ANALYSIS OF AMAZON PRODUCT REVIEWS

Venkata Jayandra Kumar Lade   Sindhuja Josyula   Sundarganesh Govindakumar   Vamsi Krishna Reddicherla   Vamsi Varma Kunaparaju

Abstract

The need to discover a product-review's helpfulness without any user's manual input, has a significant demand. That remains to be our aim. We planned to classify the product reviews posted on Amazon, the world's most successful internet retailer and tech giant. Using machine learning algorithms, we decide if a review is helpful or not for a potential buyer. Our classifiers were trained on datasets with different sizes. The predictors used for this classification, included some additional features which were extracted, from the ones already existing in the dataset. Our experimentations included feeding(inputting) different subset of features for different machine learning algorithms. Using these features, and some pre-selected advance machine learning algorithms, we have managed to achieve an accuracy of 83.1%.

I. Introduction

With a rapid pace in the developments of technology and internet, online retailing has come down to be a common man's approach, to fulfill a lot of pleasures and necessities from the means of his house. Amazon is one of the largest key player in this industry. The reviews that the users leave behind, have a great potential in helping to increase the product quality and, also provide manufacture feedback. But, the quality of these reviews may vary and, as a solution, we developed a machine learning model, to find the helpful reviews.

Once a buyer buys an item, Amazon prompts him/her to compose a review of the item and, also to rate it, on a scale of 1-5. Reading through all these thousands of reviews before buying a product, is practically impossible and thus, the problem of review helpfulness raises. Amazon currently follows a maneuver to solve this. Their solution requires a user to manually classify the review to be either helpful or not. This comes with its very own set of complications. These complications include the fact that a user cannot go through all the reviews, to rank their helpfulness. This leaves the reviews that were written a little earlier to fall back in the list irrespective of their usefulness. Also, a review which hasn't been read during the first few minutes of existence, would never to be seen again if it were to be ranked according to the helpfulness rate. To make an attempt in solving this issue, we propose a machine learning algorithm that would automatically classify the review to be helpful or not, using the text in the review and, certain other features which can be extracted from this text. This would help us to display the most helpful reviews, either positive or negative, on the first page of the product.

II. Dataset

The dataset we chose consists of product reviews and, product metadata from Amazon. It had about 142.8 million reviews from May 1996 to July 2014. It also includes reviews-ratings and helpfulness votes. The metadata includes the product description, category information, price etc. This dataset was extracted from University of California-San Diego and was granted approval by Julian McAuley. http://jmcauley.ucsd.edu/data/amazon/links.html

III. Data Preprocessing

The initial dataset we were accessed to, was in a JSON format. Our very first step was to convert the. json file format to .csv. The category of reviews we chose to work on, "electronics" had about 1.4 million rows. Some cleaning has been to the dataset before generating any additional features. The reviews which had null values in any of the attributes were removed. For the reasons of achieving better accuracy, we decided to train our models with reviews which had at least 10 ratings. We have calculated the helpfulness percentage based on these ratings. So, if a review had less than 10 ratings, it

wouldn't yield as much information. Also, we have taken in to account, the products which had at least 10 reviews. This was due to the assumption we made that, product with very less number of reviews, would not represent our cause.

A. *Anatomical Features*

From the dataset, we have generated a feature named helpfulness rating which is the ratio of the number of helpful votes, to the total number of votes. Once this value is calculated, we labeled a review as helpful if this helpfulness rating is more than 60%. In addition to this, we have also added some new features which were extracted using the review text in the dataset. The features added are listed below: [1]

- Length of review
- Character count
- Sentence count
- Number of Exclamation marks
- Number of Question marks
- Number of Words in capital.

We have anticipated that adding these features would help us in analyzing the reviews based on tokens and syntactic forms. Length of review, character count and sentence count were added with an insight that, the higher the counts of these features the more the information they carry about the product. In addition to this, we added number of exclamation marks, number of question marks and number of capital words as a measure of emotion. We assumed that higher the number of capital words or exclamation marks, higher is the emotion of the customer in the review. With the higher number of question marks in a review, it was observed that there were more questions regarding the product and the higher the number of questions, the less useful it can be.

B. *Lexical Features*

In addition to the above features, some lexical features were also added. The idea behind adding these features is to capture the complexity of a given review i.e. if it is easily understandable to the user or not. The features added were: [1]

1) Flesch Reading Ease Score
2) Automated Readability Index

Flesch Reading Ease Score: It represents the ease of understandability for each review. It is leveled on a scale of 1-100. The lesser the score is, the higher the difficulty is to understand the review and vice versa.

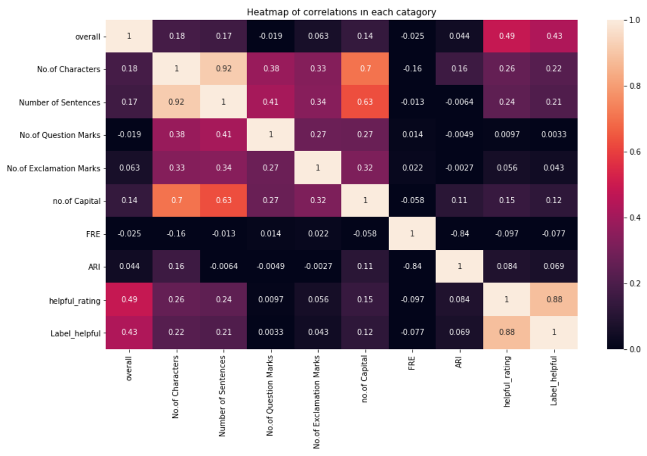
$$206.835 - 1.015\left(\frac{totalwords}{totalsentences}\right) - 84.6\ \left(\frac{totalsyllables}{totalwords}\right)$$

Automated Readability Index: It is a standard index that represents an approximate grade level in US required to understand the text. So, if the ARI = 7, it represents that one needs to have at least 7 years of education in US to understand the review. It is calculated using the following formula:

