Sentimental Analysis for Amazon Product Review Dataset

Zeba J. Vakil, Ganesh Rajasekar, and Atish Telang Patil, *Group29-ECE657A*

***Abstract*— Sentimental Analysis is the process of computationally identifying and categorizing statements made from a piece of text and determine whether the author's feelings toward a particular product is positive, negative or neutral. It is very vital for Industrial leaders like Amazon with their giant E-Commerce business to understand consumer behaviour and their sentiment to reach out better to them. With recent developments in the field of machine learning and text analytics this could be effectively achieved. In this project we aim to predict the sentiment or polarity of a review given to a particular amazon baby product using supervised machine learning algorithms. Based on that we will compare the performance of each of the approaches used and comment on the best approach for such applications.**

***Index Terms*— Classification, Feature extraction, LSTM, Sentiment analysis, Word embedding.**

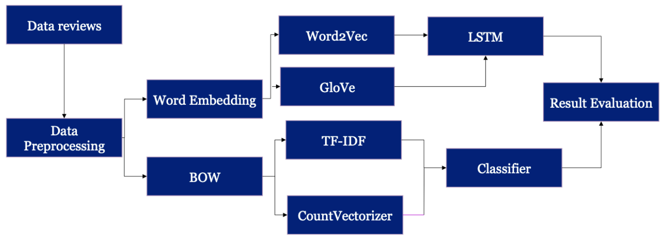


Fig. 1: Project Overflow.

I. Introduction

In today's world where large number of data is created everyday, it becomes very important for giant E-Commerce websites to gain insight of this data. The businesses of these websites are largely dependant on how well they understand their customers. The inclination of the customers towards buying a product is highly dependant on the reviews. So it is a very crucial task for the Industrial leaders like Amazon to know their product reviews which showcase the consumer's sentiment. This is where Sentimental analysis becomes useful. Using sentimental analysis we are categorizing the text mainly into positive and negative sentiments [7]. For this project we are using Amazon Review Baby Product Review Dataset. The dataset consists of 1,60,000 entries out of which we are using the review and the rating attribute from our dataset. We perform a supervised learning approach to first train the models with our two classes and train the model based on the ratings and then predict the ratings based on the reviews provided. The Figure 1 shows the process workflow of our project. Firstly we are using the data and pre-processing it. We are splitting our approach into Word Embedding approach and Bag-of-Words approach. In the Bag-of-Words approach we are implementing the Count Vectorizer and the TF-IDF methods. And in the word embedding we have two models considered mainly Word2Vec and the glOve model. We have used LSTM for the Word Embeddings and for the Bag-of-Words approach we are considering different classifiers. Finally we compare and evaluate the results.

II. Literature Review

As part of the Literature review and previous research work in the area of sentimental analysis we went through the Pang et al. [8] which was the earliest attempt to classify the document with sentiment instead of topics. Though they had mentioned that the traditional machine learning methods like Naive Bayes, Maximum Entropy Classification, and Support Vector Machines do not perform as well on sentiment classification as on traditional topic-based categorization.

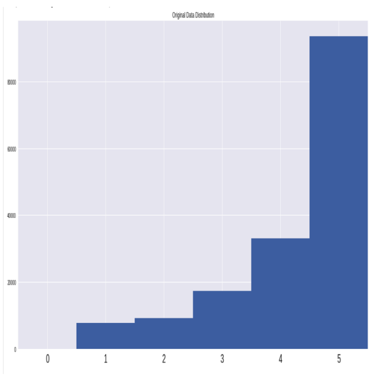
But we found some noteworthy works which contradicted those claims. In the paper [9] the author has tried to apply existing supervised learning algorithms to the Yelp dataset such as Naive Bayes, Perceptron and Multiclass SVM and compared their predictions with the actual ratings. They successfully concluded that the binarized Naive Bayes combined with feature selection with stop words removed and stemming is the best in terms of sentiment analysis of such datasets. It is to be noted that the yelp dataset features closely resemble the amazon product review dataset that we are trying to address in this project. Also we found the paper [3] that tackles the fundamental problem of sentimental analysis i.e. sentiment polarity categorization. The paper considers both the review-level and sentence-level categorization. In the study, data pre-processing is done by extracting all the subjective contents i.e. all the sentences which has at least one positive or negative word. Based on the POS tagging and the negative prefixes, negative phrases are then identified. After which the sentiment scores are computed. For training the classifiers, each training data entry is transformed to feature vector that has binary strings to represent tokens of words in the sentence. Finally for estimation, 10-fold cross validation is applied where the sentiment score is used to identify the positive and negative classes. The sentiment score proves to be a strong feature achieving 0.73 F1 score for review-level categorization and 0.8 for sentence-level categorization. However, the paper addresses two limitations where it cannot perform well i.e. when F1 scores are very low and when there are implicit sentiments. For the categorization, the classification models used in the paper are: Naive Bayes, Decision Trees and Support Vector Machine. In addition to them, in [5] they have adopted a supervised learning approach to polarize the unlabeled dataset. They have used active learning to label the data. The data pre-processing is done by tokenization, removing stop words and POS tagging. They have used a mix of two kinds of approach for feature extraction. They finally measured the classification performance based on Precision, Recall, F-measure and Accuracy parameters. Conclusively, it is observed that according to the paper SVM performs the best when large number of datasets are available.

