classifier
LSTM with Word2Vec Embedding
Analyze Results
End


Second Methodology (Scrapped Product Reviews) Start
Processing Data, removing stop words and preparing word sequences for traing and testing
Evaluate LSTM
Evaluate GRU
Evaluate Dense Layers
Evaluate LSTM with Flatten Layer
Evaluate GRU with Flatten Layer
Analyze Results
End

4.3.6.1
Experimental Methodology 1:

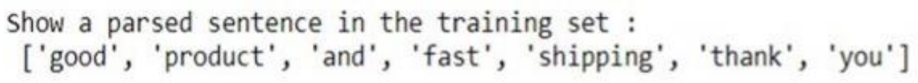
The methodology is to train Word2Vec model to create our own word vector representations using genism library. Next we fit the feature vectors of the reviews to Random Forest Classifier. And LSTM classifier and LSTM with Word2Vec Model.

Workflow:

- Step 1 : Parse review text to sentences (Word2Vec model takes a list of sentences as inputs)
- Step 2 : Create vocabulary list using Word2Vec model
- Step 3 : Transform each review into numerical representation by computing average feature vectors of words therein
- Step 4 : Fit the average feature vectors to Random Forest Classifier

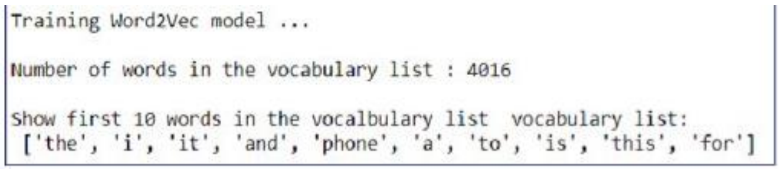
Step-1: Parsing Review into Sentences

Word2Vec model takes a list of sentences as inputs and outputs word vector representations for words in the vocabulary list created .Before we train the Word2Vec model, we must parse reviews in the training set into sentences.


```
Show a parsed sentence in the training set :
 ['good', 'product', 'and', 'fast', 'shipping', 'thank', 'you']
```

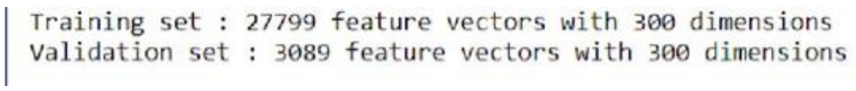
Step-2: Creating Vocabulary List using Word2Vec Model

Now we have a set of cleaned and parsed sentences from the training data, train our own word vector representations by specifying the embedding dimension.


```
Training Word2Vec model ...

Number of words in the vocabulary list : 4016

Show first 10 words in the vocalbulary list  vocabulary list:
 ['the', 'i', 'it', 'and', 'phone', 'a', 'to', 'is', 'this', 'for']
```

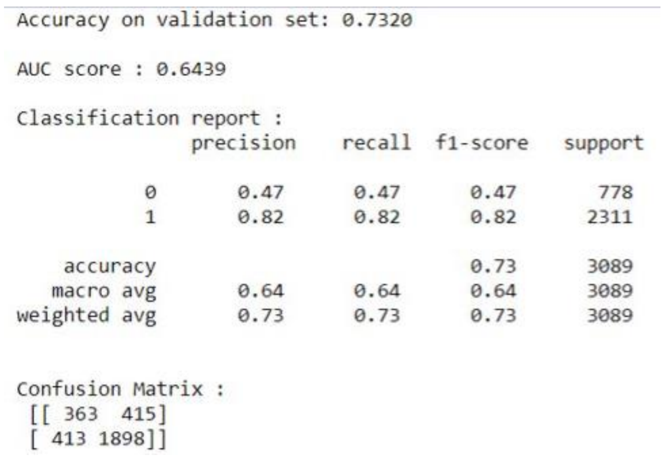
Step-3:Averaging Feature Vectors

we have created a vocabulary list of words each word having a word representation (i.e., feature vector of dim 300). To find a numerical representation for a review and run through each word in a review text ,words appear in the vocabulary list, we compute the average feature vectors of all those words. The average feature vector is the numerical representation of the review.


```
Training set : 27799 feature vectors with 300 dimensions
Validation set : 3089 feature vectors with 300 dimensions
```

Step-4:Random Forest Classifier

We can now train Random Forest Classifier using feature vectors of reviews in the training set.


```
Accuracy on validation set: 0.7320

AUC score : 0.6439

Classification report :
               precision    recall  f1-score   support

           0       0.47      0.47      0.47       778
           1       0.82      0.82      0.82      2311

    accuracy                           0.73      3089
   macro avg       0.64      0.64      0.64      3089
weighted avg       0.73      0.73      0.73      3089


Confusion Matrix :
 [[ 363  415]
 [ 413 1898]]
```

4.3.6.2 Applying LSTM to the Model

Preprocessing:

We need to preprocess the text data to 2D tensor before we fit into a simple LSTM. this data is then tokenized according to the corpus by only considering top words (top words = 20000), next we transform reviews to numerical sequences using the trained tokenizer. It is to make sure that all numerical sequences have the same length (maxlen=100) for modeling and truncating long reviews and pad shorter reviews with zero values.

To construct a simple LSTM:

Using embedding class in Keras to construct the first layer we create an embedding layer which converts numerical sequence of words into a word embedding. The next layer is the LSTM layer with 128 memory units. This is done using a dense output layer with a single neuron and a sigmoid activation function to make 0 or 1 predictions for the two classes (positive sentiment and negative sentiment). Since it is a binary classification problem, log loss is used as the loss function (binary cross entropy in Keras). ADAM optimization algorithm is used.

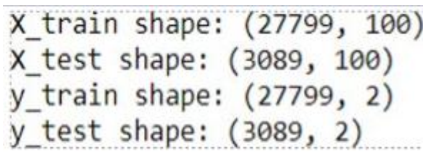
Workflow LSTM

The overall workflow of this process is as follows:

- Step 1 : Prepare X_train and X_test to 2D tensor
- Step 2 : Train a simple LSTM (embedding layer => LSTM layer => dense layer)
- Step 3 : Compile and fit the model using log loss function and ADAM optimizer

Step 1: Prepare X_train and X_test to 2D tensor

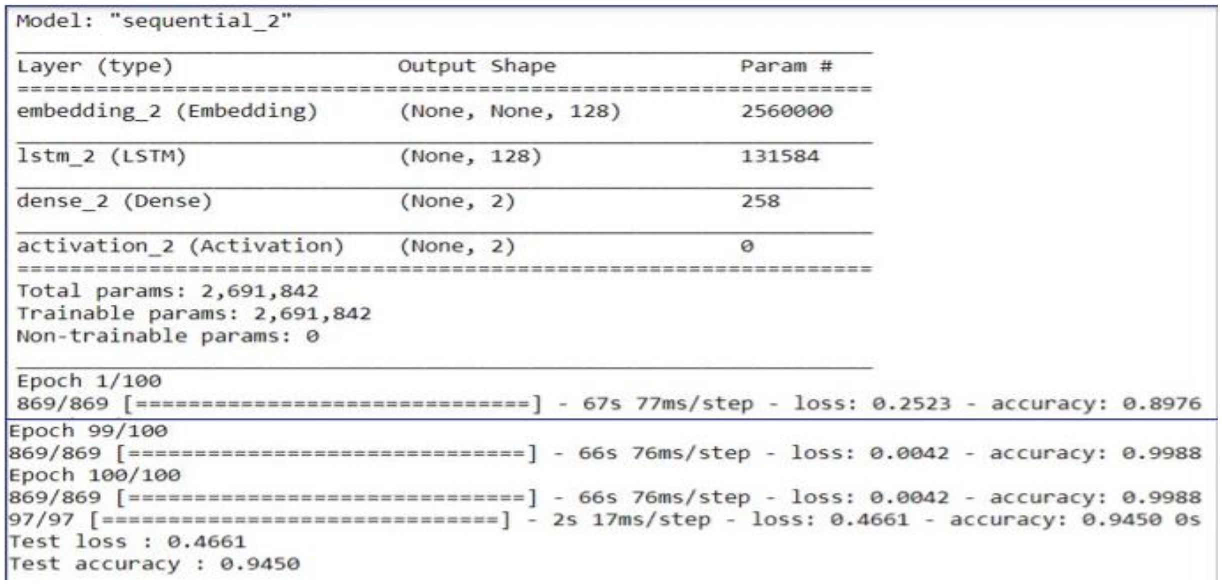
To start this part of the experiment first we ,vectorize X_train and X_test to 2D tensor and only consider the top 20000 words in the corpse. We use the tokenizer.word_index which provides access to the word-to-index dictionary of trained tokenizer and finally we perform the one-hot encoding of y_train and y_test.



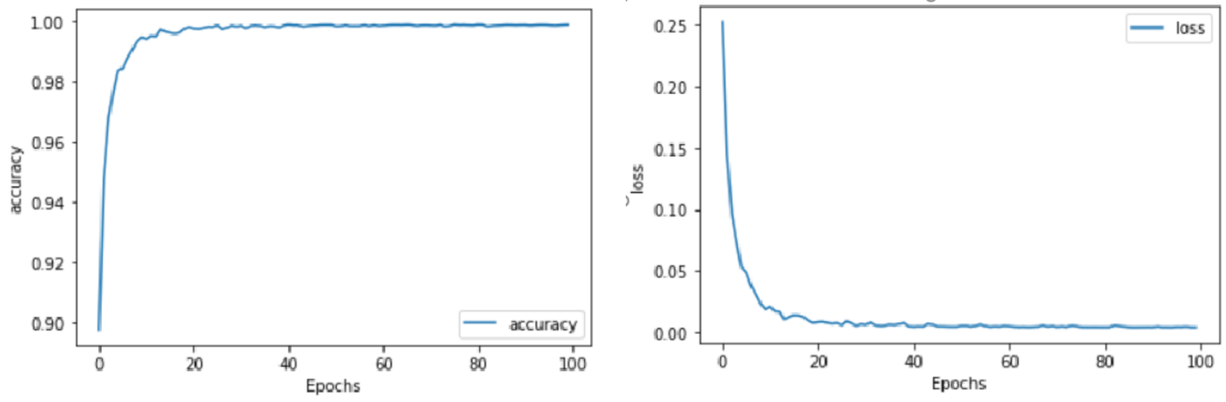
```
X_train shape: (27799, 100)
X_test shape: (3089, 100)
y_train shape: (27799, 2)
y_test shape: (3089, 2)
```

Step 2: Train a simple LSTM (embedding layer => LSTM layer => dense layer)
Step3: Compile and fit the model using log loss function and ADAM optimizer

Initially a simple LSTM is constructed, and the model is compiled and evaluated.



