PREDICTING USERS RATING BASED ON THE TEXT REVIEW

Anam Parveen Khan (0879226)
Dept of Computer Science
Lakehead University,
Thunderbay, Canada

Sandeep Nannamu(0882000)
Dept of Computer Science
Lakehead University
Thunderbay, Canada

Vatan Singh Nayal(0888966)
Dept of Computer Science
Lakehead University
Thunderbay, Canada

***Abstract*—Predicting users rating based on the text review, is a sentiment analysis technique and one of the most important topics currently in Machine learning. In this project, we will make a machine learning model that learns to predict the rating of a product, which a user gave. In this project, we have used the Amazon review dataset, which contains more than 34000 reviews that we have used to train the model. We have used light gbm machine learning model for training our data, and developed a model with 74% accuracy.**

Keywords— Sentiment Analysis, Convolutional Neural Networks, Gradient boosting, Decision Tree, Machine Learning, Amazon, LightGBM

I. INTRODUCTION

Amazon For over 20 years many e-commerce websites like Amazon have been created to fulfill rising requirement of people. Nowadays, customer can buy a product from any website having different prices but for quality of the product he can't check it. In this case, reviews from other consumers can help in deciding whether to buy a product or not. Therefore, sentiment analysis is popular among the buyers.

The largest growing business all over the world is Amazon. Many people buy different products from amazon and also before purchasing any product they look at the reviews for that product. It becomes tough to read and understand every comment. So, to make it easy, we are predicting the reviews and rating on the scale of 1,2,3,4,5. We aim to build a system that tells review's sentiment in the form of a number.

Sentiment analysis is a method of determining whether a text is positive, negative, or neutral using techniques like text analysis, natural language processing. It is a widely used topic which is applied to all the areas where there are human interactions like customer reviews. As sentiment analysis enables us to collect subjective knowledge from the investigation, there are meaningful values hidden in it. One can easily find out the strengths and weaknesses of the product by just examining the opinions on the product. Because of the bright future of sentiment analysis, many researchers are looking forward to studying it. Figure 1 below describes the total number of studies on sentiment analysis, as we can see that it is increasing rapidly over the years.

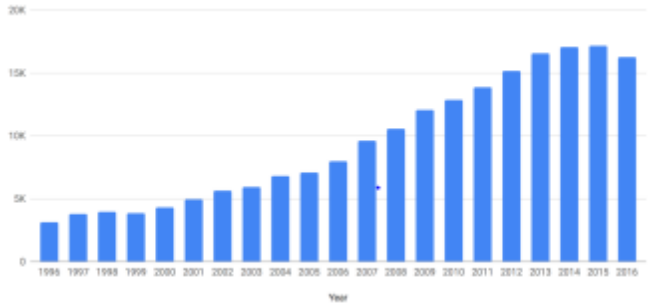

Fig. 1 – Increase in studies regarding sentiment analysis over the years.

The objective of this paper is to predict the reviews given by consumers for different products. Our dataset contains customer reviews and ratings of different amazon products. We have used a machine learning algorithm and also used lightGBM model for training our dataset. We have used lightGBM model because it can handle a large amount of data and takes less memory to run. Moreover, we have split our dataset into training and testing. For training, we have used 80% data, and for testing, we have used 20% data. Also, we have refined our data to get a better result. Finally, we got an accuracy of 74%.

Facts and opinions are the two classes on which we can analyze all the information in the world where objective statements are facts, and subjective statements are opinions. In old times there was no data available to give the opinion to the people to make any decision. But with the introduction of www, the methods of getting one's opinion has changed. One can give reviews for any product which helps other people to buy that product. With the development of technology, the ways of processing opinions have also changed. Figure 2 below shows the steps for feature extraction from reviews.

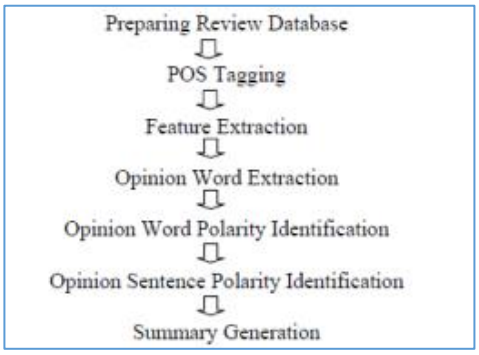

Fig. 2 - Steps for feature extraction.