Also one noteworthy work which we reviewed was [1] in which the author shows the significance of word order in sentiment classification by effectively comparing bag-of-words approach with an LSTM approach which can handle sequential data as the LSTM remembers the words from the beginning and chain of events is not lost in an LSTM model. Their results showed that Word Embedding layer with a LSTM approach performed the best. So reviewing all these works we wanted to validate their claim by picking some of the approaches and apply that to our dataset and see if we are getting good results for our dataset.

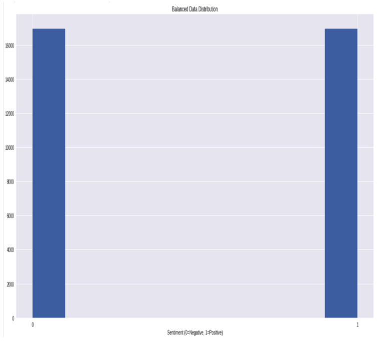
III. Dataset Analysis

A. Data Acquisition

For this project we are using Amazon Review Baby Product Review Dataset given at http://jmcauley.ucsd.edu/data/amazon/. The dataset consists of 1,60,000 entries out of which we are using the review and the ratings attribute from our dataset. We acquired our dataset which was in JSON format. We labelled our dataset based on the ratings. We labelled all the reviews which had ratings greater than 3 as 1 representing positive reviews and less than 3 as 0 representing negative reviews. We are removing all the reviews having rating equal to 3 from our dataset considering them as neutral.



(a) Original Data Distribution



(b) Balanced Data Distribution

Fig. 2: Data Distribution

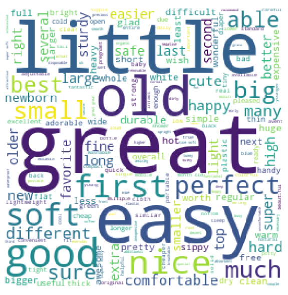
B. Data Pre-processing

From Figure 2 (a) it can be seen that our original data distribution was highly imbalanced. The reviews which had rating equal to 5 were in large number compared to other reviews such has reviews having rating 1 and 2. This shows that there were more number of positive reviews than the negative reviews. In order to analyse how the models behave in different data distributions we carry out two approaches one for original data distribution and other for balanced data distribution. For balancing the data we randomly sampled our data using RandomUnderSampler of the imblearn library. After which our data appeared to be as shown in the figure 2 (b).

For our data cleaning we are performing the following tasks:

- *Tokenization:*
  It separates our reviews into individual words known as tokens which is then used as an input for our parsing process. We are using NLTK python package to tokenize our reviews.
- *Removing Stop Words:*
  Stop words are those words which are unnecessary for our text mining process and can serve to reduce our accuracy. So we are removing them from our corpus. The NLTK package of Python has a dictionary of stop words so using that we are removing stop words from our corpus. We did not remove not and no words as we thought removing them could change the context of a sentence as ”Not Good” will become ”Good”.
- *Removing Hyper Links:*
  We noticed that our reviews contained hyperlinks also. These are again not needed for our resultant feature set. So we removed them using Beautiful Soup module.
- *Removing Punctuations:*
  Punctuation are also those objects which are not necessary for our analysis. We are removing punctuations from our corpus with the help of regular expressions.
- *Lemmitization:*
  When provided with words as input, lemmitization returns the lemma i.e. the base word. By this we are ensuring to build a meaningful corpus for our analysis.