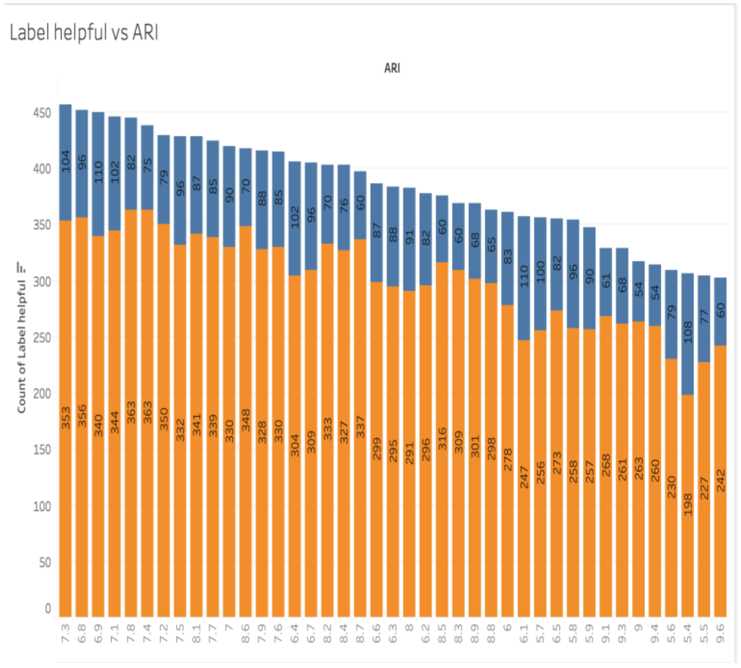
$$4.71\left(\frac{characters}{words}\right)+0.5\left(\frac{words}{sentences}\right)-21.43$$

IV. Visualizations

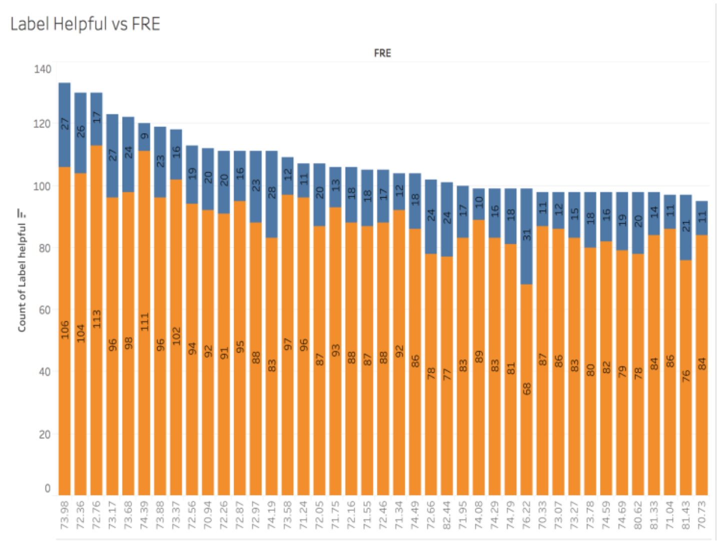
Prior to building of model, the dataset was explored using visual tools. This was done to get better understanding of the data and how the features are correlated to each other.



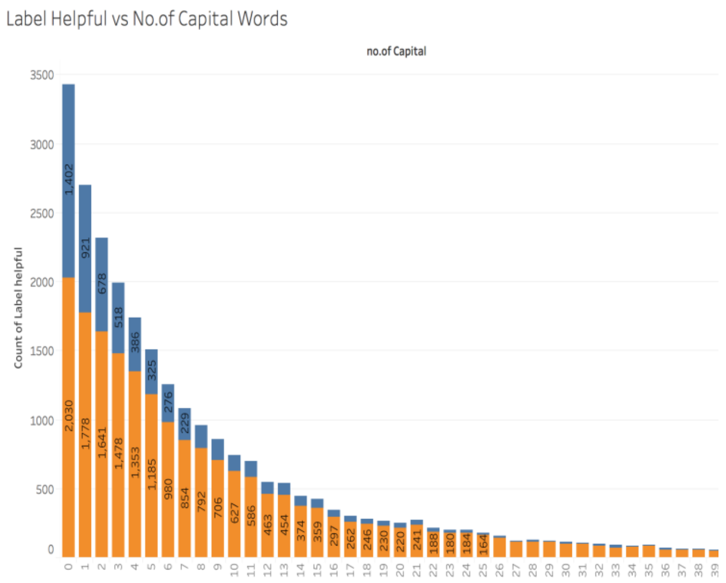
From the above heat map, we can understand how the numerical predictors depend on the response variables. Helpful rating has more correlation with Label Helpful because the Label helpful was generated based on helpful rating due to which we didn't consider helpful rating for building models. We can also observe that no other variables create multi-collinearity problem.



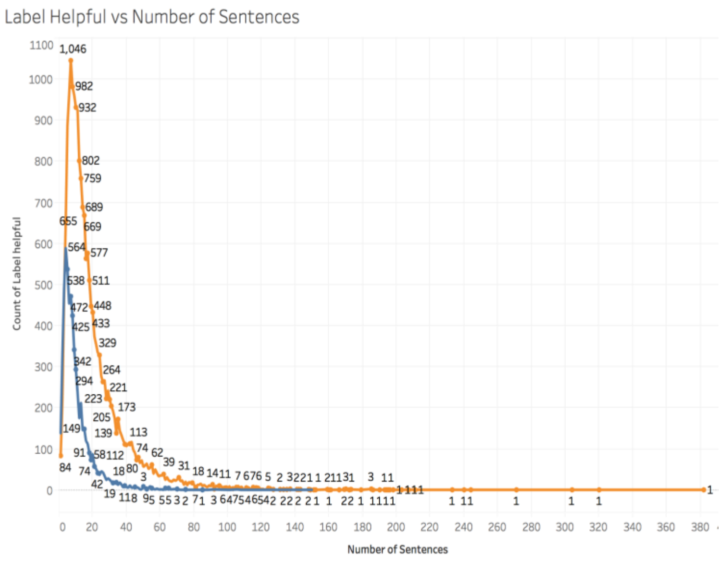
From the above graph, there is no much correlation between Label Helpful and ARI since there is no trend.



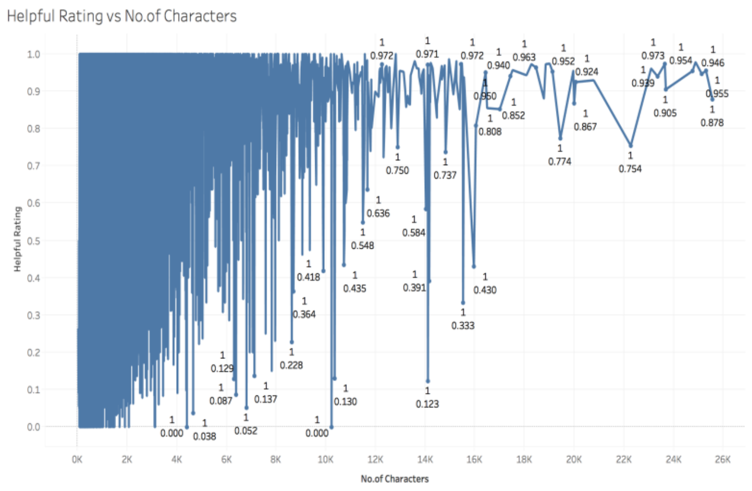
From the above graph, there is no much correlation between Label Helpful and ARI since there is no trend.