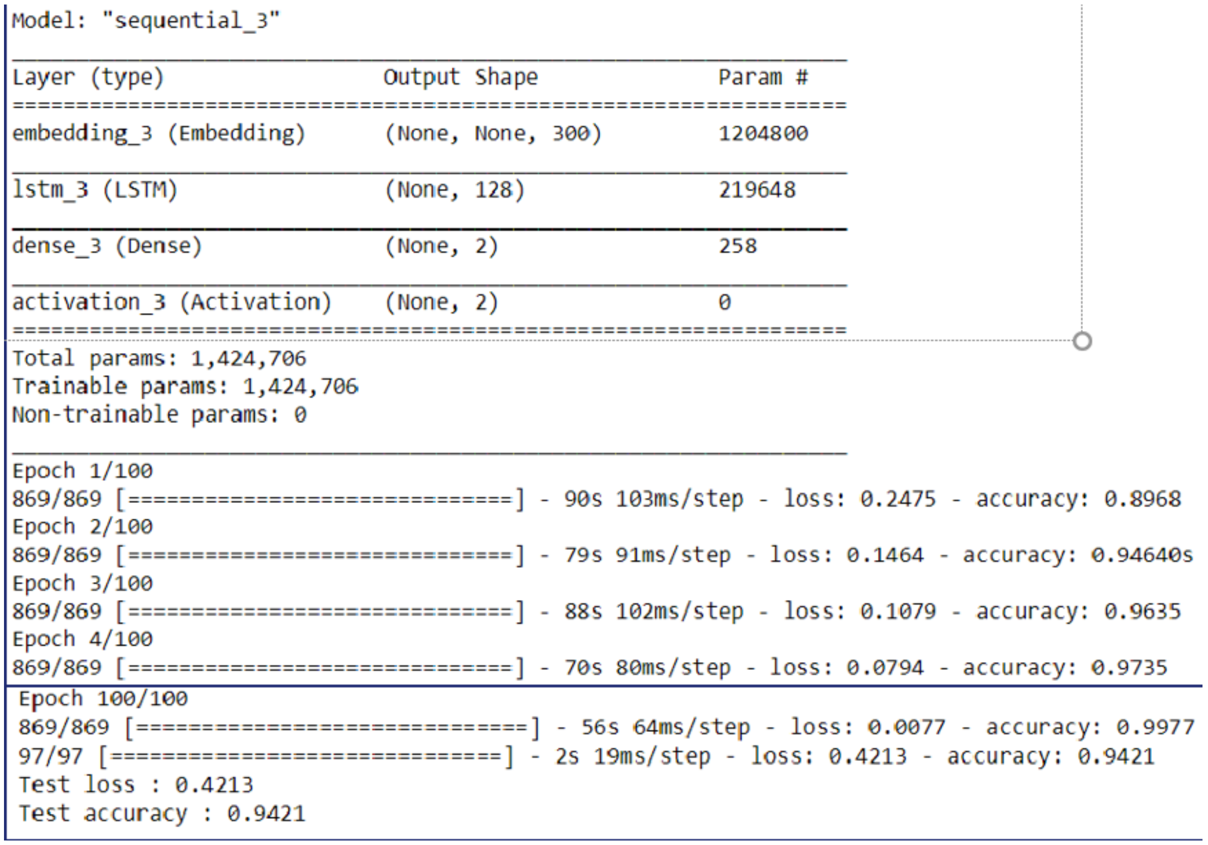
```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_2 (Embedding)      (None, None, 128)         2560000   
_________________________________________________________________
lstm_2 (LSTM)                (None, 128)               131584    
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 258       
_________________________________________________________________
activation_2 (Activation)    (None, 2)                 0         
=================================================================
Total params: 2,691,842
Trainable params: 2,691,842
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
869/869 [==============================] - 67s 77ms/step - loss: 0.2523 - accuracy: 0.8976
Epoch 99/100
869/869 [==============================] - 66s 76ms/step - loss: 0.0042 - accuracy: 0.9988
Epoch 100/100
869/869 [==============================] - 66s 76ms/step - loss: 0.0042 - accuracy: 0.9988
97/97 [==============================] - 2s 17ms/step - loss: 0.4661 - accuracy: 0.9450 0s
Test loss : 0.4661
Test accuracy : 0.9450
```



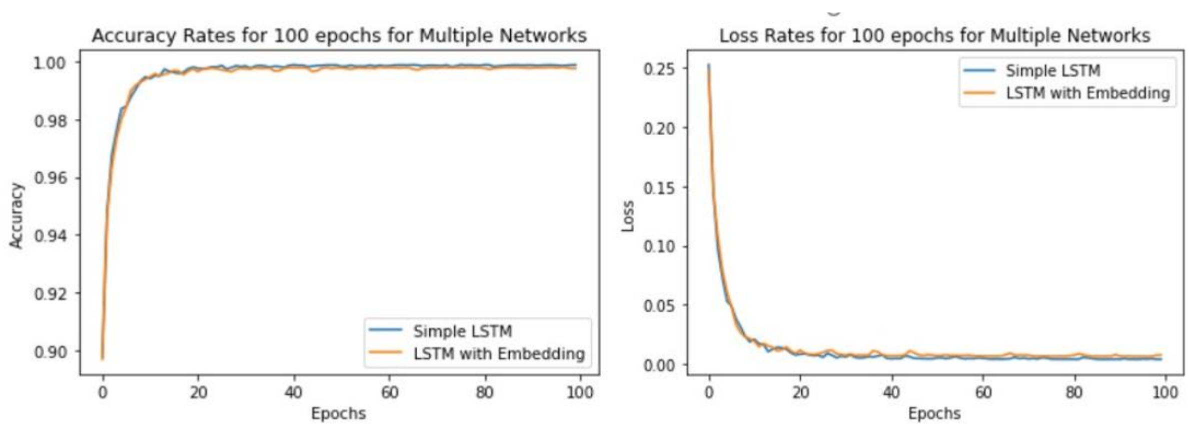
4.3.6.3 LSTM with Word2Vec Embedding

In the simple LSTM model constructed above it can be seen that the embedding class in Keras comes in handy to converts numerical sequence of words into a word embedding, it does not take the semantic similarity of the words into account. The model assigns random weights to the embedding layer and learn the embeddings by minimizing the global error of the network. Thus Instead of using random weights, we can use pre trained word embeddings to initialize the weight of an embedding layer. Hence we use the Word2Vec embedding trained in the previous part to initialize the weights of embedding layer in LSTM.

Method -1	Models/Approaches	Test Accuracies Obtained
1.	Random Forest	0.73
2.	LSTM	0.9450
3.	LSTM with Embedding	0.94218



```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, None, 300)         1204800
_________________________________________________________________
lstm_3 (LSTM)                (None, 128)               219648
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 258
_________________________________________________________________
activation_3 (Activation)    (None, 2)                 0
=================================================================
Total params: 1,424,706
Trainable params: 1,424,706
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
869/869 [==============================] - 90s 103ms/step - loss: 0.2475 - accuracy: 0.8968
Epoch 2/100
869/869 [==============================] - 79s 91ms/step - loss: 0.1464 - accuracy: 0.94640s
Epoch 3/100
869/869 [==============================] - 88s 102ms/step - loss: 0.1079 - accuracy: 0.9635
Epoch 4/100
869/869 [==============================] - 70s 80ms/step - loss: 0.0794 - accuracy: 0.9735
Epoch 100/100
869/869 [==============================] - 56s 64ms/step - loss: 0.0077 - accuracy: 0.9977
97/97 [==============================] - 2s 19ms/step - loss: 0.4213 - accuracy: 0.9421
Test loss : 0.4213
Test accuracy : 0.9421
```



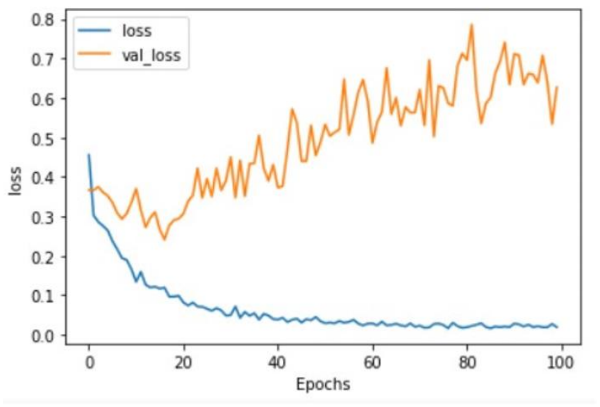
4.3.7

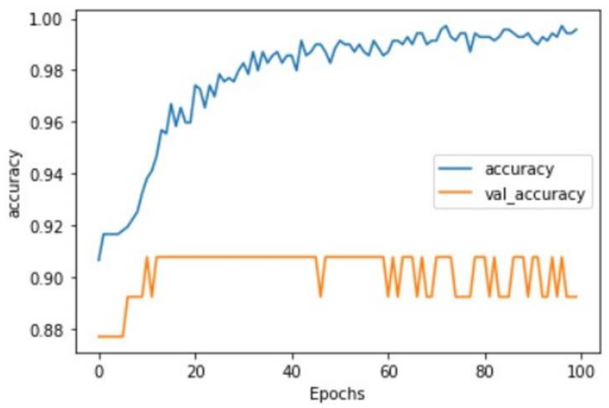
Experimental Methodology 2:

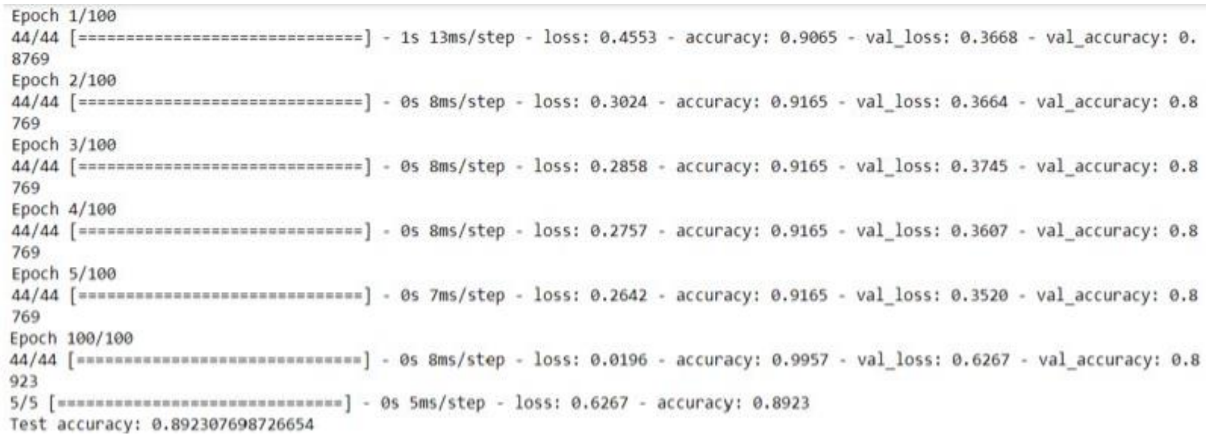
In this we implement various models and test the accuracies.