Algorithms like Probabilistic Machine Learning approach, Naive Bayes Classification, etc. can be used to determine the reviews of amazon as positive, negative, or neutral. In our work, we have used product reviews from amazon dataset. We studied our dataset and found out that the reviews are in the form of text. So we decided to classify reviews into a scale of 1 to 5. Our work mainly focusses on predicting the ratings of reviews and getting better accuracy.

II. RELATED WORK

Hu and Liu compiled a positive and negative list of words on reviews by the customer, which contains 2006 and 4783 words, respectively. There were also misspelled words. Sentiment Analysis is a text classification process, where text containing sentiment information should be recognized before the classification.

For feature selection, Pang and Lee recommended eliminating objective sentences by extracting subjective ones. They introduced a technique called text categorization that is able to classify subjective content. Gann et al. selected 6,799 tokens based on Amazon data, where sentiment score is assigned to each token as a positive or a negative token. Specifically, a TSI for a certain token is computed as:

$TSI=(p-(tp/tn)*n)/ (p+(tp/tn)*n)$

Where p is the appearance of the token in positive reviews and n is the appearance of the token in negative reviews. tptn is the ratio of total number of positive and negative reviews

III. METHODOLOGY

3.1 Problem Statement

Given a set of amazon product reviews, we think about the plausibility of classifying them into distinctive sentiment classes so that we can predict the rating given for those reviews. From each review, we extricate distinctive sets of features, allude to a manually configured training set, and utilize machine learning to perform the classification. Other than the rating classification itself, we analyze the effect of the number of review categories on the classification performance like accuracy, multi-error.

3.2 Approach

An end-to-end content text classification pipeline is primarily composed of the below components:

Training text content: It is the input content through which our trained model is able to learn and anticipate the desired class.

Feature Vector: A feature vector may be a vector that contains data depicting the characteristics of the input data.

Labels: These are the predefined categories/classes that our show will predict

Machine Learning Algorithm: It is the algorithm through which our model is able to learn about text content classification (In our case : LighGBM)

Prediction model: A model which is prepared on the verifiable dataset which can perform rating prediction.

So, considering the above process, our high level architecture can be defined as shown in figure 3

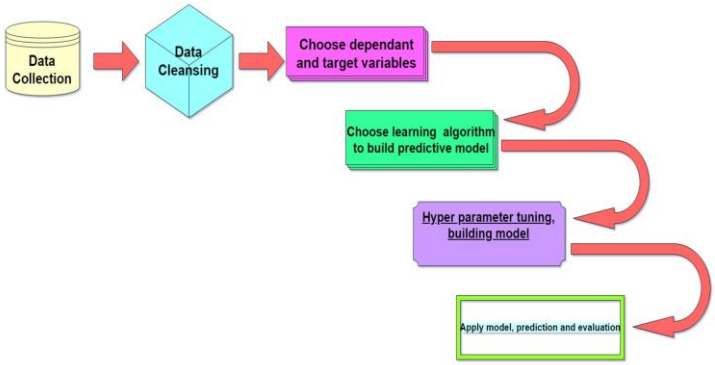

Fig. 3 – High Level Architecture.

Data Collection – First step is to collect a reasonable amount of raw data which could be enough for training a good prediction model.

Data Cleansing – Next, we will reshape the raw data into a fixed format, which is easily understandable and consistent. Data cleansing is the most important step in this whole process, the cleaner your data, the better will be your model training. It's in this step, that we try to remove the unwanted elements from our data. Dimensionality reduction like deleting unusable columns, or deleting rows with null values (: We have a review but no rating, that review isn't going to help our model during training or during evaluation, it's better to remove it)

Choose dependent and target variables – Feature selection is the vital step. We select all the important variables(columns) which we think might help the model in prediction, and we define our target variable (In this case, it's the rating) to feed into our training model.