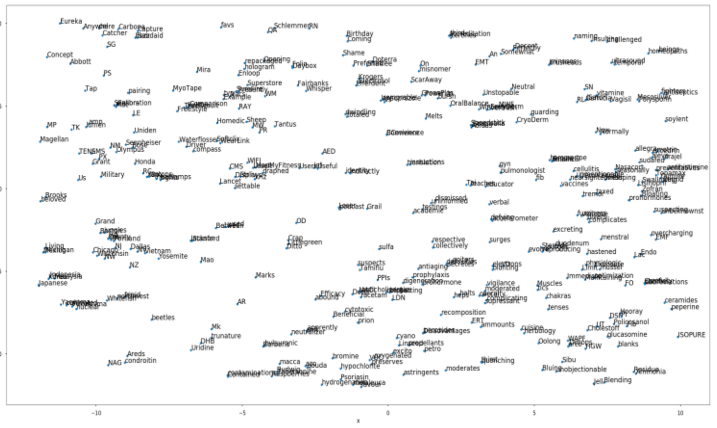
From the above graph, as the number of capital words are increasing in a review, the not-helpful reviews are decreasing with respect to the helpful reviews. There may be a chance that if the number of capital words are given to model then it may perform well.



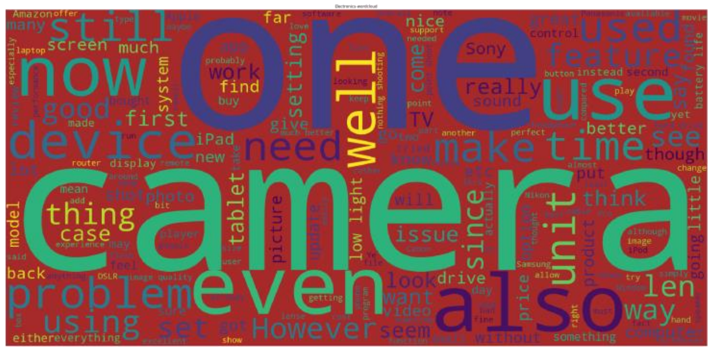
From the above graph, as the number of sentences increase in a review, the not-helpful reviews are decreasing and there is a point from which there are only helpful reviews present.



From the above graph, it can be seen that in the first bin on the X-axis which is completely dark, it gives an indication that at a particular number of characters there are reviews which are helpful and not helpful also. As the number of characters increase, the helpful ratings fluctuate. It can be said that the number of characters may not be a good feature to train the model.



The above graph is t-Distributed Stochastic Neighbor Embedding graph. The reviews were converted into word2vec and plotted. This made it possible to see how close the words are related to each other.



The above is a word cloud graph generated using Electronics dataset. This word cloud shows us the words which have high frequency in the dataset. The words ‘Camera’ and ‘One’ are the words which occur in high frequency in the dataset.

V. Prior to Machine Learning

Different NLTK modules were used and they are:

1) Tokenizing
2) Stemming
3) Stop word removal
4) N-grams

Tokenizing:

In this process we split the sentences into words which is very important while creating Term frequency for the words.

Stemming:

By this process, we remove the suffixes from the word, which will allow our algorithm to find exact trends in meaning of the sentence and it will also reduce the un necessary features that is being created.

Stop Word Removal:

By implementing this process, it acts as a trade off as it will act as a hinder for the learning algorithm as it is very difficult to understand the meaning of the sentence without the stop words.

N-grams:

It makes group words to n long and they are usually computationally expensive, and we proceeded with N grams of 2.

After the above modules were implemented, there were still some unwanted words in the dataset which had to be removed. For this purpose, regular expressions were used to further remove the symbols like “],},),*” etc.

After performing the above steps, we got bag of words by which we can know the frequency of words. By generating Tf-Idf (term frequency-inverse document frequency) vectors, we were able to get the weight of stop words and other common words that were appearing frequently in the reviews. We calculated Tf-Idf using the following formula:

$$tfidf = tf * idf$$

Where *tf* is the term frequency and:

$$tf(w,r) = 0.5 + 0.5\frac{f(w,r)}{\max_{t\in r}(f(t,r))}$$

And the for idf:

$$idf(w,R) = \log\frac{N}{1+|r\in R : w\in r|}$$

- N: the total number of reviews in the training data
- r: a specified review in the training collection of reviews R
- w: the specified word in review r that we are calculating the tf-idf statistic for
- $tf(w,r) = f(w,r)$ denotes the the raw frequency of word w in review r (the number of times w occurs in r)
- $\max_{t\in r}(f(t,r))$: the maximum raw frequency of any word t in the review r
- $1+|r\in R : w\in r|$: the number of reviews in the training set R that contain the word w, adjusted by 1 to avoid any possible division-by-zero error

For training models, different subset of features was used, and they were selected by using grid search. And dataset is divided into training and test sets in some cases and for some K-fold cross validation was performed. For our project we set minimum document frequency of .001 by this we can able to avoid errors in spelling of words and it also feature dimensionality which leads the model to work in minimum time.

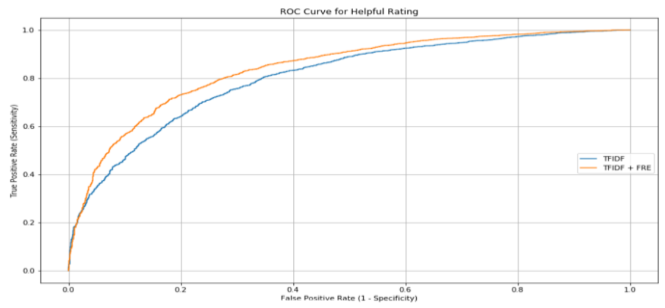
VI. Technical Models

1. *Logistic Regression:*

It is a linear model of classification rather than the regression, we took this model into consideration as the model can learn after the initial training, even though they are not as fast as the naïve Bayes.

Initially, model was trained with TF-IDF vectors matrices. We achieved an AUC score of 0.8031.

To improve the model performance, we added another feature “FRE” as they have co-relation score of 0.029 with the feature column, by using the pickle function and we again split the data into training and testing set. We used grid search technique to validate the model using the validation set using combination of the different parameters in the grid and they are mainly used to optimize number of parameters which is needed to insert into training set. Over the Grid search we used K cross validation with K =5 to avoid the bias over the optimized model.

