Amazon Reviews Sentiment Analysis

Abhishek Chandra Shukla, Sai Venkata Manoj Vungarala, Ishal Abhishek Mummidivarapu, Vishnu Rohan Surapaneni
Khoury College of Computer Science
Northeastern University
Boston, MA 02115

Abstract

Sentiment analysis holds a great value for business entities as they provide insights that might be useful for the sustenance of the business and turning the business profitable. Therefore we are performing sentiment analysis on a 9 years amazon dataset of beauty products. We will be creating different classification models trained on this data, so that we can predict the sentiment of a new review. In order to choose the best classification model, we will be training the models on different word embeddings generated from Countvectorizer, SVD embeddings, TF-IDF vectors and Word2Vec embeddings. We have chosen F1 score as a performance metrics as there is a class imbalance in the data.

1. Introduction

Sentiment analysis is a branch of Natural Language Processing (NLP) that has been gaining popularity over time. While keeping an eye on online interactions, contextual mining of text and extracting subjective information from the source material. It is also used for assisting businesses in understanding the social sentiment of their brand, product, or service.

Sentiment analysis has a wide range of uses, including areas such as marketing, politics, eCommerce etc. Businesses, particularly those involved in eCommerce, also use sentiment analysis to gather and examine client reviews of their goods. In addition, prospective customers prefer to read the reviews of previous clients before making a purchase or using a service from a business. E-commerce, as can be seen, has two main components: the online retailer, who wants to increase e-commerce sales or services, and the consumer, who prefers the finest product or service over alternatives.

In this project we are using the Amazon dataset. We intend to create a program that will recognize the positive and negative words that consumers use when they leave reviews for cosmetics as a sign of their propensity to make a purchase. We are using 9 years of customer reviews of beauty items from 2005 to 2014 to create a model that would recognize the positive and negative terms that consumers used to express their feelings about the company's beauty products. Reviews and ratings from customers make up most of our dataset.

The beauty review dataset is real commercial data that revolves around reviews written by customers. The data contains 28798 rows and 9 feature variables. This data is available on the

Stanford analysis project webpage. The original data is in a JSON file format which is converted into a CSV file format using a query.

2. Method

2.1 Data wrangling filter method

The columns present in the data are as follows:

- reviewerID - ID of the reviewer
- asin - ID of the product
- reviewerName - name of the reviewer
- helpful - helpfulness rating of the review
- reviewText - text of the review
- overall - rating of the product
- summary - summary of the review
- unixReviewTime - time of the review (unix time)
- reviewTime - time of the review (raw)

When we analyze the data we see that reviewerName has a few missing values and since customers remain anonymous for various reasons we cannot use this column as a unique identifier for the reviews. And we did not consider their reviews to make an analysis as we think we cannot rely on the opinion of a non-user for the analysis so we dropped the respective rows.

The data frame columns reviewText and summary deliver the same information, thereby we have concatenated them. Another column “helpful” gives the usefulness of the product which we classified into two columns as positive and negative feedback. The reviewer ID and reviewerName were the columns used for the identification of customers. There were a lot of anomalies pertaining to the different styles they were represented in. ‘unixReviewTime’ was dropped since it has already been represented in the ‘reviewTime’ column in a more readable format. Apart from that, the ‘reviewTime’ feature is converted to datetime data type and a new ‘year’ column was added to draw analysis between other features for future work. After which, ‘reviewTime’ column was also dropped.

We changed the names of the columns to something more meaningful and convenient.

2.2 Data structuring

Our project’s goal is the binary classification of reviews, so we classified the overall rating as good and bad. To achieve this, we have considered ratings 1,2 and 3 as bad and rating 4 and 5 as good. We created a new column “rate_class” from the “overall” column and mapped its observations as “good” and “bad”. Later the “overall’ column was dropped.