C. Feature Extraction


(a) Features from the positive reviews

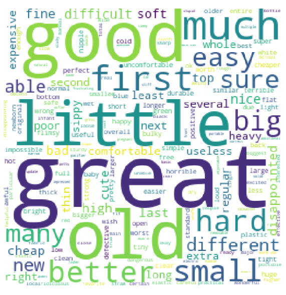

(b) Features from the negative reviews

Fig. 3: Review World Cloud

We are generating our feature set considering two techniques, Bag-of-words and the Word Embeddings.

1) Bag-of-words

The Bag-of-words approach is the way of extracting features and getting the word occurrences from the text to use in the algorithms. It concentrates on the word and its occurrences. Since we have large number of reviews, considering all of them as features is computationally expensive. Because of this, we are extracting the top 2000 most used words from our dataset to create our bag of words. For the Bag-of-words approach we are considering two methods mainly Count Vectorizer and TF-IDF. When building the vocabulary for both the methods we are taking minimum document frequency as 5 which means that words which have not appeared in at least 5 of the reviews are not considered. Also we are taking one to two number of words in a sequence i.e. n-grams = 1,2.

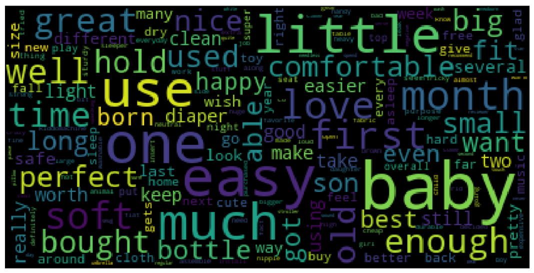


Fig. 4: Bag of Words Features

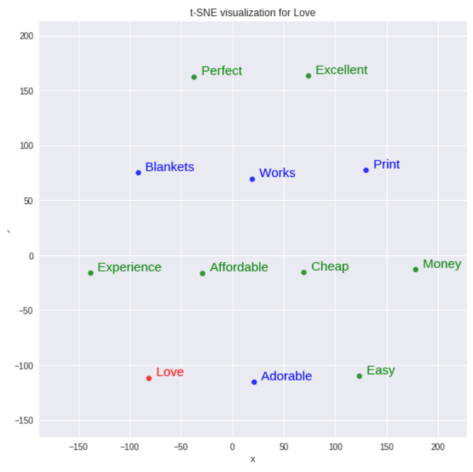
- *Count Vectorizer:* It converts a collection of text documents to a matrix of token counts. This implementation builds a sparse representation of the token occurrences.
- *TF-IDF:* Know as the term frequency-inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. If a positive word occurs in a negative review multiple times or vice versa, the weight of such words are reduced in the TF-IDF representation.

2) *Word Embedding:*

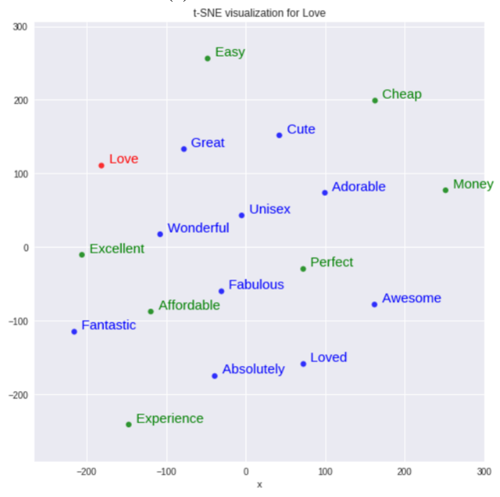
The Word Embedding captures context of a word in a document, semantic and syntactic similarity, relation with other words, etc.. They are a type of word representation that allows words with similar meaning to have a similar representation[Example: cosine similarity].For our Word Embedding we are using two models:

- *Word2Vec*: Word2vec is a two-layer neural network that processes the text. It takes the text corpus as input and output the feature vectors for all the words in that corpus. The purpose and usefulness of Word2vec is to group the vectors of similar words together in vector space. That is, it detects similarities mathematically. Word2vec creates vectors that are distributed numerical representations of word features, features such as the context of individual words https://skymind.ai/wiki/word2vec
- *glOve*: GloVe is an unsupervised learning algorithm from stanford 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 https://nlp.stanford.edu/projects/glove/
- *t-SNE for Word Embedding Visualisation*: Know as t-Distributed Stochastic Neighbor Embedding, a technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. For our project we used t-SNE to visualize our high dimensional Word Embeddings. We came up with interesting visualisations on relations between vectors captured from our texts.
  1) From the figure 5(a) it can be seen that in the word2vec model less number of related words to ”love” are near to it whereas in figure 5(b) we can see more number of similar words being captured near to it.
  2) Similarly from the figure 6(a) and 6(b) for the word ”hate” we see less number of words near to ”hate” in word2vec model compared to the glOve model.