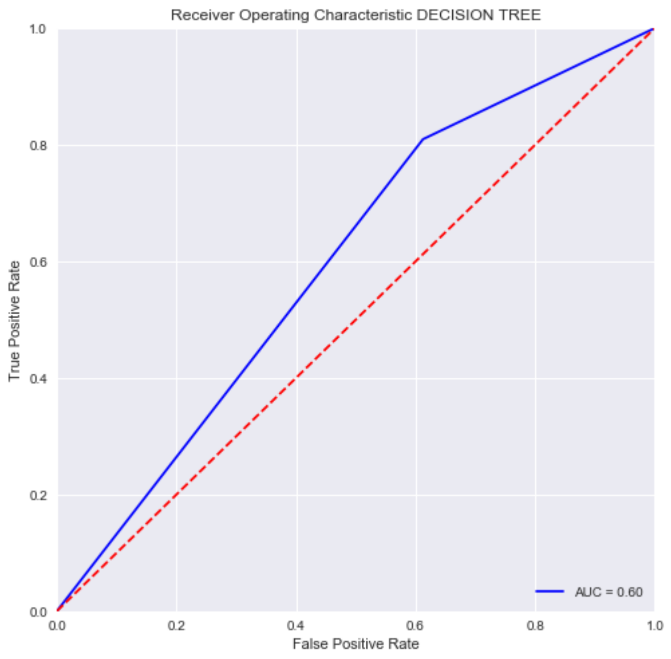


The average ROC_AUC score for 100 different random states is equal to 0.7795, the same as our optimum solution. In this regard, our solution can be considered robust.

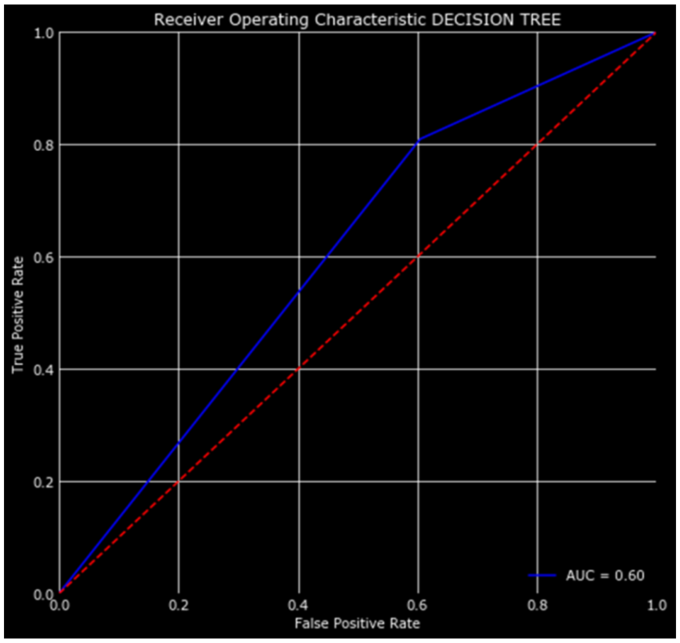
For the evaluation we took 100 random samples and the ROC score was around 0.8310 and we achieved an increase in performance of optimized model by 2.79 % and the area under the ROC curve can be considered as accuracy of the model.

2. *Decision Trees*

It is a flowchart which is presented as a tree structure where the internal nodes represent test on an attribute and the branch represents the outcome of the test. Leaf nodes represent class labels and distribution.



The above plot is a ROC (receiver operating characteristic) curve of decision trees which is trained with Tf-Idf vectors. The accuracy obtained for this curve is 70.69%.

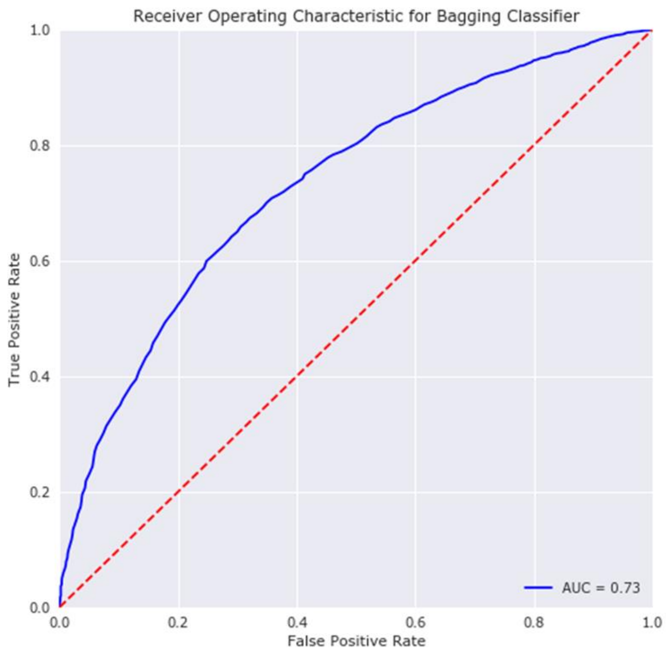


The above plot is a ROC (receiver operating characteristic) curve of decision trees which is trained with Tf-Idf vectors, FRE, ARI, No.of characters, question marks and exclamation marks. The accuracy obtained for this curve is 71.32%.

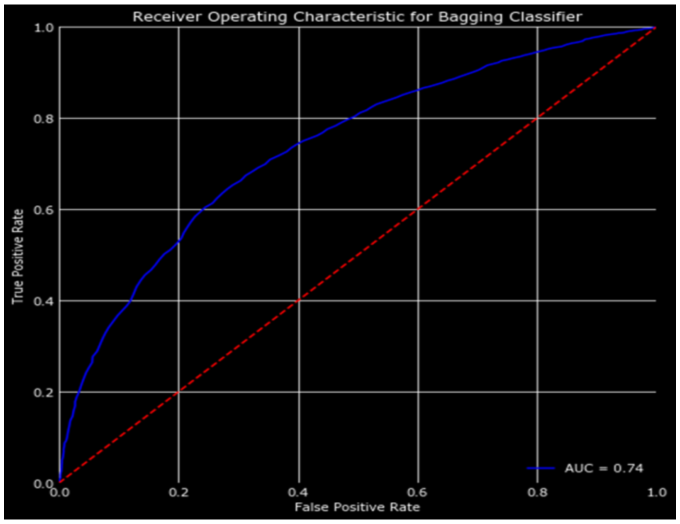
Even though the accuracy has increased, the ROC values didn't change. So, the model trained with additional features is the best. This may be due to lazy classification.

3. ***Bagging:***

As mentioned above, decision trees do not perform well due to lazy classification. Bagging is a model which works based on reducing variances and training with many datasets.



The above plot is a ROC (receiver operating characteristic) curve of bagging which is trained with Tf-Idf vectors. The accuracy obtained for this curve is 76.13%.