2.3 Data cleaning

The data available to us is the most unstructured form of data, filled with different types of noises. Without proper pre-processing analysis of that data is difficult. Therefore, we performed cleaning, and standardization of data to make it noise-free. Some of the pre-processing steps we performed are: Removing punctuations, Removing accented characters and converting the characters into ASCII characters, Removing URLS, Removing Stop words, Converting into lowercase, Removing Special Characters, Expanding contractions to original for to help us with text standardization, Tokenization, Stemming, and Lemmatization.

After preprocessing the text, we have 20,24,818 words in total with a vocabulary size of 27739, the maximum review length and the minimum review length are 1090 and 1 respectively. A clean dataset will allow the model to learn meaningful correlation and not overfit irrelevant noise.

2.4 Modeling

After getting the 'cleaned_data_review' CSV file from the Data cleaning, we then implemented the classification task using different models.

The models we are going to use for the classification are:

(a.) Logistic Regression: Logistic regression estimates the probability of an event occurring, here it is good or bad, based on a given dataset of independent variables. Logistic regression applies the logistic sigmoid function to weighted input values to generate a prediction of the data class. Since the outcome is a probability, the dependent variable is bounded between 0 and 1.

(b.) Naïve Bayes: The Multinomial Naive Bayes algorithm is a popular Bayesian learning method in Natural Language Processing (NLP). Using the Bayes theorem, the program guesses the tag of a text, such as an email or a newspaper story. It computes the likelihood of each tag for a given sample and returns the tag with the highest chance. Multinomial Naive Bayes is a subset of Naive Bayes in which the P(Feature_i|Class) distribution is multinomial (word counts, probabilities, etc.)

(c.) Random Forest: A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. For classification tasks, the output of the random forest is the class selected by most trees. Bagging is an effective ensemble algorithm as each decision tree fits on a slightly different training dataset, and in turn, has a slightly different performance.

(d.) CatBoost: CatBoost is a gradient boosting algorithm for decision trees. CatBoost's main advantage is that it can include categorical and text functions in your data without any additional preprocessing.

(e.) XGBoost: XGBoost, which stands for Extreme Gradient Boosting, is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library. It provides parallel

tree boosting and is the leading machine learning library for regression, classification, and ranking problems.

For the convenience of usage let us call them the “Group of models”.
We have computed and compared the classification of the reviews using the following Word embeddings.

2.4.1 Binary classification using CountVectorizer word embeddings

CountVectorizer: Count vectorizer breaks the sentences in words along with some basic pre-processing such as removing the punctuations, special characters etc. Words in a document are encoded in the form of indices and the count of occurrences of the word mapped to them. An encoding vector is returned whose length is equal to that of the entire vocabulary of the document.
We have generated the word embedding using CountVectorizer word embeddings. We have used the SK-learns library to import the model class.

Truncated SVD: In this method, we have considered truncated SVD. We have used the count vectorizer method to generate the word embeddings that we fed to the group of models we mentioned above. Once that has been done, we move forward to apply the models like XGBoost, Logistic Regression, CatBoost, and Random Forest.

2.4.2 Binary classification using TF-IDF word embeddings

TF-IDF: One of the most common practices to get a word embedding is one hot encoding. But this gives us a problem by giving more importance to the common words in the documents. And the less common words that are often the main differentiating factors between the documents are given less importance. This problem is solved by Tf-Idf where the term frequency (count of occurrence of word in observed document) is multiplied with inverse document frequency(count of occurance of the word in other documents/ total documents) to weight phrases in accordance.

We have used sklearn feature_extraction module’s TF-IDF to fit the train and test data to get the word embeddings. On top of these word embeddings we have deployed 5 different NLP classification methods namely Naive Bayes classifier, XGBoost, Logistic Regression, CatBoost, and Random Forest. We have created a few python functions for modeling, model comparisons etc. Used GridSearchCV for hyper parameter tuning and we have plotted confusion matrices for all the models and appended that data into a dataframe which will be later used during model comparison plots.