4.3.7.1 LSTM

The methodology for this part of the experiment is that in this, we use various ML methods, including gated recurrent networks from the Keras interface, on a well-known text processing example, that of a flipkart product reviews sentiment analysis. In this version the scrapped dataset from flipkart of product reviews to predict a positive or negative evaluation was used. A simple LSTM model on flipkart web scrapped reviews data and checked the accuracies and the loss values were done. The results obtained are given below.



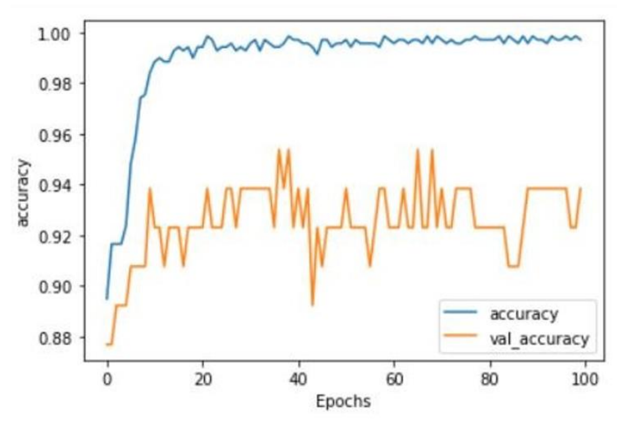


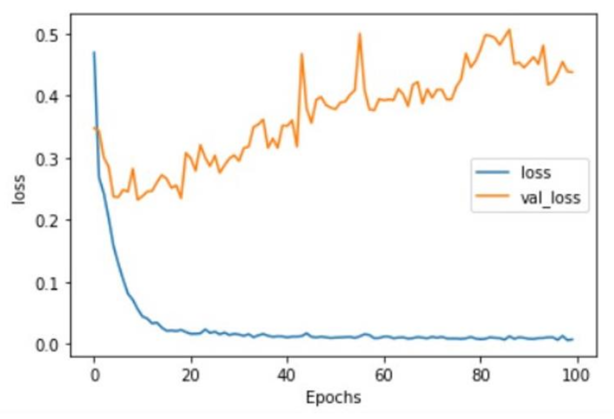


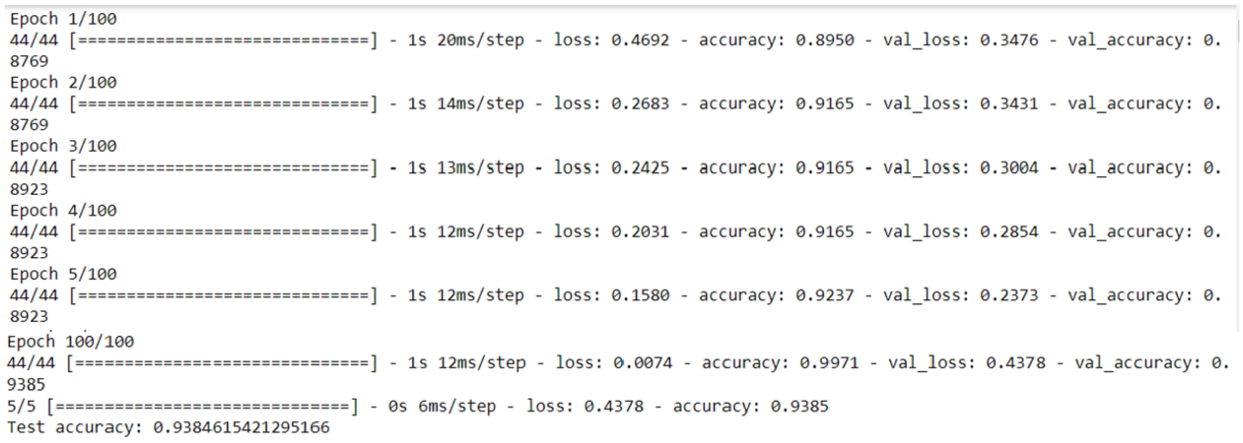
```
Epoch 1/100
44/44 [==============================] - 1s 13ms/step - loss: 0.4553 - accuracy: 0.9065 - val_loss: 0.3668 - val_accuracy: 0.
8769
Epoch 2/100
44/44 [==============================] - 0s 8ms/step - loss: 0.3024 - accuracy: 0.9165 - val_loss: 0.3664 - val_accuracy: 0.8
769
Epoch 3/100
44/44 [==============================] - 0s 8ms/step - loss: 0.2858 - accuracy: 0.9165 - val_loss: 0.3745 - val_accuracy: 0.8
769
Epoch 4/100
44/44 [==============================] - 0s 8ms/step - loss: 0.2757 - accuracy: 0.9165 - val_loss: 0.3607 - val_accuracy: 0.8
769
Epoch 5/100
44/44 [==============================] - 0s 7ms/step - loss: 0.2642 - accuracy: 0.9165 - val_loss: 0.3520 - val_accuracy: 0.8
769
Epoch 100/100
44/44 [==============================] - 0s 8ms/step - loss: 0.0196 - accuracy: 0.9957 - val_loss: 0.6267 - val_accuracy: 0.8
923
5/5 [==============================] - 0s 5ms/step - loss: 0.6267 - accuracy: 0.8923
Test accuracy: 0.892307698726654
```

4.3.7.2 LSTM with Flatten Layer

Next an LSTM with Flatten Layer model on flipkart web scrapped reviews data and checked the accuracies and the loss values were performed. The results obtained are given below.



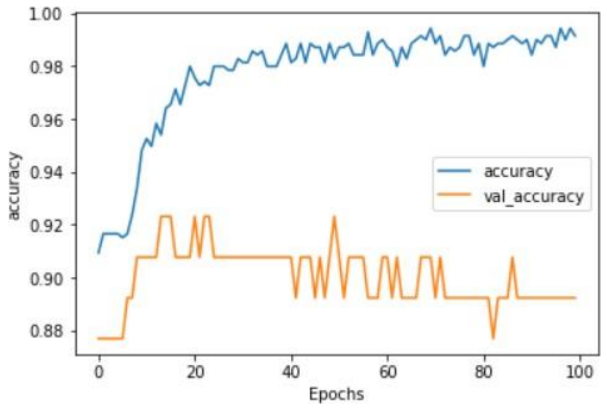


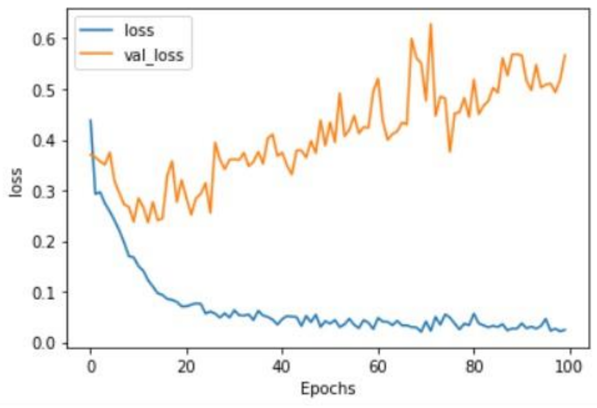


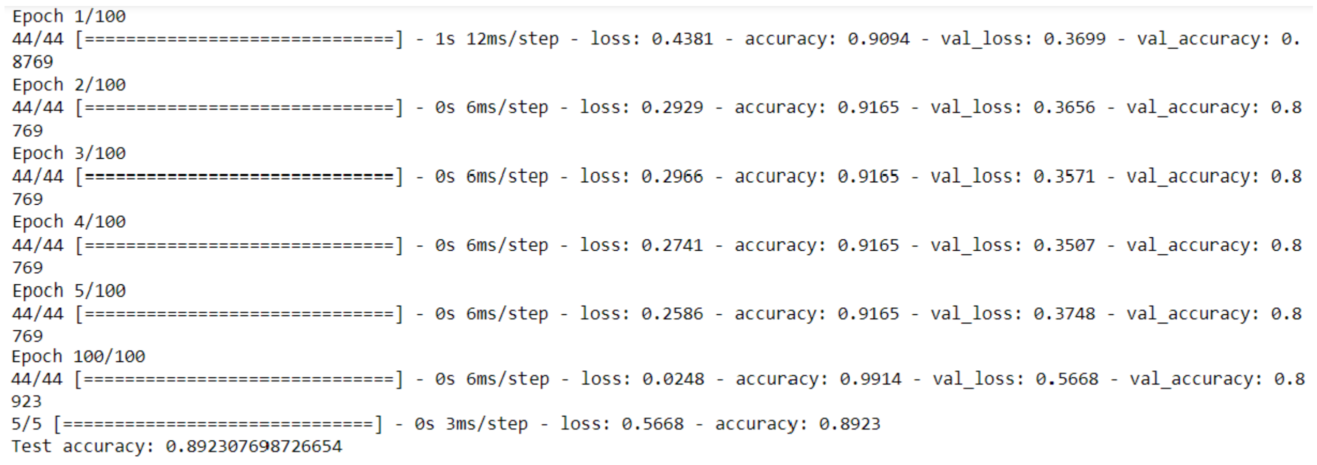
```
Epoch 1/100
44/44 [==============================] - 1s 20ms/step - loss: 0.4692 - accuracy: 0.8950 - val_loss: 0.3476 - val_accuracy: 0.
8769
Epoch 2/100
44/44 [==============================] - 1s 14ms/step - loss: 0.2683 - accuracy: 0.9165 - val_loss: 0.3431 - val_accuracy: 0.
8769
Epoch 3/100
44/44 [==============================] - 1s 13ms/step - loss: 0.2425 - accuracy: 0.9165 - val_loss: 0.3004 - val_accuracy: 0.
8923
Epoch 4/100
44/44 [==============================] - 1s 12ms/step - loss: 0.2031 - accuracy: 0.9165 - val_loss: 0.2854 - val_accuracy: 0.
8923
Epoch 5/100
44/44 [==============================] - 1s 12ms/step - loss: 0.1580 - accuracy: 0.9237 - val_loss: 0.2373 - val_accuracy: 0.
8923
Epoch 100/100
44/44 [==============================] - 1s 12ms/step - loss: 0.0074 - accuracy: 0.9971 - val_loss: 0.4378 - val_accuracy: 0.
9385
5/5 [==============================] - 0s 6ms/step - loss: 0.4378 - accuracy: 0.9385
Test accuracy: 0.9384615421295166
```

4.3.7.3 GRU

The next step is to use a Simple GRU model on the flipkart web scrapped reviews data and check the accuracies and the loss values.The results obtained are given below.