The above plot is a ROC (receiver operating characteristic) curve of bagging which is trained with Tf-Idf vectors, FRE, ARI, No.of characters, question marks and exclamation marks. The accuracy obtained for this curve is 76.35%.

So, by doing bagging the accuracy has increased when compared to decision trees and by training the model with added features, the accuracy was similar.

4. *Random Forest:*

It is a enassembling algorithm that fits 'n' number of decision trees over the subsamples of the data and they use mean to improve the prediction accuracy and it counts labels among each subtree to synthesis accurate labels and they are taken into consideration as they work well with high dimensional data as out data and they also have minimum training cost.

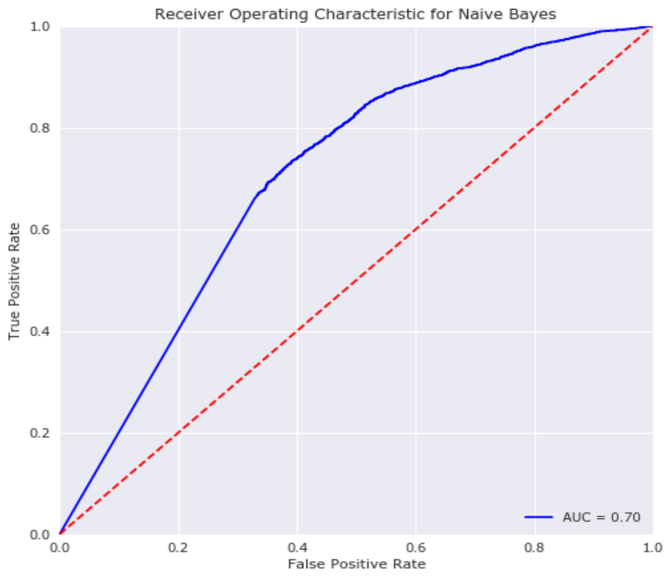
Initially, model was trained with TF-IDF vectors matrices. We achieved an AUC score of 0.7187.

5. *Adaboost:*

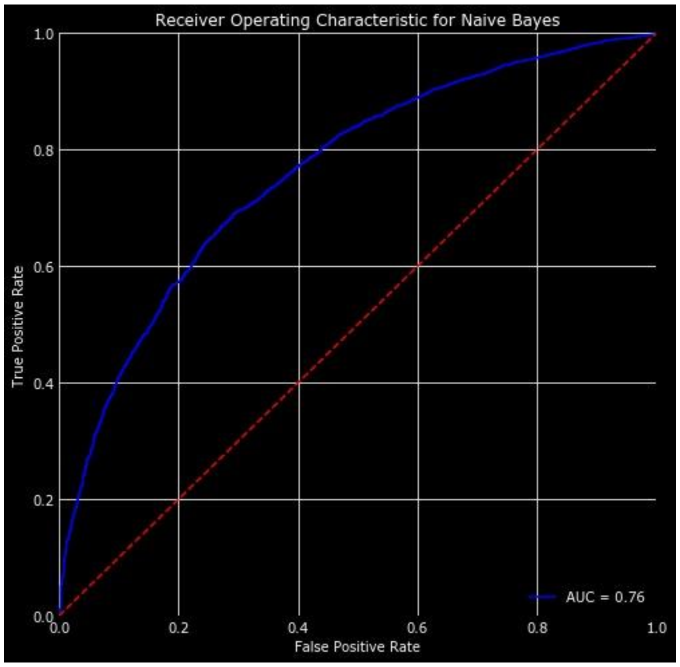
It starts by fitting weak classifier over the dataset and it adds classifiers over the data and they change weights during error circumstances and due to it boosting function we will not have over fitting of the data.

Initially, model was trained with TF-IDF vectors matrices. We achieved an AUC score of 0.7646.

6. *Naïve Bayes:*



The above plot is a ROC (receiver operating characteristic) curve of naïve Bayes which is trained with Tf-Idf vectors. The accuracy obtained for this curve is 69.87%.



The above plot is a ROC (receiver operating characteristic) curve of naïve Bayes which is trained with Tf-Idf vectors, FRE, ARI, No.of characters, question marks and exclamation marks. The accuracy obtained for this curve is 77.92%.

The trained model which has added features has the highest accuracy and there is a significance increase in it. Generally, naïve bayes works for small sample sizes so while training the model, it is recommended to divide the training data set into small sets.

7. ***Multilayer Perceptron:***

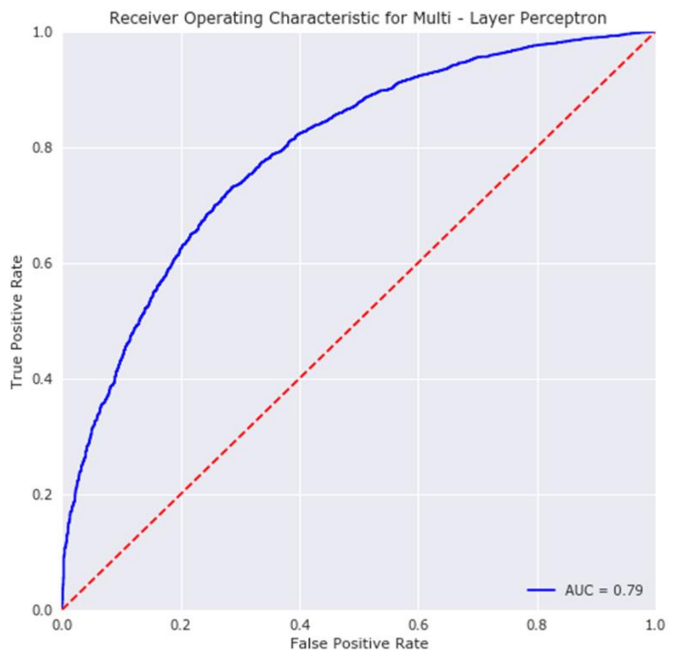
Multilayer Perceptron generally build using the cost function which is shown below:

$$h_\Theta(x) \in \mathbb{R}^K \quad (h_\Theta(x))_i = i^{th} \text{ output}$$

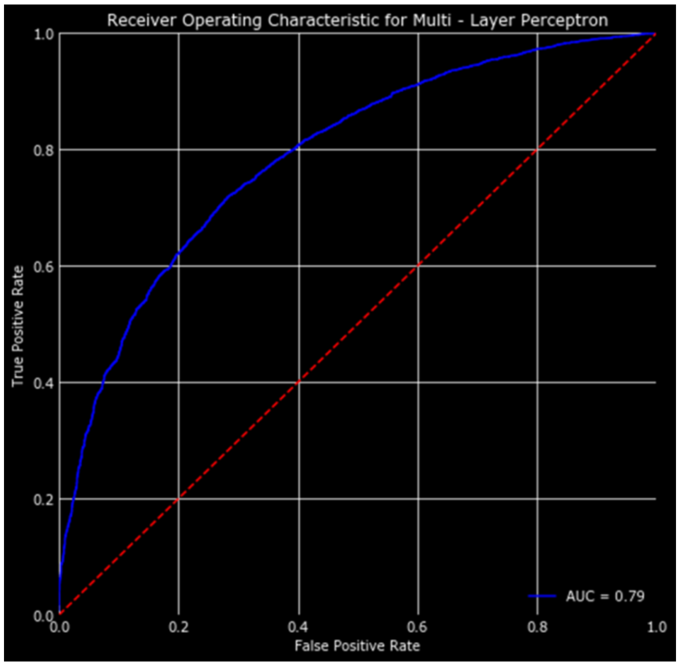
$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1-y_k^{(i)})\log(1-(h_\Theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