2.4.3 Binary classification using genism’s Word2Vec embeddings

Word2Vec: In Word2Vec each word is converted into a vector, which is then trained in a manner resembling a neural network. The vectors make an effort to depict multiple aspects of the word in relation to the whole text. The word's semantic relationship, definitions, context, and

other elements can be included in these characteristics. We can perform a lot of things with these numerical representations, including determining how similar or unlike two words are. These are undoubtedly essential as inputs to a variety of machine learning techniques. Text in its raw form cannot be processed by a machine; therefore, turning the text into an embedding will enable us to feed the embedding to conventional machine learning models

We have used genesims Word2Vec word embeddings to generate word embeddings for the words in the corpus. We have set the size of the word embeddings as 100 and the max length of the review as 200. As we know negative reviews are usually longer than positive reviews, we need to set up a bounding condition so that one case does not dominate the other. As Word2Vec embeddings are not suitable for the statistical machine learning models we have been discussing above, we have implemented a simple neural network for the purpose of classification.

(a.) Architecture:
We have used Keras's sequential model as our model
Used an embedding layer which acts like a look-up dictionary for the words and their indices.
Used a flatten layer to truncate the information which avoids overfitting.
Followed by 2 dense layers with relu and sigmoid activation for binary classification.

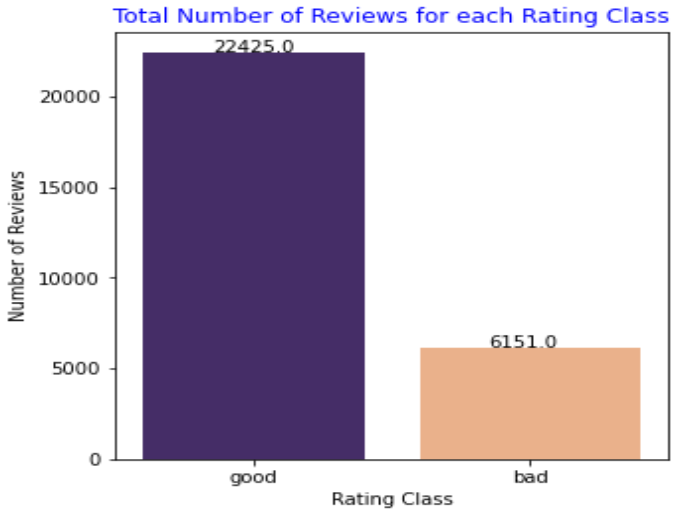
(b.) Hyperparameters:

i. Batch size: 32
ii. Optimizer: rmsprop
iii. Loss: Binary cross-entropy
iv. Epochs: 12

3. Results

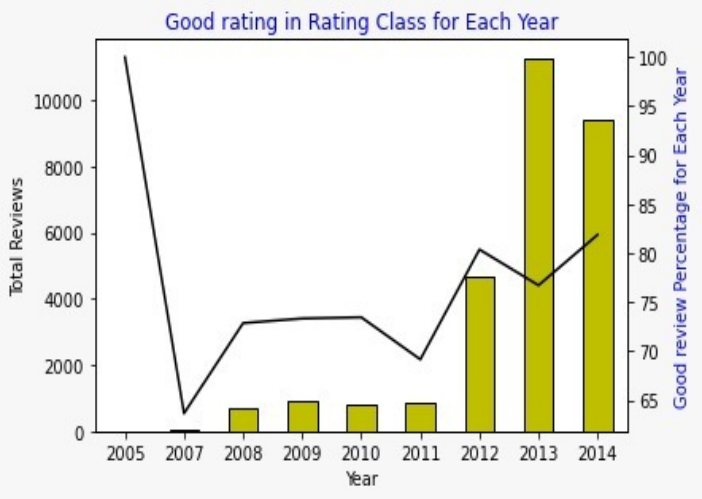
From the exploratory Data analysis, the following observations were made:

- **Target variable:**



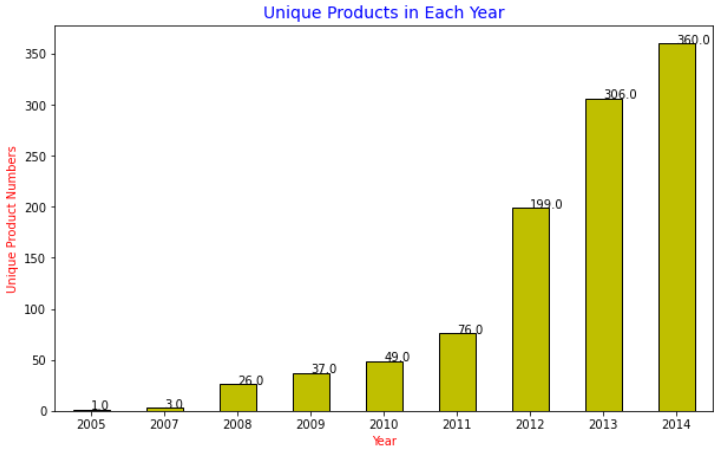
The reviews written by the customers ranged between 1 to 5 and customers seemed averagely satisfied with their purchases. 78.47% of the reviews were classified as "good", whereas 21.53% were classified as "bad".

- **Feature: “Year”**



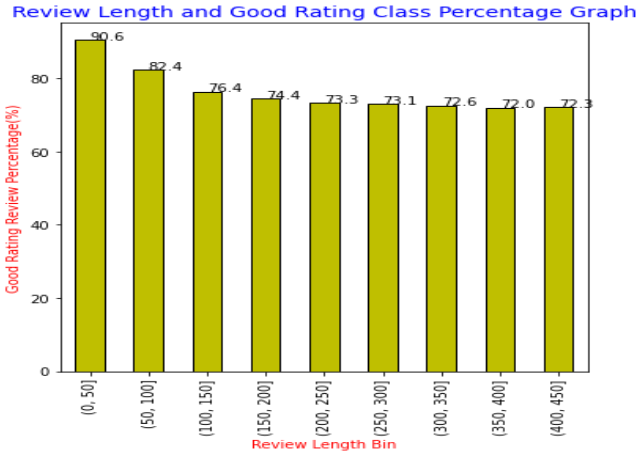
For the years 2008, 2009, 2010, 2011 the percentage of good ratings was around the range 65%. After 2011, good ratings' percentage is progressing over 80%. Before 2012, only 2009 also showed a slight increase in good ratings and a decline from 2010 to 2011. Besides those, 2007 has the lowest good ratings with less than 65% overall. As it might be seen in the graph, the overall good rating is progressing between 60% and 85% in beauty products.

- **Feature: “Product”**



We can notice in the data we have that the number of unique products have increased exponentially from 2005 to 2014. In 2005 we had one unique product, where in 2014 we had 360 unique products.

- **Feature: “Review Length”**



From the graph, we can see that when the review length is between 0 to 50 bin, the good rating percentage is the highest at 90.4%. And when the length of the review is between 350- 400 bin the good rating percentage is 72.0% and 400-450 bin having 72.3. So we can conclude that when the length of the review increases, the good rating percentage decreases slightly. The difference between a good review with shortest length and a good review with longest length is 18.3%. So insightfully, customers have written longer reviews when they are displeased with their purchase.

- **Feature: “Good rating words”**

The table shows the most common 50 words which belong to the good rating class. These words give us insight about what the products are and what kind of impression they have on the customers. For example, “Amazed” and “glad” words tell us that the products are appreciated in the reviews. Example 02: “Eczema”, and “Scar” tells us that the products are praised for covering them. Example 03: “Economical”, and “shipping” tells us that the products were reasonably priced and conveniently shipped to the customers. Example 04: “fresh” and ‘organic’‘ words tell us about the quality of products.

- **Feature: “Bad rating words”**

The table shows the most common 50 words which belong to the bad rating class. These words give us insights about what the products are and what kind of impression they have on the customers. For example, “Wax” tells us that products made of wax have a complaint. Example02: “description” tells us that the product does not suit or work as per the description of the product.