Select a machine learning algorithm – There are many NLP classification algorithms available such as Convolutional Neural Network algorithms, Recursive Neural Network Algorithms, Gradient Boosting algorithms etc. IN this case we went ahead with one of the most popular gradient boosting algorithm LightGBM.

Building your model – We convert the dataset into training and testing. Normally at least 75% of the dataset is used for training. For our model we split the dataset into 80% for training and 20% for testing. We feed the training data and the target variable into our model for

training and evaluate the performance metrics. We fine tune the model parameters and try out various combinations of the input parameters to get the best performance metrics for our model.

Apply model for prediction and evaluation of model – We then apply this model on our test dataset and evaluate the correctness of our model based on metrics such as accuracy.

Let us go in detail about the various steps that we are going to follow for developing this prediction model. Here is our model flow chart.

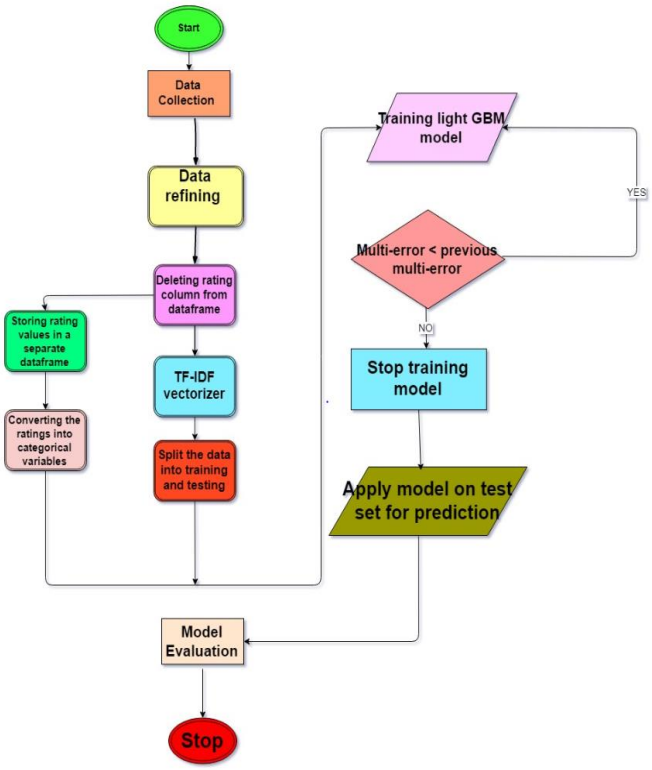

Fig. 4 – Flow chart of our model building process.

Data collection – We have collected a sample dataset of almost 35000 amazon product review with user ratings provided by datafinity.

Data refining – We analyze our dataset, clean our dataset by remove unnecessary rows and columns and select the features which contribute towards rating prediction.

Store the rating variable in a separate dataframe – We delete the rating variables from our dataset and store all those values in a separate dataframe. This is going to be our target variable which we will be feeding into our algorithm for model training.

Convert the rating values into categorical variables – The rating field will be having the following values 1,2,3,4,5 where 5 indicates extremely positive review and 1 indicates extremely negative review, which the model considers as numerical values by default. All these values represent different classes of users' opinion and not just numerical numbers. So for the model to consider the values as 5 different categories, it is vital that we convert this variable from numerical to categorical variable. Nobody would like to see rating predicted in the from 3.678555…..

TF-IDF vectorize – Next, we transform the dataframe into a machine understandable format. We will be using the TF-IDF (term frequency-Inverse document frequency) vectorizer to convert the entire dataframe into a standard CSR matrix format which the machine learning can understand.

Splitting the data – We split the dataset into two different sets, one for training the model and one for testing the model. For our model, 80% of the data is used for training and 20% is used for testing.

Training model – We use a gradient boosting machine learning algorithm called LightGBM to train our model. This training is an iterative process, wherein we define the total number of rounds we want the model to run, define the number of rounds after which the model stops and checks the error-rate. The evaluation metric used here for this model in multi-error which is the error rate for multi-class classification. The model keeps on training as long as the multi-error keeps reducing. When the multi-error stops reducing, it means the model's accuracy stopped improving and there is no point running more rounds on the model and hence the training stops, and model is finalized.