For better result, generally the weights should be changed. For changing the weights back propagation algorithm is used. Back propagation uses inverse derivatives to find best weights. And it finds using the formula below:

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1-y_k^{(i)})\log(1-(h_\Theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$



The above plot is a ROC (receiver operating characteristic) curve of multilayer perceptron which is trained with Tf-Idf vectors. The accuracy obtained for this curve is 79%.



The above plot is a ROC (receiver operating characteristic) curve of multilayer perceptron which is trained with Tf-Idf vectors, FRE, ARI, No.of characters, question marks and exclamation marks. The accuracy obtained for this curve is 79%.

In this, the accuracies remain the same even though new features were added. Generally, this works for any type of classifications or regressions.

8. ***Support Vector Machine:***

Support vector machines are supervised learning which are used for classification or regression analysis. SVMs basically trains using support vectors. SVM's works on two different types of data which are linear or non-linear. Since, we don't know how our data looks we tried using different SVMs.

8.1 ***Linear SVM:***

The linear SVM model will be build using the formula mentioned below:

$$\min_{\theta} C \sum_{i=1}^{m} \left[y^{(i)} cost_1(\theta^T x^{(i)}) + (1-y^{(i)}) cost_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$

Here SVM is trained using two different sets of features. First the model trained with only bag of words which has accuracy of 80.64% and other one is trained with Bag of words, FRE, ARI, No.of characters, question marks and exclamation marks has accuracy of 75.44%

8.2 *Radial SVM:*

The radial SVM model will be build using the formula mentioned below:

$$\min_{\theta} C \sum_{i=1}^{m} y^{(i)} cost_1(\theta^T f^{(i)}) + (1-y^{(i)}) cost_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

These type is used when the data is non-linear.

Here SVM is trained using two different sets of features. First the model trained with only Bag of words which has accuracy of 76.13% and other one is trained with Bag of words, FRE, ARI, No.of characters, question marks and exclamation marks has accuracy of 76.35%

So, by comparing both accuracies we can say that our data works well for linear.

9. *Passive Aggressive Classifier:*

This classifier works on the algorithm shown below:

Algorithm 2 Passive-Aggressive Active learning algorithms (**PAA**)

```
INPUT : penalty parameter $C > 0$ and smoothing parameter $\delta \geq 1$.
INITIALIZATION : $\mathbf{w}_1 = (0, \dots, 0)^\top$.
for $t = 1, \dots, T$ do
  observe: $\mathbf{x}_t \in \mathbb{R}^d$, set $p_t = \mathbf{w}_t \cdot \mathbf{x}_t$, and predict $\hat{y}_t = \mathrm{sign}(p_t)$;
  draw a Bernoulli random variable $Z_t \in \{0, 1\}$ of parameter $\delta/(\delta + |p_t|)$;
  if $Z_t = 1$ then
    query label $y_t \in \{-1, +1\}$, and suffer loss $\ell_t(\mathbf{w}_t) = \max(0, 1 - y_t \mathbf{w}_t \cdot \mathbf{x}_t)$;
    compute $\tau_t$ according to Eq. (1), and $\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t$;
  else
    $\mathbf{w}_{t+1} = \mathbf{w}_t$;
  end if
end for
```

Here the classifier is trained with two different set of features. One is trained with only bag of words which has accuracy of 74.51% and the other one is trained with bag of words, FRE, ARI, No.of characters, question marks and exclamation marks has accuracy of 75.37%. So, we can say that second model is best for this classification.

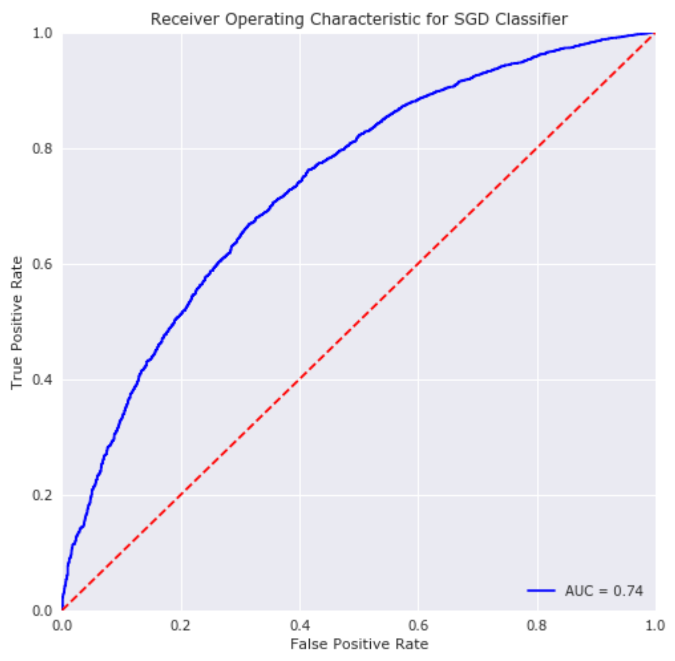
10. *Stochastic Gradient Classifier:*

This classifier works on the updation of theta of the objective J(theta) as,

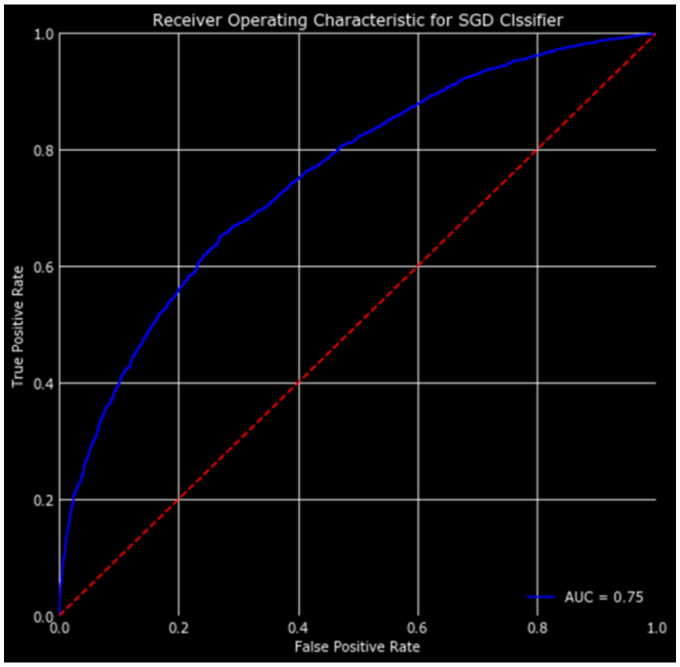
$$\theta = \theta - \alpha \nabla_\theta E[J(\theta)]$$