We use these trained model as the pre-trained embedding layer for our LSTM network and we suspected that the GlOve model might produce us better results than the word2Vec model.

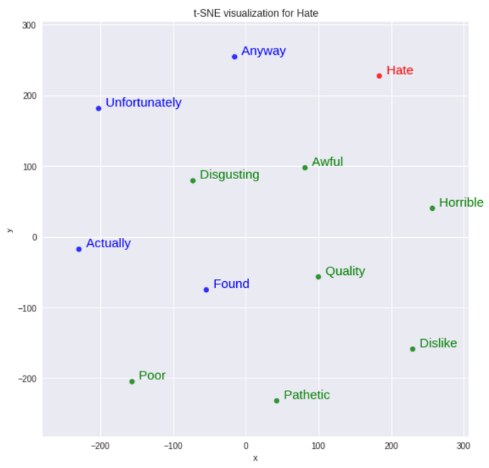


(a) Word2Vec Model

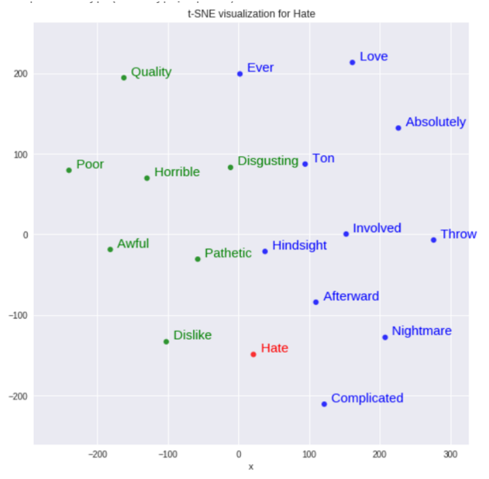


(b) GloVe Model

Fig. 5: Representation of ”Love”



(a) Word2Vec Model



(b) GloVe Model

Fig. 6: Representation of ”Hate”

IV. Classification

We determine a series of classifiers based on a similar approach used in [9]. As for the classifiers, we have used Support Vector Machine(SVM), Multinomial Naive Bayes(MNB), K-Nearest Neighbour(kNN), Logistic Regression(LR) and Adaboostng using the Scikitlearn python package. Other than the classifiers, we have done a neural network approach where we are using LSTM RNN model using the keras library. The classification metrics are compared for all the classifiers and the generalization of best classifier for our dataset is made then. Along with it, we are comparing the best classifier with the LSTM model and find the best suited approach for our dataset.

A. Baseline Approach 1: Support Vector Machine [9]

The Support Vector Machine works on the ideology of determining the decision boundary or the best suited hyper-plane separating the given classes. Traditionally it's well known for a two class problem. Linear SVM performs exceptionally in separating the two classes within a given feature space if the data is indeed linear. The Support Vector Machine is used in our classification and it was observed that they showed good performance compared to other models. A reason for this is that they possess the ability to generalise well in high dimensional feature spaces and eliminate the need for feature selection, making them a suitable choice of models for text categorisation tasks [1].

B. Baseline Approach 2: Multinomial Naive Bayes [9]

The basic ideology behind Multinomial Naive Bayes is that any vector that represents a sentence will have to contain information about the probabilities of appearance of the words of the sentence within all the sentences of a given category so that the algorithm can compute the likelihood of that sentence belonging to the category. Naive Bayes is also a probabilistic approach, computation time is extremely fast for a two class problem as the freedom of choice is just a yes or no problem so we decided to choose this classifier and compare the results.

C. Baseline Approach 3: K-Nearest Neighbour [9]

The kNN classifier works on the assumption that a particular classification of an object is most similar to the classification of other objects which are nearby to it in the vector space. The intuition behind kNN is that you find K-neighbors for a data point and based on the majority voting of the neighbor's labels decide the label of the data point under analysis. We have picked to see how kNN under performs when it comes to the task of text classification. The reason for it under performing is because of the high variance in the data where it is unable the classify the neighbours due to the uniqueness of each review and how the writing patterns are different. The other factor why kNN is a not feasible solution is that it stores the entire training data for the classification and is extremely sensitive to noise during training.