Due to the existence of class imbalance in the dataset, we are choosing F1 score as the performance metrics as accuracy would not give us a meaningful insight on the performance of the model. F1 score takes into account the class distribution and pays more attention to the minority class. The precision and recall helps us correctly detect the positive samples when the positive class is smaller. This is the reason we have employed F1 score (which is a harmonic average of precision and recall) as an evaluation metric.

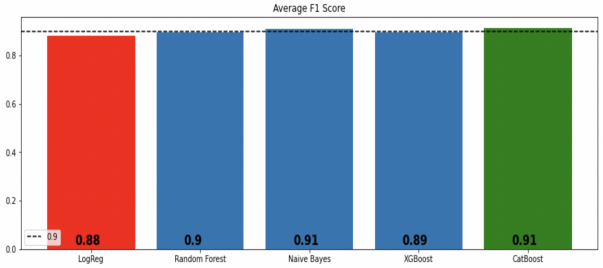
A. Models with SVD vectorizer

In order to see any changes in evaluation metrics, the most and least common 70 words were added to the stopwords list and models were applied. Once the cleaning is done, we then proceed with

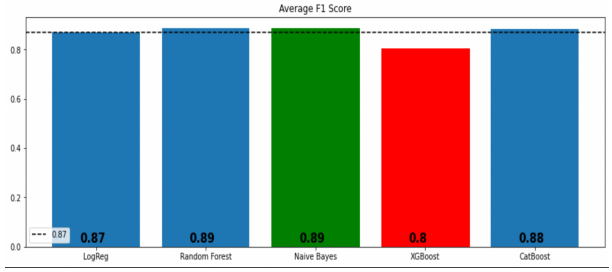
(i.) CountVectorizer, where we convert a collection of text documents to a vector of term/token counts. Moving further, we apply the models and the best model for CountVectorizer turned out to be Naive Bayes and CatBoost.

(ii.) CountVectorizer (SMOTE), here we use SMOTE technique as the data is imbalanced. Once the data has been balanced using SMOTE, we apply the models to it. Naive Bayes has produced better results here as well.

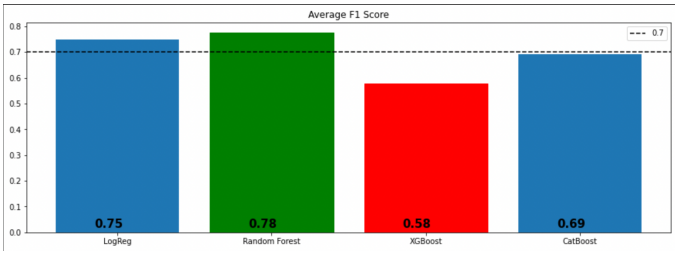
(iii.) Truncated SVD, where data matrices with the number of columns equal to the truncation are factored using Truncated SVD. In this, Random Forest has produced better results.



CountVectorizer



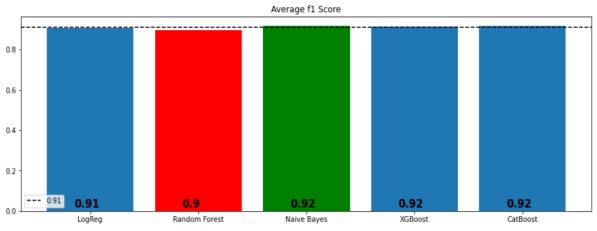
CountVectorizer(SMOTE)



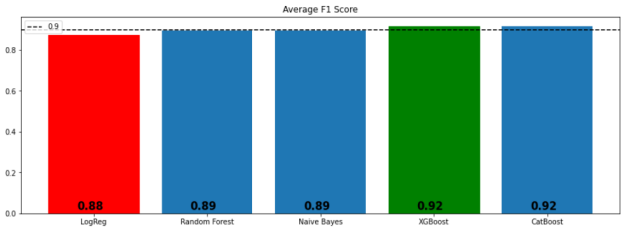
Truncated SVD

B. Models with CountVectorizer and TF-IDF

With CountVectorizer we have achieved a best f1 score of .92 with Naive Bayes and worst with .90 with random forest whereas withTF-IDF we have achieved a best f1 score of .92 with XGboost and worst of .88 with logistic regression.