Apply model on test set – Finally the model is applied on testing dataset and ratings are predicted. These ratings are then compared against the actual user ratings and the model accuracy is calculated.

3.3 Overview of dataset

For our research, we have collected a dataset containing user reviews and ratings of amazon products. Amazon is an online retail website, founded in 1994 by Jeff Bezos. It is one of the biggest E-commerce websites in the world, and millions of reviews gets posted on Amazon on various products each month. There are two types of reviews expressed in a review, either direct or comparative. Example of direct review is "The sound quality is very good", and example of comparative review is "X Speaker is better than Y". We took an Amazon review dataset from Kaggle.com provided by Datafiniti. Datafiniti provides refined information for data-driven businesses. They give the foremost accurate, comprehensive information on businesses, individuals, items, and properties to control our applications and analytics. The dataset taken by us is a sample dataset which contains more than 34,000 consumer reviews for Amazon electronic products such as Kindle, Fire TV stick, Alexa etc. This dataset is part of a larger dataset provided by Datafiniti and the original dataset has millions of reviews.

This dataset was at first collected to analyze Amazon's most fruitful customer electronic item launches; find critical knowledge into user audits and help with machine learning models. E.g.: What are the most looked into Amazon products? What are the initial and current number of client surveys for each product? How do the reviews in the starting 90 days after an item release compare to the price of the product? How do the reviews in the starting 90 days after a product release compare to the days accessible for sale? Match the keywords within the user review content with the user ratings to assist in sentiment prediction models. We will be using these reviews for training a model which can predict the rating based on a user review.

The dataset contains 24 attributes described as below:

- ID: Unique ID of each and every review
- Name: Name of the product.
- Asins: Amazon Standard Identification Number
- Brand: Brand name of the product.
- Categories: Category of the product (Electronics, Clothing etc)
- Keys: Amazon key of the product
- Manufacturer: manufacturer of the product.
- Reviews.date : Date of the review
- didPurchase: Did the user who posted the review, purchased the product or not.
- reviews.doRecommend does the user recommend this product or not
- Reviews.id: Amazon ID of the review
- reviews.numHelpful: Number of people that upvoted this review
- Reviews.rating: The rating that the user gave to the product with the review, we will predict this attribute.
- reviews.sourceURLs: URL of the review
- Reviews.text: Text of the review
- Reviews.title: Title of the review
- reviews.userCity: City of the user
- reviews.userProvince: Province of the user
- Reviews.username: Username of the user who posted this review

But we will not be using all the above mentioned attributes for predicting the rating. Some of them cannot help us in predicting the rating for example the username, user city and some attributes like "reviews.DoRecommend", cannot be used because they contain direct information about the rating. If a user recommends a particular product, then he/she has given 4-5 rating to the product. So we will only use category of the product, name of the product, title of the review and text of the review for predicting the rating.

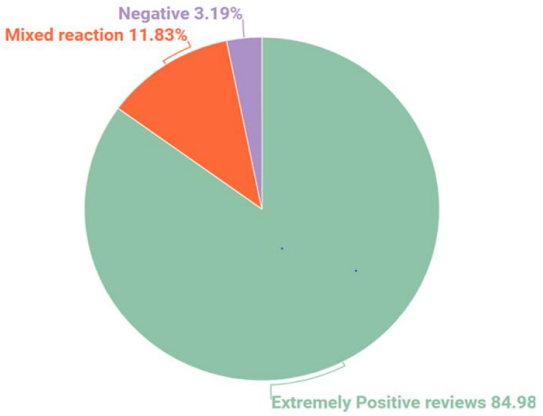

Fig. 5 – Pie chart showing the percentage of different types of reviews present in our dataset.

The above pie chart shows a comparison of the amount of different type of reviews present in our dataset. We can further compare the amount of reviews for each rating in dataset by looking at a bar chart as shown below.