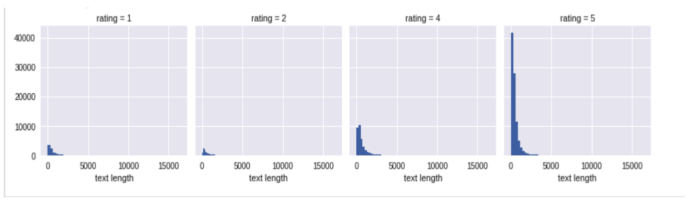
D. Baseline Approach 4: Logistic Regression [2]

Logistic regression estimates probabilities using a sigmoid function between the categorical dependent variable and one or more independent variables. This model works very well for linearly separable data and shows a good performance traditionally for sentimental analysis. Both logistic regression and naive bayes are linear classifiers meaning that they both find a best fit hyper-plane to separate the classes however the fundamental difference is that logistic regression tries to optimize a discriminate set of objective function and it is proven in literature that it performs better than Naive Bayes when there is a large training data. [4]

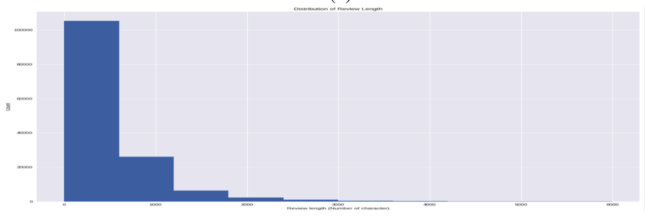
E. Baseline Approach 5: Adaboosting

Adaboosting focuses on classification problems and aims to convert a set of weak classifiers into a strong one. We used Adaboosting for our approach to see the results when several weak learners are combined (Ensemble Learning Approach). The idea of adaboosting is that we take several weak learners with less accuracy and run it multiple times on training data then let the learned classifiers take a weighted vote before the final prediction. Adaboosting refers to adaptive boosting where the sequential learning happens on weighted version of the data where the successive classifiers focus on the wrongly classified data from the previous classifiers. Adaboosting is also robust in terms of overfitting on the training data.

F. Long Short-Term Memory RNN



(a)



(b)

Fig. 7: Distribution of Review Length

Long Short-Term Memory (LSTM) networks are an extension for recurrent neural networks, which basically extends their memory. LSTMs enable RNNs to remember their inputs over a long period of time. Therefore, the model is best suited to remember right information over a long period of time. For our project, we implemented LSTM with four variations: LSTM on the pre-trained word2vec embedding and LSTM on the glOVe embedding for both balanced and original data distribution. While fitting our word embeddings on the LSTM model one important factor that we considered was the maximum review length. After analysing the distribution of our review length as seen in the figure 7, we observed that most of the reviews were under the length of 1000 as a result we took our maximum length as 1000 and tried to fit data in this particular maximum length while padding zero to the reviews having length less than 1000. LSTM with trained GloVe model as Embedding Layer gave us the best result among all the methods. After multiple iterations, the hyper-parameters which gave the best performance are as shown in table I. The Results are calculated based on the above hyper-parameters [1] and was run for 3 Epochs. Beyond which the model showed the signs of over-fitting.

Input length	Batch size	Embedding size	LSTM size	Hidden layer size	Dropout	Recurrent dropout	Activa tion	Output length	Optimizer	Output
1000	64	300	200	128	0.25	0.25	ReLU	2	Nadam	Sigmoid

TABLE I: Hyper-parameters for LSTM Model

V. Evaluating Measures

For getting the best parameters we have done hyper-parameter tuning using GridSearchCV. Our results are based on 5-fold cross validation and then for the comparisons we are taking weighted average of all folds. As our problem was a binary classification problem there are possible four outcomes for our classification:

- *True positives(TP):* Data points labeled as positive that are actually positive
- *False positives(FP):* Data points labeled as positive that are actually negative
- *True negatives(TN):* Data points labeled as negative that are actually negative
- *False negatives(FN):* Data points labeled as negative that are actually positive

The metrics that we have taken for evaluation are as follows: [6]

A. Accuracy:

The accuracy of a classifier on a given test dataset is the fraction of predictions our models got right. As our problem is binary classification problem, accuracy has the following definition:

$$Accuracy = \frac{TP + TN}{Totalnumberofexamples} \tag{1}$$

B. Precision:

It is a number of correct positives our model predicts compared to the total number of positives it predicts. Precision is a measure of exactness, quality, or accuracy. High precision means that more or all of the positive results predicted are correct. A precision score of 1.0 means that every item labeled positive, does indeed belong to the positive class. A precision score by itself though does not say anything about how many items of that class were not labeled. It is defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

C. Recall:

Recall is the number of positives that our model predicts compared to the actual number of positives in our data. Recall is a measure of completeness. High recall means that our model classified most or all of the possible positive elements as positive. A recall score of 1.0 means that every item from that class was labeled as belonging to that class. However, having just the recall score, we cannot know how many other items were incorrectly labeled.