SGD generally computes gradient using few training examples. So, the updates given by

$$\theta = \theta - \alpha \nabla_\theta J(\theta; x^{(i)}, y^{(i)})$$



The above plot is a ROC (receiver operating characteristic) curve of SGD which is trained with Tf-Idf vectors. The accuracy obtained for this curve is 76.05%.



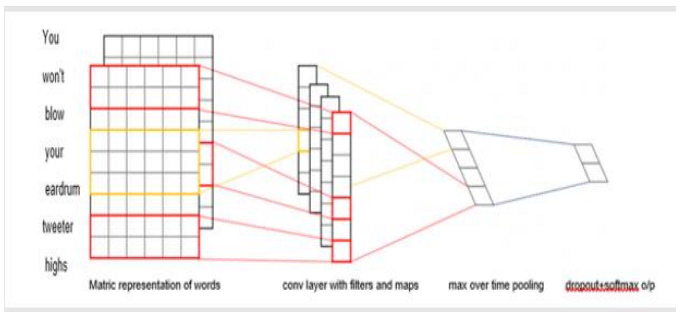
The above plot is a ROC (receiver operating characteristic) curve of SGD which is trained with Tf-Idf vectors, FRE, ARI, No.of characters, question marks and exclamation marks. The accuracy obtained for this curve is 74.77%.

Here by adding the features accuracy has been down. So, the model trained with TF-IDF only is the best

.

11.Convolution Neural Network:

Convolution neural network was implemented using tensor flow. For this implementation we divided the dataset into two classes namely helpful and not helpful based on the classification labels and we did the same preprocessing and NLTK techniques which were done for the ML algorithm implementation and we used 10 cross validation technique such that 10% of the data is allocated for the testing purpose. The below image describes architecture used in project:



Before ingesting the review sentences into the embedding layer of the CNN, we padded each of the review sentence to maximum length of 64 to avoid memory leak and each of the sentences should be of the same length for the efficient batch processing of the data and we used GENSIM package to form word2vec where each of the words is mapped towards the corresponding vector.

Embedding Layer:

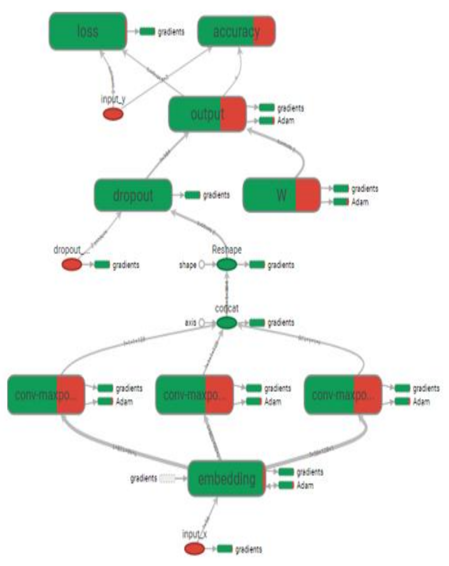
In this layer the vocabulary in the review sentences were converted into low dimensional vectors such that they act as a look up table we used the function called Tensor flow name scope for the embedding visualization in the tensor board and embedding lookup function will lead to the formation of 3d tensors, but as the CNN expects 4d and as we use only static word we declared none for the static dimension.

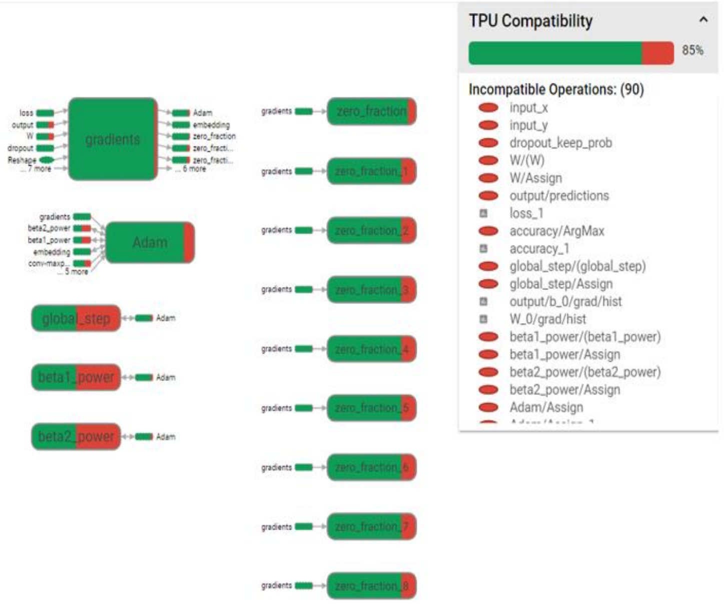
Convolutional and Max Pooling Layer:

Each convolution produces tensor of different shape, so it is necessary to form neuron for each tensor and finally merge them together to form feature vector, such that they will be fed towards the Maxpool layer where they are multiplied with filter matrix which will again result in a tensor, all the tensors from the Maxpool layer were again combined to form feature vector

Drop Out Layer:

Regularization of the neural network tend to take place in this layer such that they disable certain neuron to avoid interdependency among neuron which will allow each neuron to learn individually about features. For predicting the accuracy of the model, the output is made to undergo matrix multiplication process.