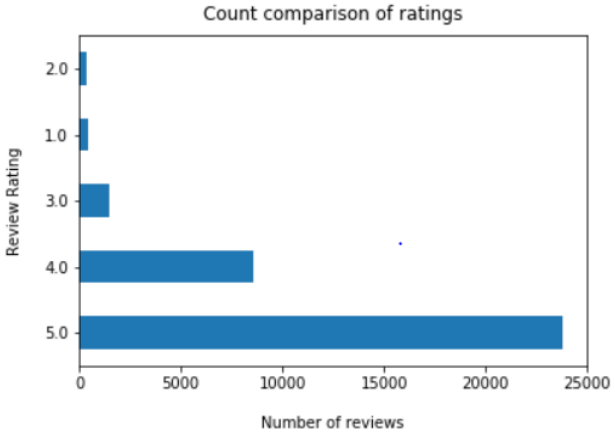

Fig. 6 - Bar chart comparing the amount of reviews for each rating.

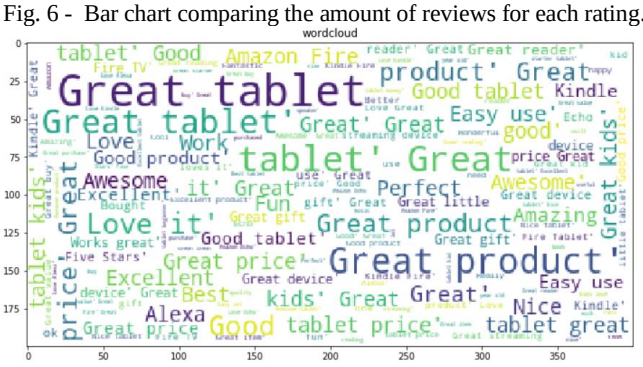

Fig. 7 – A wordcloud showing the most frequently occurring words in the reviews in our dataset.

After having a high level look at the data, we can conclude that majority of reviews are positive and negative reviews are significantly less compared to positive reviews.

3.4 LightGBM Algorithm

Let us have a look at the machine learning algorithm being used here for developing our model. Machine learning algorithms are anticipated to supplant 25% of the employments over the world, within the following 10 years. With the quick development of enormous information and accessibility of programming apparatuses like Python and R –machine learning has become a staple mode of application for information researchers. Machine learning applications are profoundly mechanized and self-modifying which proceed to make strides over time with negligible human mediation as they learn with more information. There are many prominent machine algorithms which can be used for text classification like Naïve Bayes Classifier Algorithm, K Means Clustering Algorithm, Support Vector Machine Algorithm, Decision Trees, Nearest Neighbours. We have used a gradient boosting algorithm known as LightGBM.

Gradient boosting is a machine learning procedure for regression and classification problems, which produces a forecast model within the frame of an ensemble of powerless forecast models, normally decision trees. It builds the model in a stage-wise mold like other boosting strategies do, and it generalizes them by permitting optimization of an subjective differentiable loss function. Some prominent gradient boosting algorithms are XGBoost, AdaBoost, CatBoost and LightGBM.

The reason to use LightGBM is because it is designed to be dispersed and proficient with advantages such as faster processing speed and higher efficiency, lower memory usage, better accuracy, support of parallel and GPU learning, capable of dealing with large-scale information.

LightGBM employs histogram-based calculations, which bucket attribute values into discrete bins. This speeds up training and decreases memory utilization.

Some major plus points of histogram-based calculations incorporate the following: Cost of calculating the gain for each split is greatly reduced. Use histogram subtraction for increased speedup. To get one leaf's histograms in a parallel tree, it utilizes the histogram subtraction of its parent and its neighbor. So it must develop histograms for one leaf (with littler #information than its neighbor). It at that point can get histograms of its neighbor by histogram subtraction with little fetched (O(#bins)). Reduce memory usage, replaces persistent values with discrete canisters. In the event that #bins is little, can utilize small data types, e.g. uint8_t, for training data storage, need not store extra data for pre-sorting features, communication cost for parallel learning is greatly reduced.

Some of metrics which LightGBM supports for evaluation of model correctness are L1 loss, L2 loss, Log loss, Multi-class error rate, Fair, Huber. Because our predicted outputs are in the form of multiple categories, we will be using the multi-calss error rate metric for measuring the accuracy of our model during training phase.