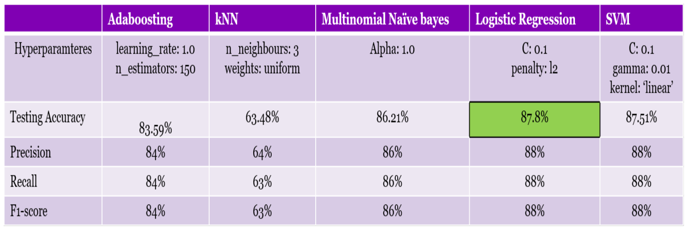
$$Recall = \frac{TP}{TP + FN} \tag{3}$$

D. F1 Score:

Precision and recall are often used together because they complement each other in how they describe the effectiveness of a model. The F1-score ombines these two as the weighted harmonic mean of precision and recall.

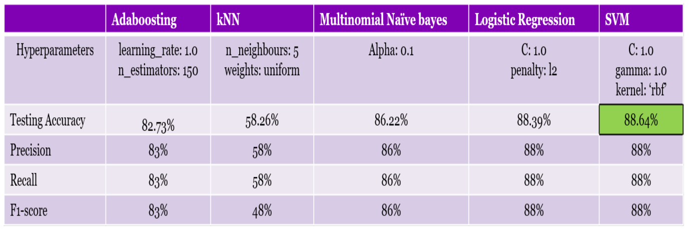
$$F1 = \frac{2 * (Precision * Recall)}{(Precision + Recall)} \tag{4}$$

VI. Results



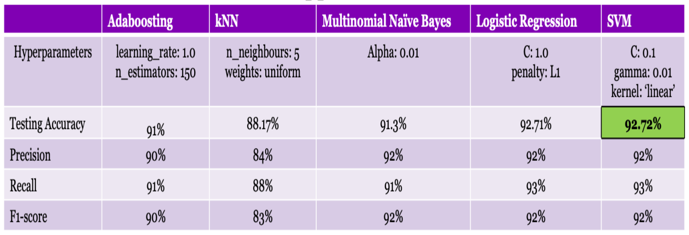
	Adaboosting	kNN	Multinomial Naïve bayes	Logistic Regression	SVM
Hyperparamteres	learning_rate: 1.0 n_estimators: 150	n_neighbours: 3 weights: uniform	Alpha: 1.0	C: 0.1 penalty: l2	C: 0.1 gamma: 0.01 kernel: 'linear'
Testing Accuracy	83.59%	63.48%	86.21%	87.8%	87.51%
Precision	84%	64%	86%	88%	88%
Recall	84%	63%	86%	88%	88%
F1-score	84%	63%	86%	88%	88%

TABLE II: Results for Balanced Data Distribution for Count Vectorizer approach



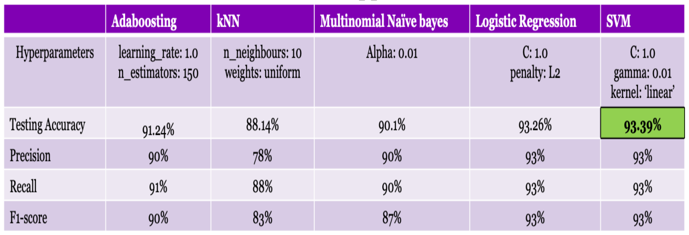
	Adaboosting	kNN	Multinomial Naïve bayes	Logistic Regression	SVM
Hyperparameters	learning_rate: 1.0 n_estimators: 150	n_neighbours: 5 weights: uniform	Alpha: 0.1	C: 1.0 penalty: l2	C: 1.0 gamma: 1.0 kernel: 'rbf'
Testing Accuracy	82.73%	58.26%	86.22%	88.39%	88.64%
Precision	83%	58%	86%	88%	88%
Recall	83%	58%	86%	88%	88%
F1-score	83%	48%	86%	88%	88%