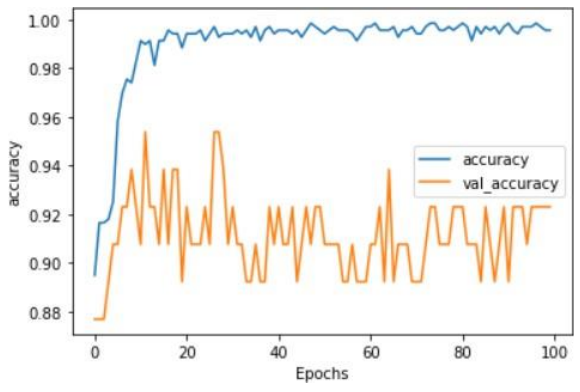


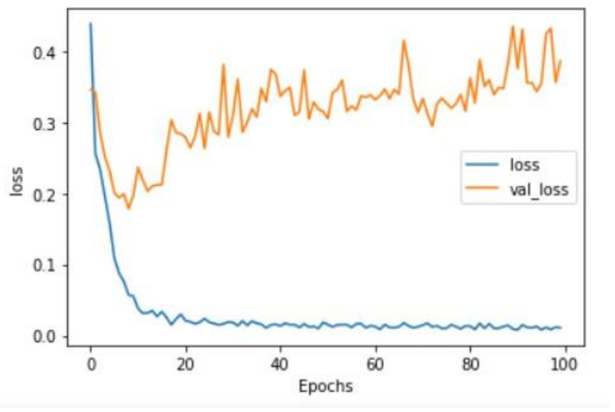


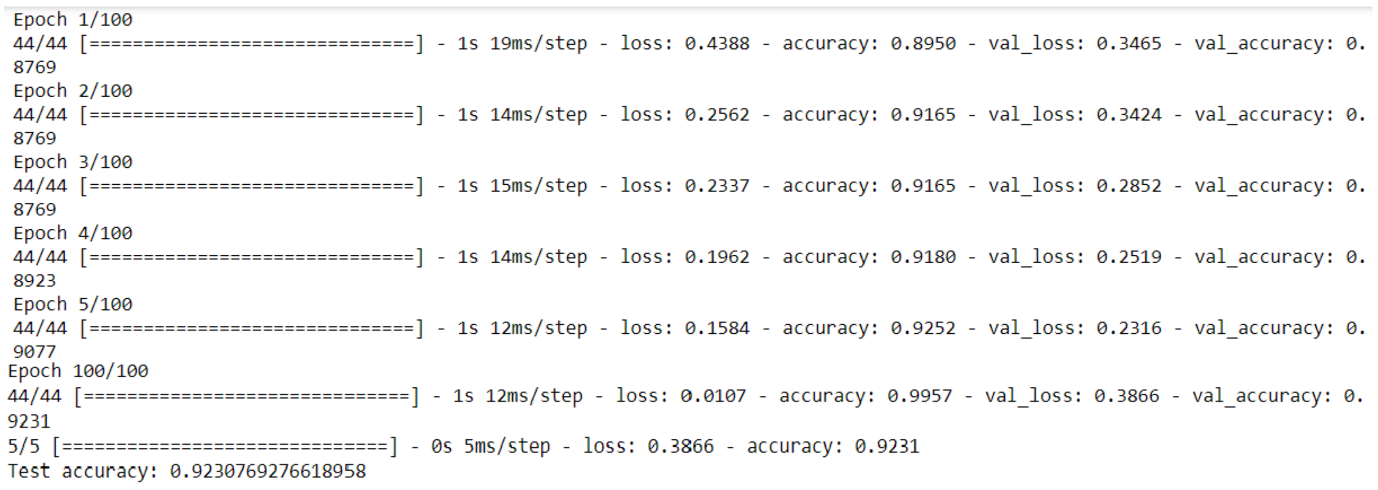
```
Epoch 1/100
44/44 [==============================] - 1s 12ms/step - loss: 0.4381 - accuracy: 0.9094 - val_loss: 0.3699 - val_accuracy: 0.
8769
Epoch 2/100
44/44 [==============================] - 0s 6ms/step - loss: 0.2929 - accuracy: 0.9165 - val_loss: 0.3656 - val_accuracy: 0.8
769
Epoch 3/100
44/44 [==============================] - 0s 6ms/step - loss: 0.2966 - accuracy: 0.9165 - val_loss: 0.3571 - val_accuracy: 0.8
769
Epoch 4/100
44/44 [==============================] - 0s 6ms/step - loss: 0.2741 - accuracy: 0.9165 - val_loss: 0.3507 - val_accuracy: 0.8
769
Epoch 5/100
44/44 [==============================] - 0s 6ms/step - loss: 0.2586 - accuracy: 0.9165 - val_loss: 0.3748 - val_accuracy: 0.8
769
Epoch 100/100
44/44 [==============================] - 0s 6ms/step - loss: 0.0248 - accuracy: 0.9914 - val_loss: 0.5668 - val_accuracy: 0.8
923
5/5 [==============================] - 0s 3ms/step - loss: 0.5668 - accuracy: 0.8923
Test accuracy: 0.892307698726654
```

4.3.7.4 GRU with Flatten Layer

In addition to the previous step a flatten layer is also added to the above model. The results obtained are given below for GRU with flatten layer are as follows:



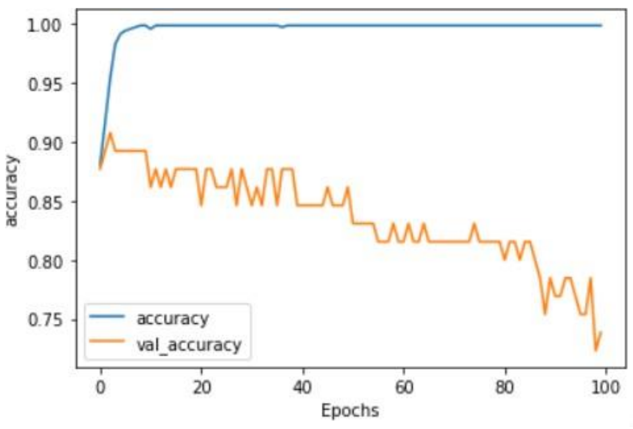


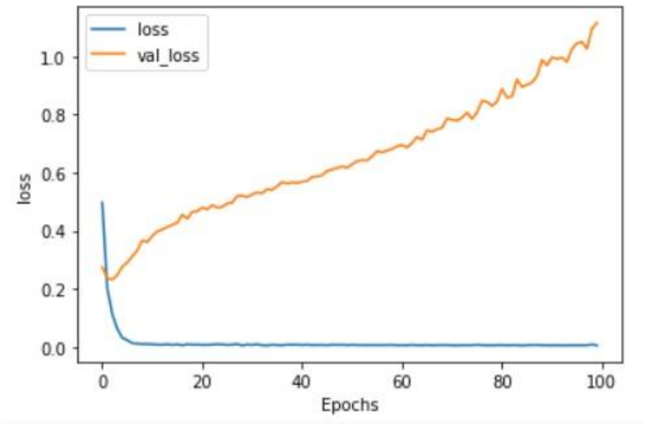


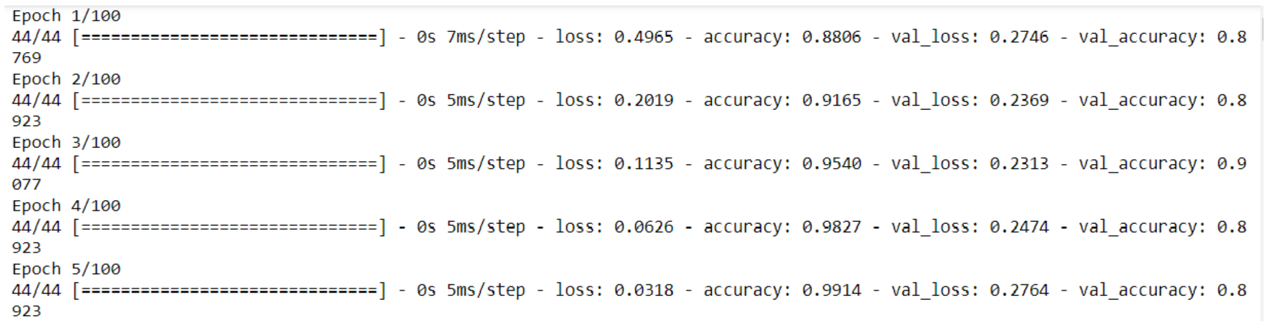
```
Epoch 1/100
44/44 [==============================] - 1s 19ms/step - loss: 0.4388 - accuracy: 0.8950 - val_loss: 0.3465 - val_accuracy: 0.
8769
Epoch 2/100
44/44 [==============================] - 1s 14ms/step - loss: 0.2562 - accuracy: 0.9165 - val_loss: 0.3424 - val_accuracy: 0.
8769
Epoch 3/100
44/44 [==============================] - 1s 15ms/step - loss: 0.2337 - accuracy: 0.9165 - val_loss: 0.2852 - val_accuracy: 0.
8769
Epoch 4/100
44/44 [==============================] - 1s 14ms/step - loss: 0.1962 - accuracy: 0.9180 - val_loss: 0.2519 - val_accuracy: 0.
8923
Epoch 5/100
44/44 [==============================] - 1s 12ms/step - loss: 0.1584 - accuracy: 0.9252 - val_loss: 0.2316 - val_accuracy: 0.
9077
Epoch 100/100
44/44 [==============================] - 1s 12ms/step - loss: 0.0107 - accuracy: 0.9957 - val_loss: 0.3866 - val_accuracy: 0.
9231
5/5 [==============================] - 0s 5ms/step - loss: 0.3866 - accuracy: 0.9231
Test accuracy: 0.9230769276618958
```

4.3.7.5 Dense Layer

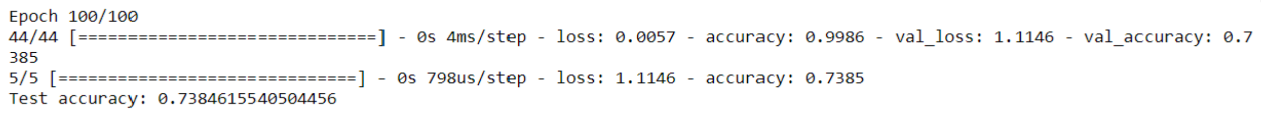
And finally, a Dense layer model is used on the flipkart reviews data and the accuracy and the loss values are obtained.