This is how the Light GBM algorithm works. For a number of boosting rounds M and a differentiable loss function L:
Let $F0(x)=\arg\min_\gamma \sum=L(yi,\gamma)$
For m = 1 to M:

1. Calculate the pseudo residuals

   $$rim=-\partial L(yi,Fm-1(xi))/\partial Fm-1(xi)$$

2. Fit decision tree $hm(x)$ to rimrim
3. Compute the step multiplier γm for each leaf of hm(x)
4. Let $Fm(x)=Fm-1(x)+\lambda m\gamma mhm(x)$, where λm is the learning rate for iteration m

LightGBM can use categorical features directly (without one-hot encoding). The makers claim that it is 8 times more faster compared with algorithms involving one-hot encoding.

3.5 Procedure

Step 1. Vectorization: After data pre-processing and cleaning steps, we are using Tfidfvectorizer(). This helps in converting words into a matrix of TF-IDF features.
Step 2. Word Embedding: we apply word embedding function to name, text and title columns and store result in new variables using vectorize.fit_transform function.
Step 3. The last column in the df is category. This field contains multiple categories separated by ",". So, we split the column by ",". Then we use pandas.get_dummies function to convert categorical variable into dummy variables.

```
df = df.categories.str.split(",",expand=True)
```

Step 4. Using hstack function, we stack arrays/ concatenate horizontally.

```
[ ]  df = hstack((csr_matrix(df),nameNLP,reviewTextNLP,reviewTitleNLP ))
```

Step 5. In the dataframe y, ratings are in numeric type. We want the model to predict rating in categorical format – 1,2,3,4,5. So, we convert the rating to category format.

```
y = y.astype('category').cat.codes
```

Step 6 - Splitting the data: Using train_test_split, we split the data into train and test datasets. The test size was 20% and train size 80%.

```
[14] from sklearn.model_selection import train_test_split

[15] train, test, y, y_test = train_test_split(df, y, test_size=0.2, random_state=42)
```

Step 7. LightGBM: Here, we give train and test variables to lightGBM that helps to transform them to a format that machine accepts.

Step 8. Model training:

Parameters used:

Objective - multi-class. The objective is multi class when we have more than two categories.

Num_leaves - This parameter controls the complexity of model.

Num_class - Total number of classes here is 5 (ratings).

Learning_rate - It is a hyperparameter that determines to what extent newly acquired information overrides old information.

Verbosity: controls the level of lightGBM's verbosity.

We train the model using lightgbm.train function. Number of epochs/iterations have been set to 500. The valid_sets are training and test set. So, the model is designed in such a way that it will check for every 50th round of training whether the accuracy is still increasing. If the accuracy stops increasing, the training is stopped.

```
train_set = lightgbm.Dataset(train, label=y)
test_set = lightgbm.Dataset(test, label=y_test)

params = {'objective':'multiclass',
          'num_leaves' : 30,
           'num_class':5,
          'learning_rate': 0.05,
          "metric": 'multi_error',
          "verbosity": -1}
evals_result = {}
model = lightgbm.train(params
                       , train_set
                       , num_boost_round = 500
                       , valid_sets = [train_set,test_set]
                       , verbose_eval = 100
                       , early_stopping_rounds = 50
                        ,evals_result=evals_result

                      )

Training until validation scores don't improve for 50 rounds.
[100]   training's multi_error: 0.217104        valid_1's multi_error: 0.267109
[200]   training's multi_error: 0.183387        valid_1's multi_error: 0.264222
Early stopping, best iteration is:
[222]   training's multi_error: 0.178333        valid_1's multi_error: 0.262778
```

Fig. 8 – Snippet of code running the lightGBM model.

```
print('Plotting metrics recorded during training...')
ax = lightgbm.plot_metric(evals_result, metric='multi_error')
plt.show()

Plotting metrics recorded during training...
```

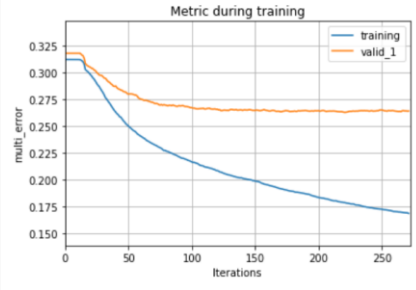

Fig. 9 – Plotting the metric during training.