TABLE III: Results for Balanced Data Distribution for TF-IDF approach



	Adaboosting	kNN	Multinomial Naïve Bayes	Logistic Regression	SVM
Hyperparameters	learning_rate: 1.0 n_estimators: 150	n_neighbours: 5 weights: uniform	Alpha: 0.01	C: 1.0 penalty: L1	C: 0.1 gamma: 0.01 kernel: 'linear'
Testing Accuracy	91%	88.17%	91.3%	92.71%	**92.72%**
Precision	90%	84%	92%	92%	92%
Recall	91%	88%	91%	93%	93%
F1-score	90%	83%	92%	92%	92%

TABLE IV: Results for Original Data Distribution for Count Vectorizer approach



	Adaboosting	kNN	Multinomial Naïve bayes	Logistic Regression	SVM
Hyperparameters	learning_rate: 1.0 n_estimators: 150	n_neighbours: 10 weights: uniform	Alpha: 0.01	C: 1.0 penalty: L2	C: 1.0 gamma: 0.01 kernel: 'linear'
Testing Accuracy	91.24%	88.14%	90.1%	93.26%	**93.39%**
Precision	90%	78%	90%	93%	93%
Recall	91%	88%	90%	93%	93%
F1-score	90%	83%	87%	93%	93%

TABLE V: Results for Original Data Distribution for TF-IDF approach

There are several classifiers used in our experiment like Support Vector Machine, Multinomial Naive Bayes, K-Nearest Neighbour, Logisitic Regression and Adaboosting. Additionally we used the LSTM Recurrent Neural Network for our Word Embedding. From all the experiments it can be seen that after getting hyper parameters as mentioned in the tables through GridSearchCV, Linear Support Vector machine with TF-IDF approach for the original data distribution gave the highest result with testing accuracy of 93.39% as well as in Count Vectorization SVM performed much better than the other classifiers achieving testing accuracy of 92.72%. We also found that Logistic Regression being the fastest in computation came second best with 87.8% testing accuracy for Count Vectorizer and 88.64% testing accuracy for TF-IDF for balanced data distribution. Additionally for balanced data distribution it showed 92.71% testing accuracy for CountVectorizer and 93.26% testing accuracy for TF-IDF and it almost gave as good result as SVM.

Balanced Data	LSTM + Word2Vec	LSTM + GlOve
Testing Accuracy	88.15%	92.38%
Precision	89%	92%
Recall	88%	93%
F1-score	89%	92%

TABLE VI: Results for Balanced Data Distribution for Word Embedding approach

Original Distribution	LSTM + Word2Vec	LSTM + GlOve
Testing Accuracy	94.25%	95.05%
Precision	94%	94%
Recall	94%	95%
F1-score	94%	95%

TABLE VII: Results for Original Data Distribution for Word Embedding approach

However kNN gave extremely bad results in balanced data distribution for both Count Vectorizer and TF-IDF with 63.48% and 58.26% testing accuracy respectively. Though it gave a little good result in the original data distribution as expected in our initial hypothesis thereby failing to generalize. Considering the balanced data distribution for the LSTM approach in the Word Embedding we got good result with glOve model achieving 92.38% testing accuracy. However we did not get results as expected for the word2vec model. The word2vec for the LSTM gave result just as good as the best classifier i.e. SVM. In original data distribution both glOve and word2vec model performed much better than the other classifiers showing 94.25% and 95.05% testing accuracy respectively. From the figure 8 and figure 9 the training and the testing accuracy can be seen and it can be observed that LSTM with glove model performed the best.