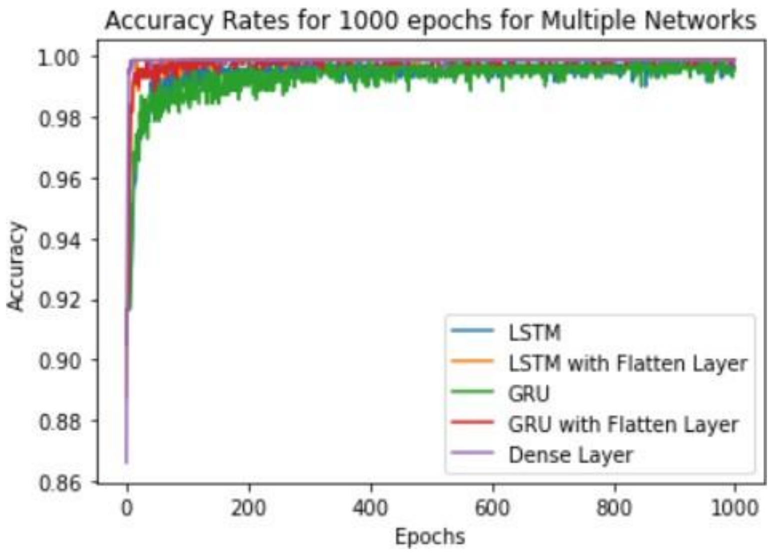


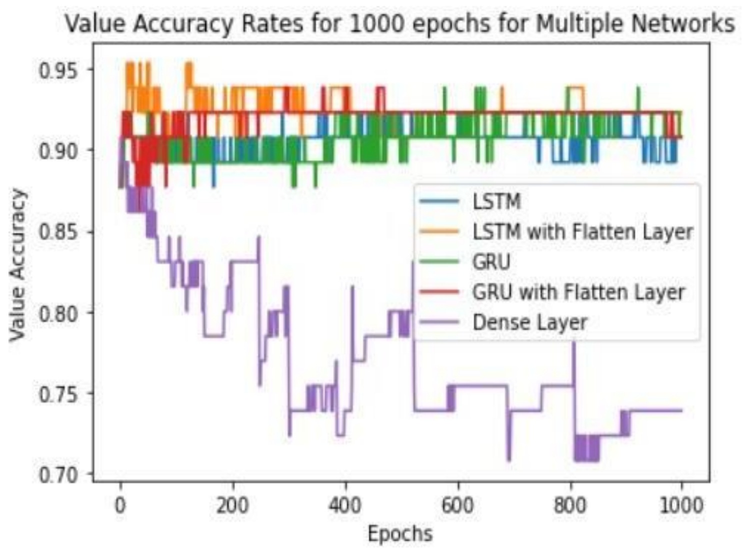
```
Epoch 1/100
44/44 [==============================] - 0s 7ms/step - loss: 0.4965 - accuracy: 0.8806 - val_loss: 0.2746 - val_accuracy: 0.8
769
Epoch 2/100
44/44 [==============================] - 0s 5ms/step - loss: 0.2019 - accuracy: 0.9165 - val_loss: 0.2369 - val_accuracy: 0.8
923
Epoch 3/100
44/44 [==============================] - 0s 5ms/step - loss: 0.1135 - accuracy: 0.9540 - val_loss: 0.2313 - val_accuracy: 0.9
077
Epoch 4/100
44/44 [==============================] - 0s 5ms/step - loss: 0.0626 - accuracy: 0.9827 - val_loss: 0.2474 - val_accuracy: 0.8
923
Epoch 5/100
44/44 [==============================] - 0s 5ms/step - loss: 0.0318 - accuracy: 0.9914 - val_loss: 0.2764 - val_accuracy: 0.8
923
```

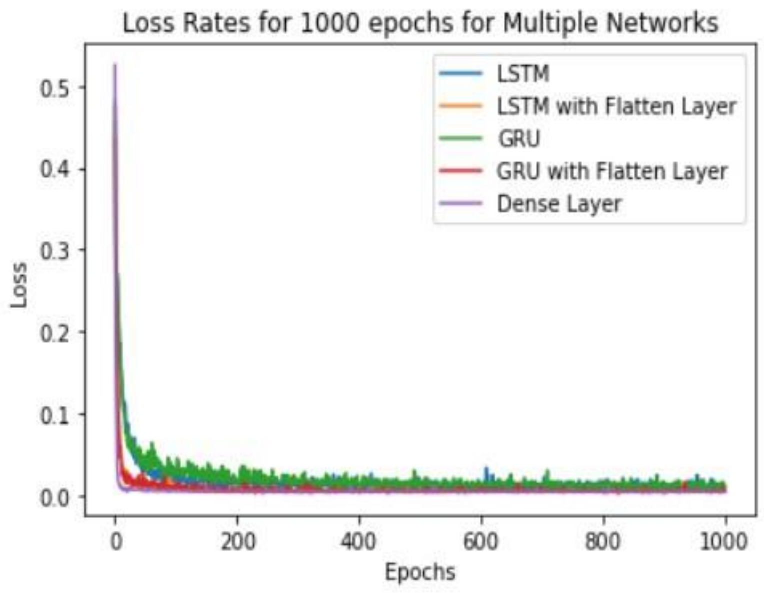


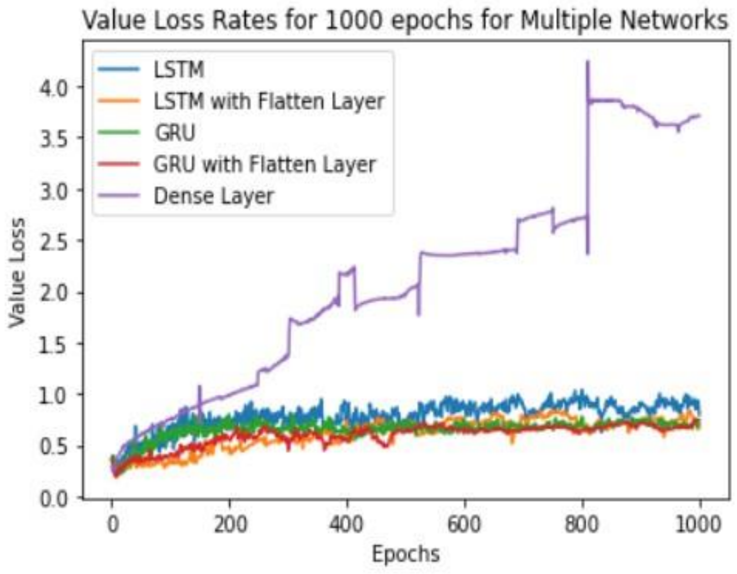
```
Epoch 100/100
44/44 [==============================] - 0s 4ms/step - loss: 0.0057 - accuracy: 0.9986 - val_loss: 1.1146 - val_accuracy: 0.7
385
5/5 [==============================] - 0s 798us/step - loss: 1.1146 - accuracy: 0.7385
Test accuracy: 0.7384615540504456
```