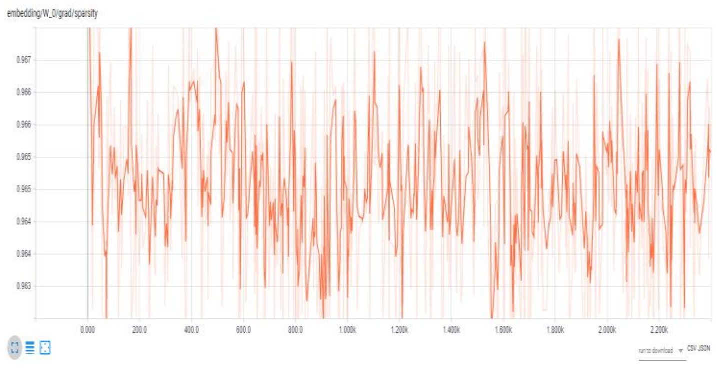
TPU Compatibility

85%

Incompatible Operations: (90)

- input_x
- input_y
- dropout_keep_prob
- W/(W)
- W/Assign
- output/predictions
- loss_1
- accuracy/ArgMax
- accuracy_1
- global_step/(global_step)
- global_step/Assign
- output/b_0/grad/hist
- W_0/grad/hist
- beta1_power/(beta1_power)
- beta1_power/Assign
- beta2_power/(beta2_power)
- beta2_power/Assign
- Adam/Assign

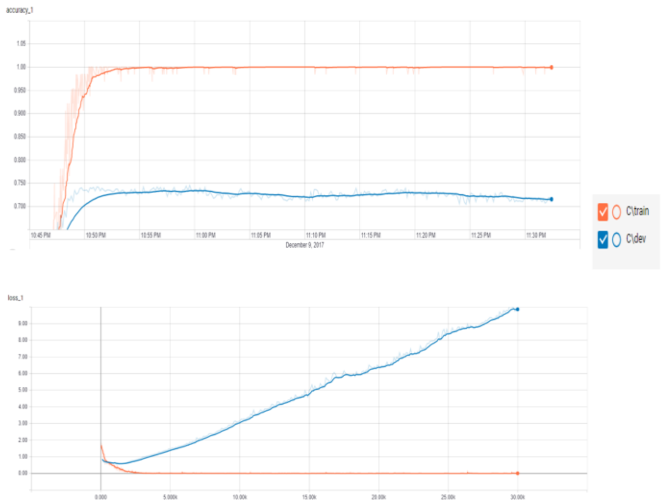
Using the tensor board, we visualized every layer in our network and we can able to visualize in which layer loss has been occurred.



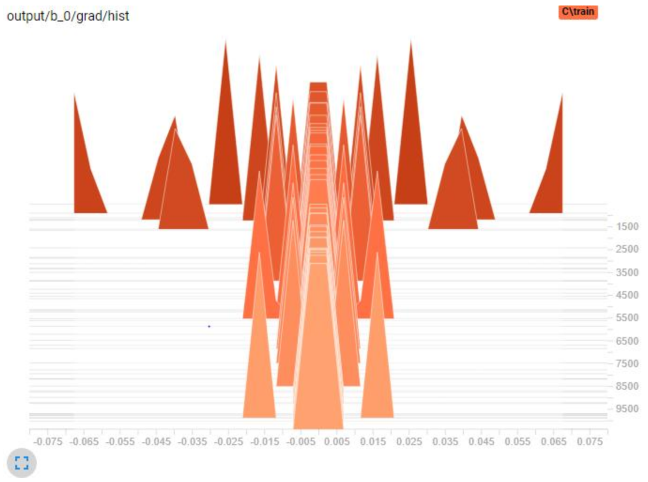
The above graph denoted the embedding layer activity during the training phase.

Accuracy and Loss estimation of CNN :

Accuracy of CNN :	0.7147



Above Figure denotes that our CNN achieved an accuracy of 0.7147 and we can conclude that training metrices were not smooth as we used batch of minimum size and dev which test accuracy is way below than that of the train phase suggesting that we need more data for the process and they may be low due to drop out function which was applied in the last layer.



The above figure displays the weight updates that took place over the layers and each epoch.

VII. Conclusion

From the models generated, we can promise that our models can classify the reviews successfully with a comfortable accuracy. Naïve Bayes is one of the best models for this type of classifications particularly when small sample sizes are involved. It has also achieved good accuracy. Linear SVM model also achieved good accuracy but it was unrealizable because it takes a lot of time to train and is difficult to find the correct parameters. Logistic model which was trained with TFIDF and FRE achieved highest accuracy among all other models. We conclude that logistic model is the best model for our dataset.

VIII. Future Scope

- Introducing more features such as Entropy and giving them as input to the models.
- Acquiring and working on even larger datasets with more reviews per product might lead to better classifying models.
- More complicated models such as LSTM, RNN and other deep learning techniques might yield better results.

- As we know that CNN is the state of art in the field of deep learning, we can improve our model by increasing dimensions of the input embedding and using the pre-trained one and the accuracy can be improved by adding regularization function in the last layer and by increasing the dropout rate.

References:

[1] R. He, J. McAuley, 2016, Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering, http://jmcauley.ucsd.edu/data/amazon/links.html

[2] Predicting helpfulness rating for Amazon Product Reviews , http://cs229.stanford.edu/proj2014/Jordan%20Rodak,%20Minna%20Xiao,%20Steven%20Longoria,%20Predicting%20Helpfulness%20Ratings%20of%20Amazon%20Product%20Reviews.pdf

[3] HUI BWU Y. Anti-spam model based on semi-Naive Bayesian classification model. Journal of Computer Applications. 2009;29(3):903-904.

[4] Liaw A. Weiner M. Classification and Regression by randomForest. R News. 2002;Vol 2(2):18-22.

[5] Casanova R, Saldana S, Chew EY, Danis RP, Greven CM, et al. (2014) Application of Random Forests Methods to Diabetic Retinopathy Classification Analyses. PLoS ONE 9(6): e98587. doi: 10.1371/journal.pone.0098587

[6] Jones M. Viola A. Robust Real-Time Face Detection. International Journal of Computer Vision. 2004. pg 137–154.

[7] Edwin Chenn, 2011, Choosing a Machine Learning Classifier, http://blog.echen.me/2011/04/27/choosing-a-machine-learning-classifier/

[8] Ye Zhang, Byron Wallace, 2015 revised in 2016, A Sensitivity analysis of CNN for Sentence Classification, https://arxiv.org/abs/1510.03820

[9] T Lanigan, Amazon Review-Machine Learning Project, https://t-lanigan.github.io/amazon-review-classifier/

[10] Shitij Bhargava, Predicting Amazon Review Helpfulness, https://cseweb.ucsd.edu/~jmcauley/cse255/reports/wi15/Shitij_Bhargava.pdf

[11] Jing Lu, Peilin Zhao, Steven C.H. Hoi, 2016, Online Passive- Aggressive Active Learning, https://link.springer.com/article/10.1007/s10994-016-5555-y

[12] Analytics Mantra, March 2017, Word Cloud from Text Data. https://www.youtube.com/watch?v=d_zt5XjWVn4&t=358s

GIT-HUB Code Reference:

The entire project was uploaded in GitHub. The below is the link: https://github.com/jayandrakumar/Analysis-of-amazon-product-reviews