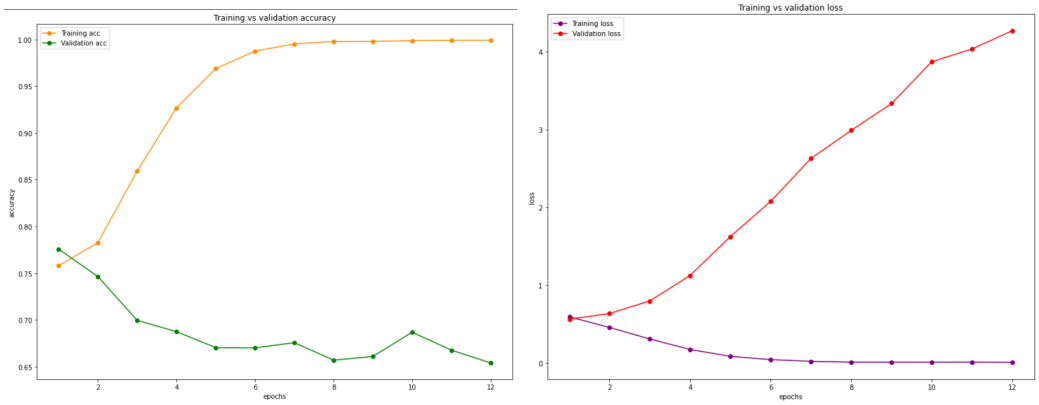
CountVectorizer



TF-IDF

C. Model with Word2Vec embeddings

Given a small number of Training sets, Variance is high and the model starts to quickly overfit. A good way to shuffle all the data and split into train and test sets then fitting the model. There are 2.7 million learning parameters in the model. Reducing the number of learning parameters will also help to tackle the over fitting. Using a simple cost and loss function will also help alongside regularization i.e., dropout.



4. Discussion

Our objective is to generate predictions for the review sentences. We believe we have achieved success in that realm and are reporting the results. These models, however, are not without issues. As this is a supervised learning task, the model is impacted by data distribution. We discovered that this is class imbalance data, with 20% data being in the minority class and the rest in the majority class. The challenge of working with imbalanced datasets is that most machine learning techniques will ignore, and in turn have poor performance on, the minority class, although typically it is performance on the minority class that is most important. Our approach to address imbalanced datasets is to oversample the minority class. The simplest approach involves duplicating examples in the minority class, although these examples don't add any new information to the model. Instead, new examples can be synthesized from the existing examples. This is a type of data augmentation for the minority class and is referred to as the Synthetic Minority Oversampling Technique, or SMOTE. As it is a class imbalance problem F1 score is a better evaluation metric than accuracy.

5. Conclusion

In this project, we predicted the rating category based on customer reviews. Before looking deeper into the model results, it was made clear that preparation of the dataset and feature engineering are just as critical as model construction. We applied

- Count Vector, TF-IDF, Word2Vec
- Classification Models and Simple Neural Network
- Adding most and least common words to CountVect, - SMOTE, - PCA + SMOTE - Truncated SVD + SMOTE

Naïve Bayes with Count Vectorizingproduced the best results with an f1 score of 0.924423. Adding most and least common words to the stopword list didn't have an impact on models' performance. Resampling technique also did not make a positive improvement of the model accuracy but decreased the model performance.

6. Acknowledgement

This project was completed for CS6120 under the instruction of Professor Uzair Ahmad.The TA's were always such a help, and we appreciate their support in every way. Their assistance was invaluable in helping our team get everything done in a very timely manner.

References

[1] Data link source:
http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/reviews_Beauty_10.json.gz

[2] https://www.analyticsvidhya.com/blog/2021/06/nlp-sentiment-analysis/

[3] https://www.geeksforgeeks.org/using-countvectorizer-to-extracting-features-from-text/

[4] https://towardsdatascience.com/word2vec-from-scratch-with-numpy-8786ddd49e72