4.3.7.6 All Approaches Mixed Accuracy and Loses for 1000 epochs

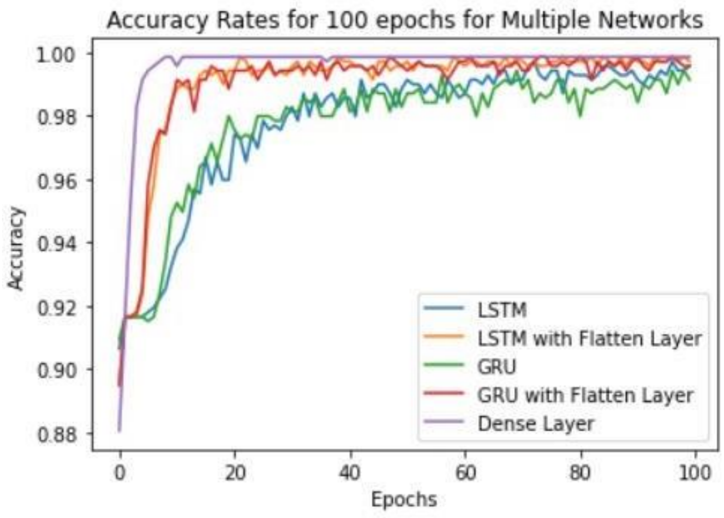


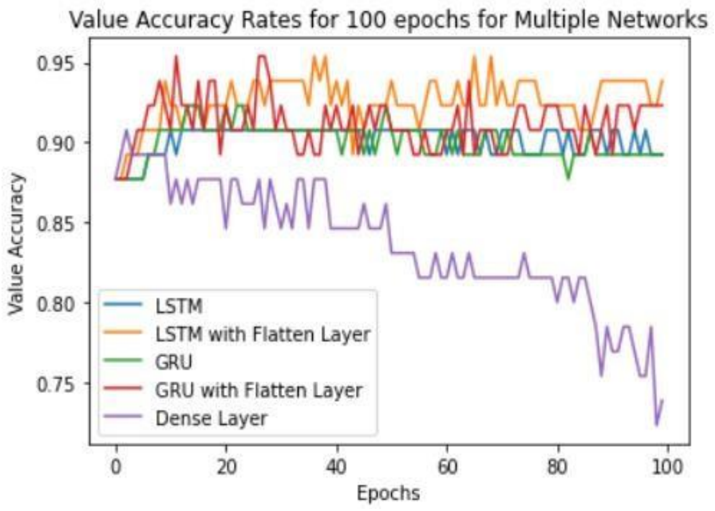


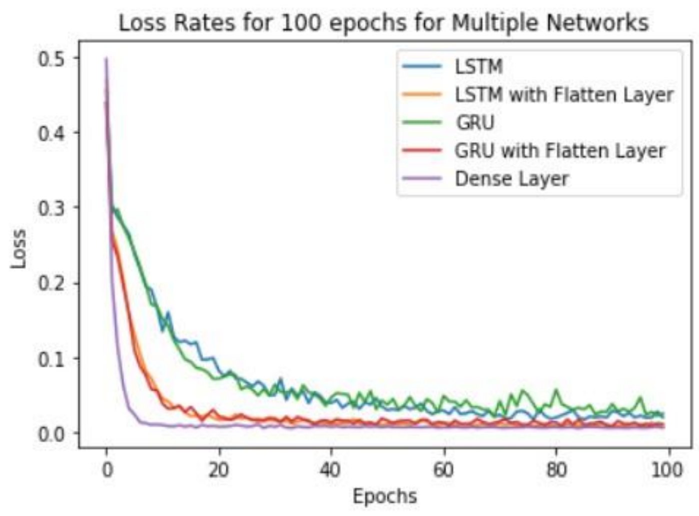


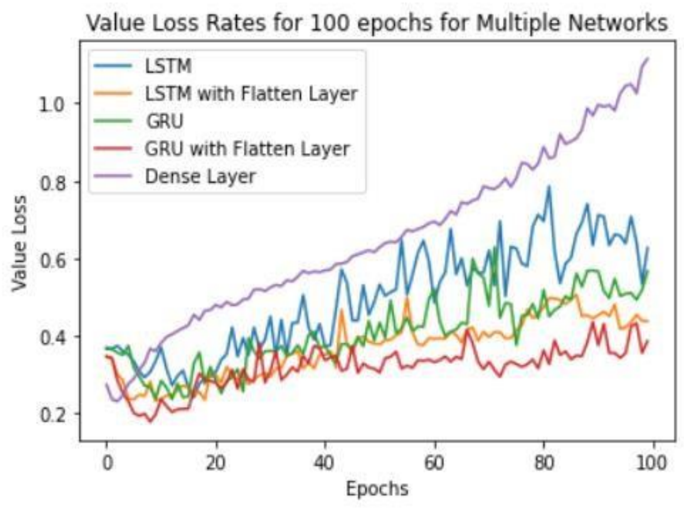


4.3.7.7 All Approaches Mixed Accuracy and Loses for 100 epochs





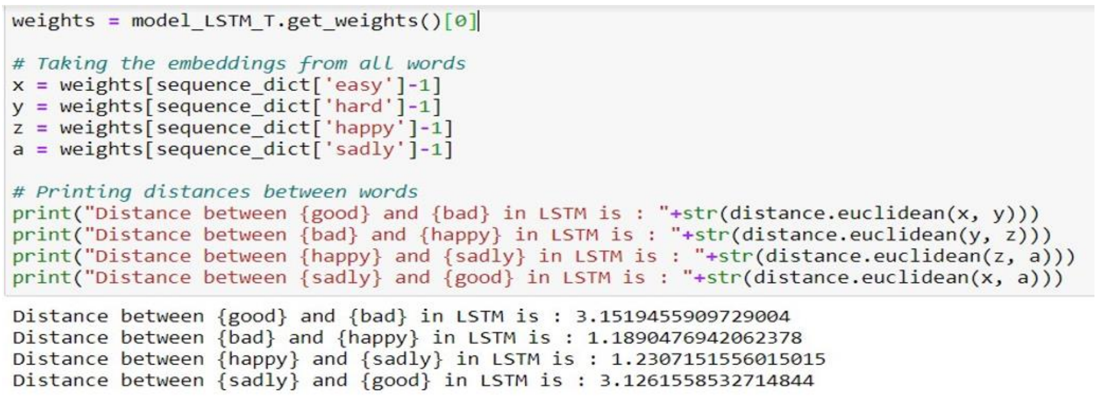




Method -2	Models/Approaches	Test Accuracies Obtained
1.	LSTM	0.907692313
2.	LSTM with Flatten Layer	0.923076928
3.	GRU	0.892307699
4.	GRU with Flatten Layer	0.923076928
5.	Dense Layer	0.738461554

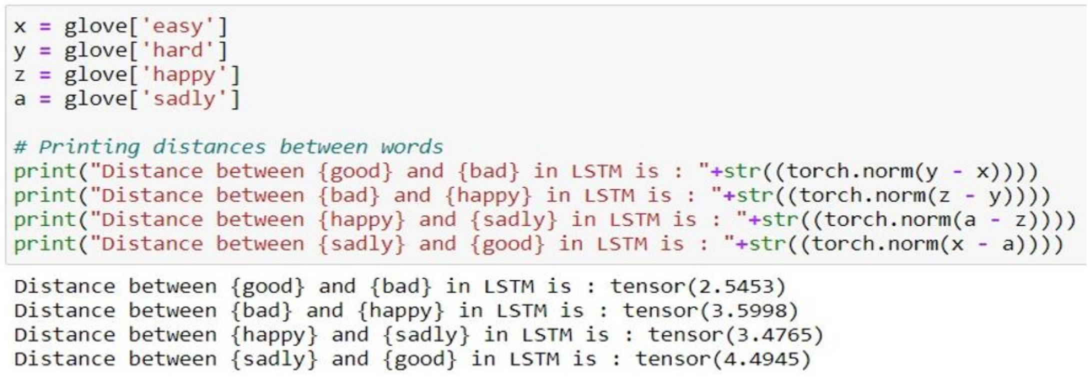
4.3.8.

To conclude this experiment a pairwise distance of words from the vocabulary in the embedding space of LSTM and Glove model. An example of this is given below.



```
weights = model_LSTM_T.get_weights()[0]

# Taking the embeddings from all words
x = weights[sequence_dict['easy']-1]
y = weights[sequence_dict['hard']-1]
z = weights[sequence_dict['happy']-1]
a = weights[sequence_dict['sadly']-1]

# Printing distances between words
print("Distance between {good} and {bad} in LSTM is : "+str(distance.euclidean(x, y)))
print("Distance between {bad} and {happy} in LSTM is : "+str(distance.euclidean(y, z)))
print("Distance between {happy} and {sadly} in LSTM is : "+str(distance.euclidean(z, a)))
print("Distance between {sadly} and {good} in LSTM is : "+str(distance.euclidean(x, a)))
```

```
Distance between {good} and {bad} in LSTM is : 3.1519455909729004
Distance between {bad} and {happy} in LSTM is : 1.1890476942062378
Distance between {happy} and {sadly} in LSTM is : 1.2307151556015015
Distance between {sadly} and {good} in LSTM is : 3.1261558532714844
```



```
x = glove['easy']
y = glove['hard']
z = glove['happy']
a = glove['sadly']

# Printing distances between words
print("Distance between {good} and {bad} in LSTM is : "+str((torch.norm(y - x))))
print("Distance between {bad} and {happy} in LSTM is : "+str((torch.norm(z - y))))
print("Distance between {happy} and {sadly} in LSTM is : "+str((torch.norm(a - z))))
print("Distance between {sadly} and {good} in LSTM is : "+str((torch.norm(x - a))))
```

```
Distance between {good} and {bad} in LSTM is : tensor(2.5453)
Distance between {bad} and {happy} in LSTM is : tensor(3.5998)
Distance between {happy} and {sadly} in LSTM is : tensor(3.4765)
Distance between {sadly} and {good} in LSTM is : tensor(4.4945)
```

Here as we can see the distance between bad, sadly and happy are coming similar from LSTM with flattened layer and Glove model, so we can say that our embeddings are coming perfectly form the reviews with somewhat similar accuracy to the Glove models.

4.3.9 Inference

From this experiment we can infer the following:

- We have received maximum and similar accuracies for LSTM Approach and GRU Approach as they have the Logic Gates Logic Implemented with 90%.
- LSTM has three gates (input, output and forget gate) and it needs large parameters, takes more memory, and takes longer time to execute. This is Lossless and highly dependent.
- GRU has two gates (reset and update gate), and it takes lesser parameters, lesser training time with consuming less memory with faster execution times.
- This is Losly and non-highly dependent. Here we got very high accuracies using this model with low compared to LSTM and GRU as here we are not using any gates.
- In LSTM and GRU when we used to flatten layer the accuracies boosted from 90% to 92%, as we have used flattened later.
- In Dense Layer as we do not have memory to remember the previous nuances to the later sentences, so we have received least test accuracies of 73%.

4.3.10 Conclusion on Experiments

In the first methodology we when trained and tested we could get similar accuracies for Simple LSTM and LSTM with Embedding

In the second methodology we when trained our approaches we could see that both LSTM and GRU with flattened layer performed better.

By all the approaches mixed accuracy and loses we could get to know that the dense layer is having higher losses in larger epochs as it does not remember the semantic relations between the reviews. On the other hand, the LSTM, LSTM with flattened layer, GRU and GRU with flattened layer performed much better with higher accuracy scores as they have forgot and remember essential information managed through the gates present in them. Thus, as per the experiments LSTM with flattened layer performed much better than all the models evaluated in the experiments.

4.4 Implementation Part 4

Speech recognition is the ability of a computer to identify and respond to the sounds produced in human speech. In this model we use the web scraped data obtained from implementation part-1 and develop a quality-of-life improvement by implementing speech recognition to search for the appropriate product.

4.4.1 Workflow

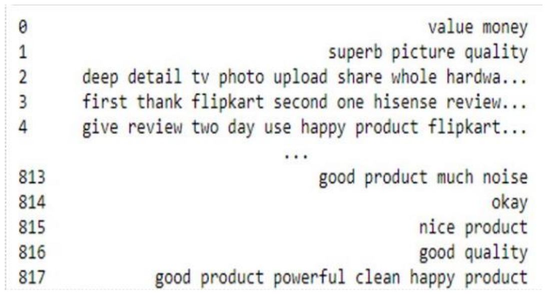
The workflow for this part of the experiment is :

- Step 1: Reading Data and Data Preprocessing
- Step 2: Training Amazon Reviews Data using Decision Tree
- Step 3: Testing the model on scraped Flipkart Data
- Step 4: Getting input data through speech
- Step 5: Calculating top 3 and Bottom 3 products
- Step 6: Extracting Entities of top 3 products reviews

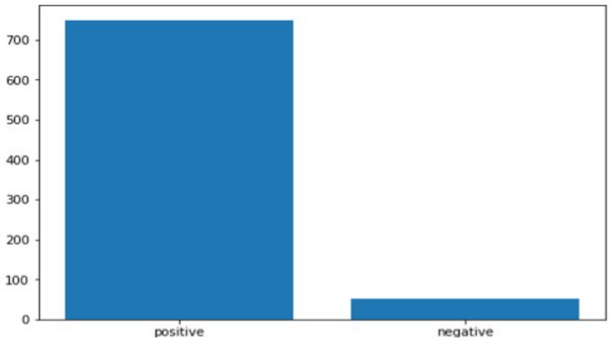
Step 1: Reading Data and Data Preprocessing

We read the data obtained from the web scraping done in part -1 next we preprocess the data by taking only words using RE (Regular Expression) ,Remove spaces and remove stop words.

Next Apply lemmatization and store the preprocessed data .