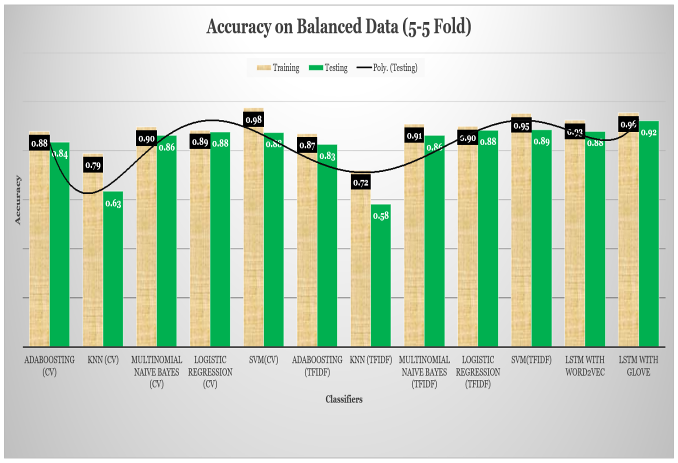


Fig. 8: Training and Testing accuracy for balanced data

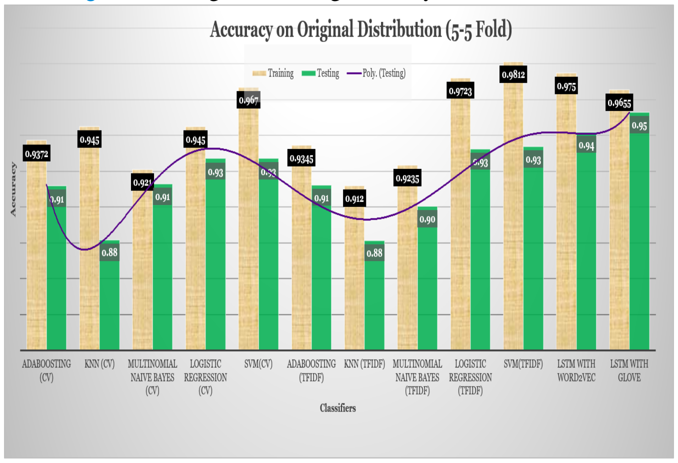


Fig. 9: Training and testing accuracy for original distribution

VII. Difficulties and Improvements

Some of the difficulties that we faced was that after tuning parameters for the word2vec model on the balanced data distribution for our architecture, we were getting almost the same result as the best classifier i.e. SVM in BoW approach. Though a better result was expected. Thus we opted for the GloVe model as a embedding layer over the Word2Vec model as it served better features. However Word2vec with LSTM on the original distribution of the data gave better results than the BoW approach. We also noted that by POS tagging and selectively considering only the adjectives as mentioned in paper [5] showed little or no signs of improvement. As our dataset was huge and due to lack of hardware resources we faced hard time computing our results.

VIII. Conclusion

In this project, we performed a supervised learning approach for detecting the polarity of our reviews in our dataset. We classified our reviews for both balanced and original data distribution. After 5 fold cross validation for evaluating our approaches we came to some interesting results. Based on our results we found that LSTM approach using GloVe embedding performed the best for our dataset in both balanced and original distribution of the data. In terms of BoW approach SVM with TF-IDF outperforms all other classifiers. Its worthy noting that MNB and LR classifiers computation time was extremely fast and provided decent results though lesser than our best classifier. On the whole we arrive to conclusion that KNN is the worst performing model in this kind of application due to the high variance in the data. Also we saw that the distribution of ratings in the data has a meaningful impact on model performance where the original distribution gave us better performance than the balanced data.

IX. Future Work

In future we would like to apply our techniques for a Multiclass Classification for ratings between (1-5). Also we will try to see the performance of classifiers using Over-Sampling techniques. Our future work also include to perform a text summarization of the reviews. We would also like to improve our models by applying hyper-parameter tuning and adding more LSTM layers. We would like to see how the model behaves for reviews which are sarcastic or of longer length than our assumption stated in this scope of our project.

References

[1] James Barry. Sentiment analysis of online reviews using bag-of-words and lstm approaches. In *AICS*, 2017.
[2] Maria Soledad Elli and Yi-Fan Wang. Amazon reviews, business analytics with sentiment analysis.
[3] Xing Fang and Justin Zhan. Sentiment analysis using product review data. volume 2, page 5. Springer, 2015.
[4] Andrew Goldberg. Cs838-1 advanced nlp: Automatic summarization. Madison: University of Wisconsin-Madison, 2007.
[5] Tanjim Ul Haque, Nudrat Nawal Saber, and Faisal Muhammad Shah. Sentiment analysis on large scale amazon product reviews. In *Innovative Research and Development (ICIRD), 2018 IEEE International Conference on*, pages 1–6. IEEE, 2018.
[6] https://towardsdatascience.com/beyond-accuracy-precision-and recall. Evaluation metrics definitions. www.towardsdatascience.com, 2016.
[7] Yi Sun Mingxiang Chen. Sentimental analysis with amazon review data. Stanford University, 2016.
[8] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.
[9] Qinxia Wang, X Wu, and Y Xu. Sentiment analysis of yelps ratings based on text reviews. 2016.