In figure 8 and figure 9, we can notice that in 222nd iteration the model stopped training as the accuracy was not increasing. We can observe that when the model training started at 0th iteration, the error rate was approximately 30%, but when the model finished training the error rate improved and stood at 17%.

Model Evaluation: Here, we evaluate the accuracy of the model using model.predict function. So, we tested our model with different epochs and early stopping rounds, the accuracy achieved by the model in different cases is listed below:

Num_boost_round	early_stopping_rounds	Accuracy
500	50	73.72%
200	30	73.56%
100	20	73.28%
50	10	68.19%

Table 1 – Comparing the accuracy after changing the model parameters.

After comparing original ratings with actual ratings, we have observed that the model distinguishes well between and negative and positive reviews. It performs better for positive reviews prediction than the negative reviews rating prediction. This might be due to the factor of our sample dataset having positive reviews in majority. The model also has trouble differentiating positive and extremely positive reviews (rating 4 and rating 5). In order to improve the correctness of this model, much more training is needed with larger volume of dataset.

IV. CONCLUSION

In this paper, we have discussed what a discussed the importance of sentiment analysis in today's world, the application of machine learning algorithms in sentiment analysis. We have developed a gradient boost model using amazon product reviews dataset for training and we proposed some metrics to evaluate the model performance. We finally we deduced that in order for this model to perform accurate predictions, it still has to trained more. Sentiment analysis prediction models often falter because of factors such as context dependency, multiple sentiments present in a text which makes it harder to extract the overall sentiment, presence of words with multiple meanings in content and handling negation in statements. These challenges have to be overcome with gruesome training on large volumes of data in order to build an accurate prediction model.

REFERENCES

[1] K. Ghag and K. Shah, Comparative analysis of the techniques for sentiment analysis, in Proc. 2013 Int. Conf. on Advances in Technology and Engineering, Mumbai, India, 2013, pp. 1–7.

[2] K. H. Y. Lin, C. H. Yang, and H. H. Chen, What emotions do news articles trigger in their readers? In Proc. 30th Annu. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, Amsterdam, Netherlands, 2007, pp. 733–734

[3] L. Ye, R. F. Xu, and J. Xu, Emotion prediction of news articles from reader's perspective based on multilabel classification, in Proc. 2012 Int. Conf. on Machine Learning and Cybernetics, Xi'an, China, 2012, pp. 2019–2024.

[4] W. B. Liang, H. C. Wang, Y. A. Chu, and C. H. Wu, Emoticon recommendation in microblog using affective trajectory model, in Proc. 2014 Asia-Pacific Signal and Information Processing Association Annu. Summit and Conf., Chiang Mai, Thailand, 2014, pp. 1–5.

[5] Pang and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In ACL, pages 115–124

[6] T. Wilson, J. Wiebe, and P. Hoffman. 2005. Recognizing contextual polarity in phrase level sentiment analysis. ACL.

[7] M Hu and B Liu. 2004. Mining and summarizing customer reviews

[8] Y. Xu, X. Wu, and Q. Wang. Sentiment analysis of yelps ratings based on text reviews, 2015

[9] M. S. Elli and Y.-F. Wang. Amazon reviews, business analytics with sentiment analysis

[10] C. Rain. Sentiment analysis in amazon reviews using probabilistic machine learning. Swarthmore College, 2013.

[11] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. Journal of Machine Learning Research, 12(Aug):2493–2537, 20

[12] Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. Technical report, Stanford.

[13] Michael Gamon. 2004. Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. Proceedings of the 20th international conference on Computational Linguistics.

[14] Apoorv Agarwal, Fadi Biadsy, and Kathleen Mckeown. 2009. Contextual phrase-level polarity analysis using lexical affect scoring and syntactic n-grams. Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009), pages 24–32, March.