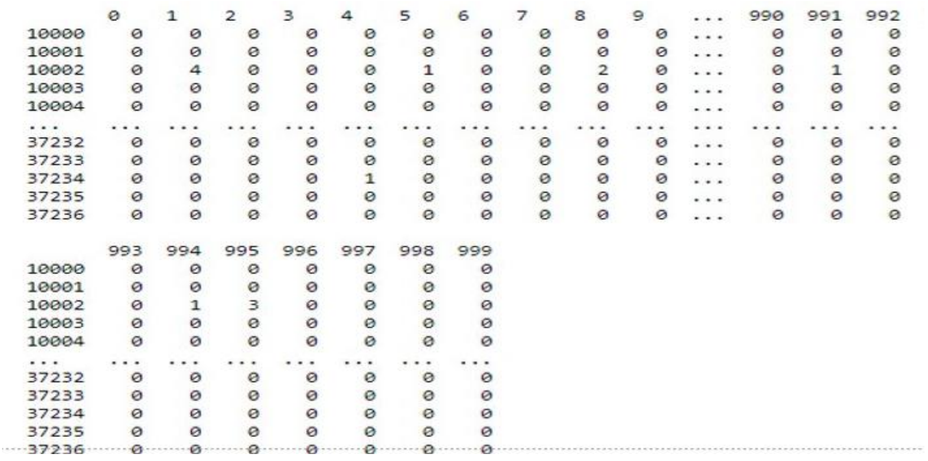


```
0                                           value money
1                                 superb picture quality
2      deep detail tv photo upload share whole hardwa...
3      first thank flipkart second one hisense review...
4      give review two day use happy product flipkart...
                              ...
813                              good product much noise
814                                                 okay
815                                         nice product
816                                         good quality
817             good product powerful clean happy product
```



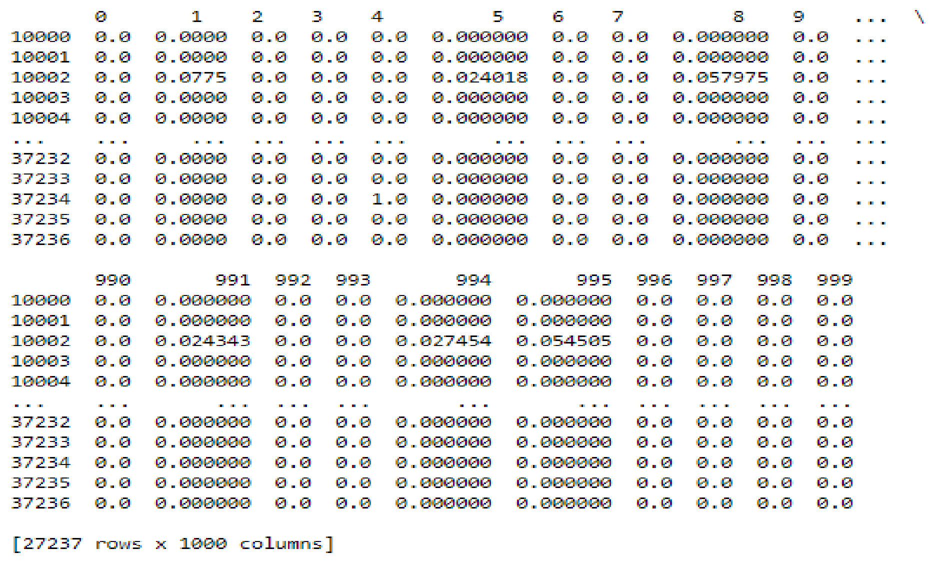
Step 2: Training Amazon Reviews Data using Decision Tree

Converting Data into Count Vectorize form which returns integer values and Tf-idf form which returns float values



```
        0    1    2    3    4    5    6    7    8    9   ...  990  991  992
10000   0    0    0    0    0    0    0    0    0    0   ...    0    0    0
10001   0    0    0    0    0    0    0    0    0    0   ...    0    0    0
10002   0    4    0    0    0    1    0    0    2    0   ...    0    1    0
10003   0    0    0    0    0    0    0    0    0    0   ...    0    0    0
10004   0    0    0    0    0    0    0    0    0    0   ...    0    0    0
...   ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
37232   0    0    0    0    0    0    0    0    0    0   ...    0    0    0
37233   0    0    0    0    0    0    0    0    0    0   ...    0    0    0
37234   0    0    0    0    1    0    0    0    0    0   ...    0    0    0
37235   0    0    0    0    0    0    0    0    0    0   ...    0    0    0
37236   0    0    0    0    0    0    0    0    0    0   ...    0    0    0

      993  994  995  996  997  998  999
10000   0    0    0    0    0    0    0
10001   0    0    0    0    0    0    0
10002   0    1    3    0    0    0    0
10003   0    0    0    0    0    0    0
10004   0    0    0    0    0    0    0
...   ...  ...  ...  ...  ...  ...  ...
37232   0    0    0    0    0    0    0
37233   0    0    0    0    0    0    0
37234   0    0    0    0    0    0    0
37235   0    0    0    0    0    0    0
37236   0    0    0    0    0    0    0
```

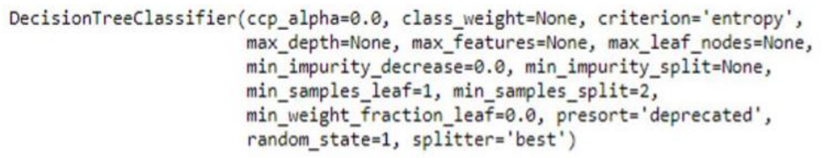
Count Vectorized form



```
        0       1    2    3    4         5    6    7         8    9  ...  \
10000 0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.0  0.000000  0.0  ...
10001 0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.0  0.000000  0.0  ...
10002 0.0  0.0775  0.0  0.0  0.0  0.024018  0.0  0.0  0.057975  0.0  ...
10003 0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.0  0.000000  0.0  ...
10004 0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.0  0.000000  0.0  ...
...   ...     ...  ...  ...  ...       ...  ...  ...       ...  ...  ...
37232 0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.0  0.000000  0.0  ...
37233 0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.0  0.000000  0.0  ...
37234 0.0  0.0000  0.0  0.0  1.0  0.000000  0.0  0.0  0.000000  0.0  ...
37235 0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.0  0.000000  0.0  ...
37236 0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.0  0.000000  0.0  ...

      990       991  992  993       994       995  996  997  998  999
10000 0.0  0.000000  0.0  0.0  0.000000  0.000000  0.0  0.0  0.0  0.0
10001 0.0  0.000000  0.0  0.0  0.000000  0.000000  0.0  0.0  0.0  0.0
10002 0.0  0.024343  0.0  0.0  0.027454  0.054505  0.0  0.0  0.0  0.0
10003 0.0  0.000000  0.0  0.0  0.000000  0.000000  0.0  0.0  0.0  0.0
10004 0.0  0.000000  0.0  0.0  0.000000  0.000000  0.0  0.0  0.0  0.0
...   ...       ...  ...  ...       ...       ...  ...  ...  ...  ...
37232 0.0  0.000000  0.0  0.0  0.000000  0.000000  0.0  0.0  0.0  0.0
37233 0.0  0.000000  0.0  0.0  0.000000  0.000000  0.0  0.0  0.0  0.0
37234 0.0  0.000000  0.0  0.0  0.000000  0.000000  0.0  0.0  0.0  0.0
37235 0.0  0.000000  0.0  0.0  0.000000  0.000000  0.0  0.0  0.0  0.0
37236 0.0  0.000000  0.0  0.0  0.000000  0.000000  0.0  0.0  0.0  0.0

[27237 rows x 1000 columns]
```

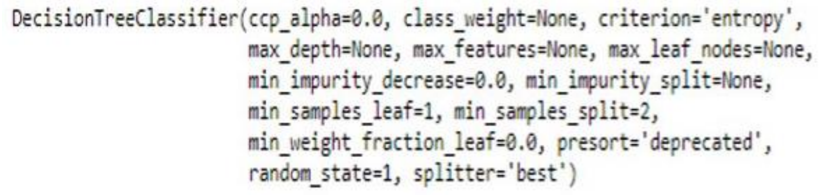
tf-idf form

Next begin training with a count vectorizer.



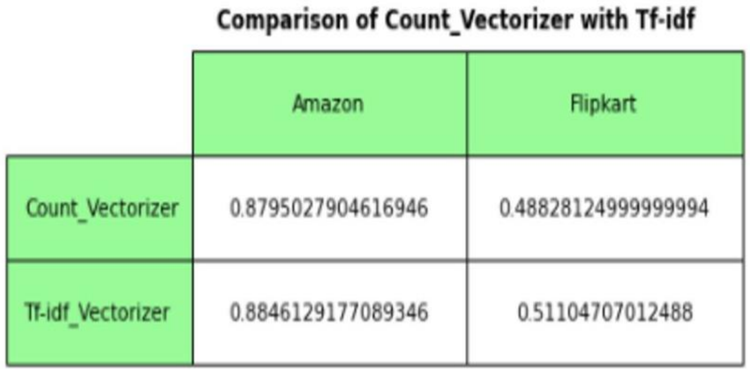
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=1, splitter='best')
```

And following that we train with a tf-idf vectorizer.



```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=1, splitter='best')
```

Step 3: Testing on scraped Flipkart Data

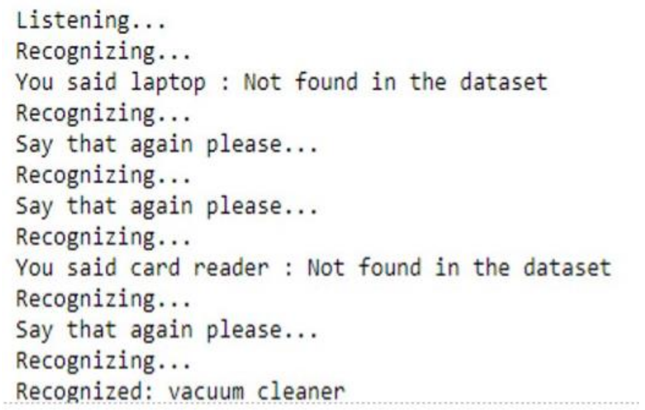


Comparison of Count_Vectorizer with Tf-idf

	Amazon	Flipkart
Count_Vectorizer	0.8795027904616946	0.48828124999999994
Tf-idf_Vectorizer	0.8846129177089346	0.51104707012488

Step 4: Getting input data through speech

The model takes an input from the microphone this input is passed to the neural network and is converted into plain text(Speech Recognition library). A sample output of the conversion model is given below.



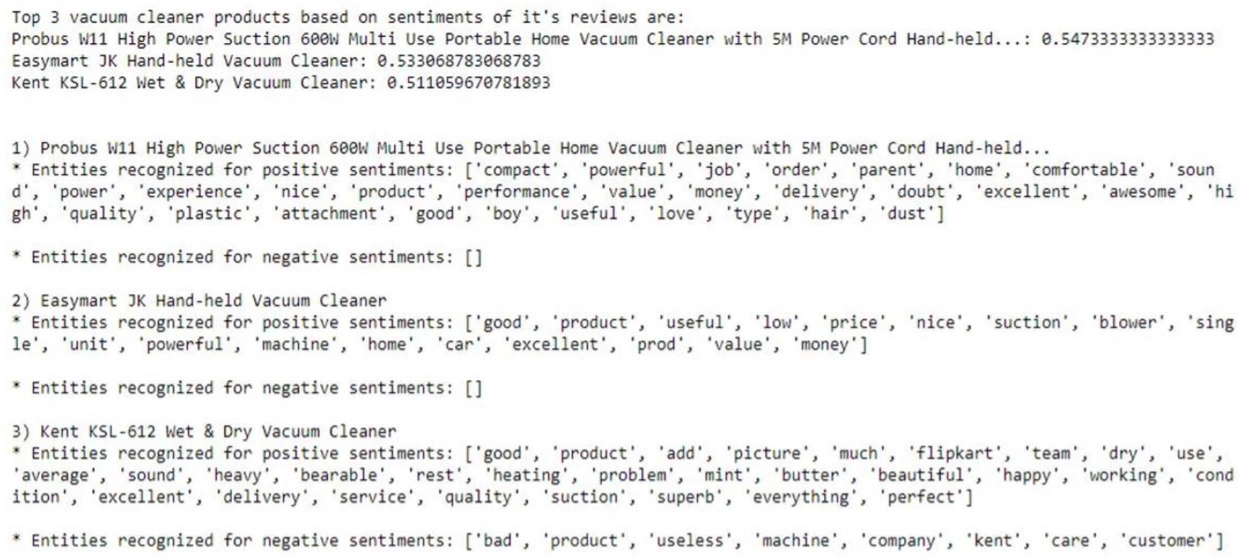
```
Listening...
Recognizing...
You said laptop : Not found in the dataset
Recognizing...
Say that again please...
Recognizing...
Say that again please...
Recognizing...
You said card reader : Not found in the dataset
Recognizing...
Say that again please...
Recognizing...
Recognized: vacuum cleaner
```

Step 5: Calculating Top 3 and Bottom 3 products based on sentiment polarity

Taking a data frame of only input items a dictionary with every product as key and all reviews associated with that product as values in the list. This list is then sorted by average polarity (Text Blob) of all items in decreasing order

4.4.2 Top 3 products

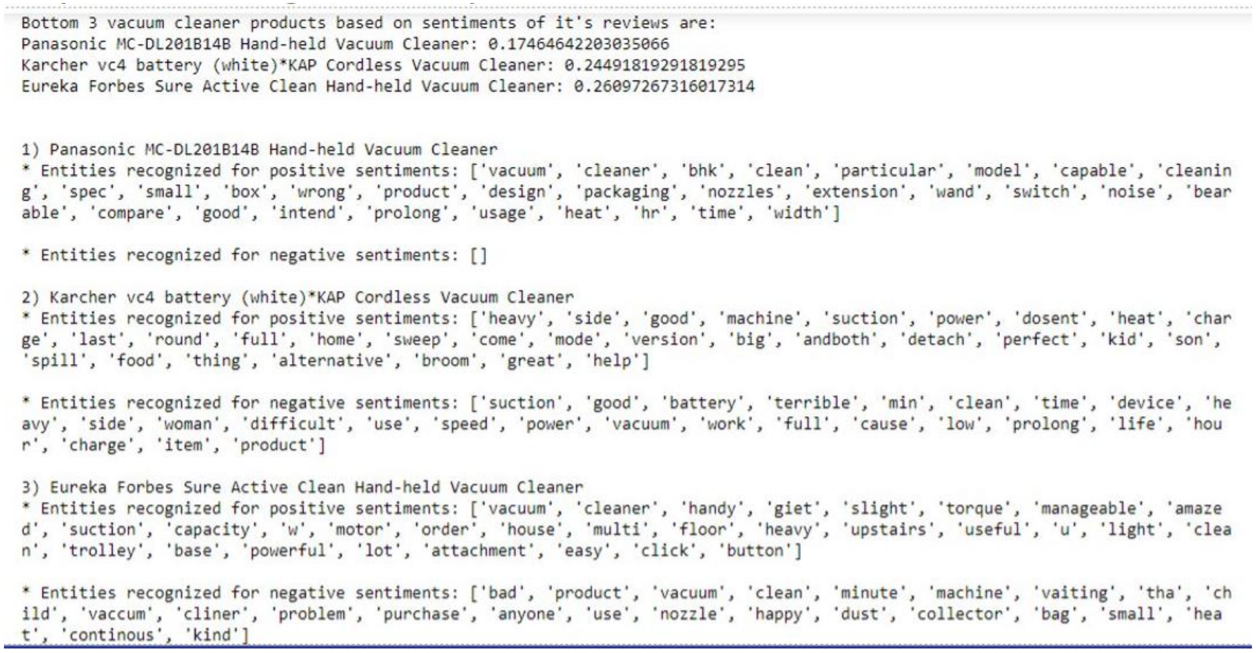
And the final output of the top 3 products with the reasoning for their purchase is also listed below.



```
Top 3 vacuum cleaner products based on sentiments of it's reviews are:
Probus W11 High Power Suction 600W Multi Use Portable Home Vacuum Cleaner with 5M Power Cord Hand-held...: 0.5473333333333333
Easymart JK Hand-held Vacuum Cleaner: 0.533068783068783
Kent KSL-612 Wet & Dry Vacuum Cleaner: 0.511059670781893


1) Probus W11 High Power Suction 600W Multi Use Portable Home Vacuum Cleaner with 5M Power Cord Hand-held...
* Entities recognized for positive sentiments: ['compact', 'powerful', 'job', 'order', 'parent', 'home', 'comfortable', 'soun
d', 'power', 'experience', 'nice', 'product', 'performance', 'value', 'money', 'delivery', 'doubt', 'excellent', 'awesome', 'hi
gh', 'quality', 'plastic', 'attachment', 'good', 'boy', 'useful', 'love', 'type', 'hair', 'dust']

* Entities recognized for negative sentiments: []

2) Easymart JK Hand-held Vacuum Cleaner
* Entities recognized for positive sentiments: ['good', 'product', 'useful', 'low', 'price', 'nice', 'suction', 'blower', 'sing
le', 'unit', 'powerful', 'machine', 'home', 'car', 'excellent', 'prod', 'value', 'money']

* Entities recognized for negative sentiments: []

3) Kent KSL-612 Wet & Dry Vacuum Cleaner
* Entities recognized for positive sentiments: ['good', 'product', 'add', 'picture', 'much', 'flipkart', 'team', 'dry', 'use',
'average', 'sound', 'heavy', 'bearable', 'rest', 'heating', 'problem', 'mint', 'butter', 'beautiful', 'happy', 'working', 'cond
ition', 'excellent', 'delivery', 'service', 'quality', 'suction', 'superb', 'everything', 'perfect']

* Entities recognized for negative sentiments: ['bad', 'product', 'useless', 'machine', 'company', 'kent', 'care', 'customer']
```

4.4.3 Bottom 3 products

Conversely the bottom 3 products are also listed with the reasons as to why they are bad and under purchased.



```
Bottom 3 vacuum cleaner products based on sentiments of it's reviews are:
Panasonic MC-DL201B14B Hand-held Vacuum Cleaner: 0.17464642203035066
Karcher vc4 battery (white)*KAP Cordless Vacuum Cleaner: 0.24491819291819295
Eureka Forbes Sure Active Clean Hand-held Vacuum Cleaner: 0.26097267316017314


1) Panasonic MC-DL201B14B Hand-held Vacuum Cleaner
* Entities recognized for positive sentiments: ['vacuum', 'cleaner', 'bhk', 'clean', 'particular', 'model', 'capable', 'cleanin
g', 'spec', 'small', 'box', 'wrong', 'product', 'design', 'packaging', 'nozzles', 'extension', 'wand', 'switch', 'noise', 'bear
able', 'compare', 'good', 'intend', 'prolong', 'usage', 'heat', 'hr', 'time', 'width']

* Entities recognized for negative sentiments: []

2) Karcher vc4 battery (white)*KAP Cordless Vacuum Cleaner
* Entities recognized for positive sentiments: ['heavy', 'side', 'good', 'machine', 'suction', 'power', 'dosent', 'heat', 'char
ge', 'last', 'round', 'full', 'home', 'sweep', 'come', 'mode', 'version', 'big', 'andboth', 'detach', 'perfect', 'kid', 'son',
'spill', 'food', 'thing', 'alternative', 'broom', 'great', 'help']

* Entities recognized for negative sentiments: ['suction', 'good', 'battery', 'terrible', 'min', 'clean', 'time', 'device', 'he
avy', 'side', 'woman', 'difficult', 'use', 'speed', 'power', 'vacuum', 'work', 'full', 'cause', 'low', 'prolong', 'life', 'hou
r', 'charge', 'item', 'product']

3) Eureka Forbes Sure Active Clean Hand-held Vacuum Cleaner
* Entities recognized for positive sentiments: ['vacuum', 'cleaner', 'handy', 'giet', 'slight', 'torque', 'manageable', 'amaze
d', 'suction', 'capacity', 'w', 'motor', 'order', 'house', 'multi', 'floor', 'heavy', 'upstairs', 'useful', 'u', 'light', 'clea
n', 'trolley', 'base', 'powerful', 'lot', 'attachment', 'easy', 'click', 'button']

* Entities recognized for negative sentiments: ['bad', 'product', 'vacuum', 'clean', 'minute', 'machine', 'vaiting', 'tha', 'ch
ild', 'vaccum', 'cliner', 'problem', 'purchase', 'anyone', 'use', 'nozzle', 'happy', 'dust', 'collector', 'bag', 'small', 'hea
t', 'continous', 'kind']
```

4.4.4 Inference

After performing all these models, we have found that the model which outperformed is LSTM model with flatten layer. Moreover, the results that we got from experimenting with calculating polarity and extracting selected POS-tags is promising for top-3 and bottom-3 products.

5. CONCLUSION

We experimented with various models for sentiment analysis on product reviews dataset. We found that with Vader sentiment, sentiment concentration towards positivity decreases as we move from top to lower brands. And as we increase complexity of supervised models, the performance of the model is also increased. Compared to bag of words models, LSTMs and GRUs perform way better at recognizing sentiments because they take context into consideration. LSTMs and GRUs perform well with flatten layer than dense layer.

Finally, these models are used on the dataset to get the polarity of all the reviews and calculate top 3 products in any category which is determined by given input speech(speech recognition). We also provided the entities for the top 3 product reviews that are responsible for classifying positive and negative sentiments.

6. REFERENCES

[1] Yang Liu Jian , Wu Bia Zhi , Ping Fan. Ranking products through online reviews: A method based on sentiment analysis technique and intuitionistic fuzzy set theory , July 2017 . Pages 149-161.

[2] Mukesh Chapagain. Python NLTK: Sentiment Analysis on Movie Reviews [Natural Language Processing (NLP)], April 2018.

[3] E. Suganya ,S. Vijayarani . Sentiment Analysis for Scraping of Product Reviews from Multiple Web Pages Using Machine Learning Algorithms. 2018 ,Pages 677-685.

[4] Dishi Jain, Bitra Harsha Vardhan, Saravanakumar Kandasamy . Sentiment Analysis of Product Reviews – A Survey. December 2019.

[5] C. Hutto , Eric Gilbert. VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text .May 2015.

[6] Susanne Wagner , Intralingual speech-to-text-conversion in real-time: Challenges and Opportunities 2013.

[7] Susan Li , Named Entity Recognition with NLTK and SpaCy . Aug 2017.