Fantastic Food Finder : Rating Prediction& Sentiment Analysis on Amazon Food Reviews

Jasmijn Bleijlevens jasmijn.bleijlevens@hotmail.com
Lanie Preston laniepreston@gmail.com
Floor Kouwenberg floorkouwenberg@hotmail.nl

Abstract

As commerce moves from brick and mortar stores to online webshops, providing consumers with relevant information about products and providing merchants with accurate feedback is increasingly important. We have created a classifier that can predict the number of stars a food item will be given in an Amazon review based on textual analysis of review. Our program also indicates which aspects of a product, such as its packaging and flavor, are particularly positive and which are particularly negative, which is useful information for manufacturers and consumers alike. The predictions were made by training a Naive Bayes classifier on the number of stars that the review earned on a selection of Amazon food reviews in the Review.csv file that was part of a Kaggle library. We also counted the co-occurrences of nouns with particularly positive and negative adjectives, based on scores from NLTK's SentiWordNet, to find the relevant characteristics of the product. Ultimately, we created a Naive Bayes classifier that classifies the number of stars a review will receive with an accuracy of 77%. Our project aims to improve and facilitate the process of designing, adapting, and purchasing food products.

1 Introduction

Our project aims to answer two research questions: Is it possible to estimate the number of stars a food review will receive based on the language of the review? If so, can we isolate specific attributes of a product that contribute to its rating?

2 Related Works

We found several sources relating to sentiment analysis and machine learning prediction for online reviews that steered us in the right direction when beginning this project.

2.1 Text-Mining & Sentiment Analysis (Balasubramanian, 2018)

This was the first paper we read when brainstorming ideas for our project. Balasubramanian used text mining to find specific qualities about an item that were positive or negative so that manufacturers would be better informed about their product's performance. He achieved this through feature mining and sentiment analysis on the text of a review to provide specific words, like flavor or packaging, that were associated with positive or negative contexts. After we read this paper, we had the idea to examine textual co-occurrences and their SentiWordNet measures of words within these tuples; however, we decided to use them as features, since we planned on creating a classifier.

2.2 Polarity Extraction and Classification Using Naive Bayes, SVM, and Maximum Entropy (Rathora et al, 2017)

The second paper we read extracted Amazon reviews from an Amazon API, trained Naive Bayes, SVM, and Maximum Entropy classifiers on weighted unigrams, and used these classifiers to categorize reviews as either positive, neutral or negative. They found the SVM classifier to be the most accurate (81% accuracy), followed by the Naive Bayes Classifier (77% accuracy) and finally Maximum Entropy (70% accuracy). This paper followed most closely the work we did in our project. We chose to use a Naive Bayes classifier since we did not have experience working with an SVM classifier.

2.3 Categorical Classification of Reviews with SVM and the Maximum Entropy Algorithm (Yang et al, 2015)

Finally, in order to see an example of text-mining based review analysis that was not specific to Amazon, we read at this paper, in which researchers used classified reviews based on its matched the category from a pre-existing corpus to find the most informative reviews in a list of reviews about a single product. These researchers used SVM and the maximum entropy algorithm as their classifiers; since we did not learn how to implement these methods in the Machine Learning class we took last year, we chose not to use these classifiers, but we thought it significant that these researchers achieved an accuracy of more than 93% with their approach.

3 Data Collection

Our data set was obtained from this Kaggle repository: https://www.kaggle.com/snap/amazon-fine-food-reviews

The original data set, Reviews.csv, contains nearly 600,000 reviews and was too robust for our computers to run efficiently. We shuffled the data set to ensure we got an even distribution of reviews and took a slice of 10,000 reviews from the data set, which was the largest slice that we could analyze efficiently with Jupyter notebooks and upload to GitHub. We stored the processed data set in a pandas data frame.

4 Data set Description

The original data set contains nearly 600,000 reviews of fine foods from Amazon. For every review, 10 variables are visible in the data set. To identify the review and reviewer, the following things are present: the ID of the review, a time stamp of the review, the ID (combination of letters and numbers) of the product reviewed and the ID of the user who reviewed the product and their profile name. Then three scores are given: the number of stars (1-5), given by the reviewer themselves, and a helpfulness numerator and denominator, given by other clients of Amazon. Then the review itself is given in two parts, first the summary and then the full review.

The reviews are written between October 1999 and October 2012 on nearly 75,000 unique products by more than 250,000 unique users. 260 of those users wrote more than 50 reviews.

5 Methods & Main Algorithms

We first ran our data set through a pre-processing pipeline where we created an 80/20 train/test split, eliminated stopwords (except for negation), and converted characters to be uniformly lowercase. We lemmatized each word in each review using the nltk WordNet lemmatizer and saved the processed corpus in a pickle file.

We used a Naive Bayes classifier with the number of stars a review has as our classes. Since a review can receive between 1 and 5 stars, our classifier has five classes, one for each potential star. A Naive Bayes classifier was ideal for this purpose because it assumes a bag of words representation, which is easily constructed with the data that we have and because Naive Bayes returns a class based on a prediction made using a class' features.

We first made a classifier based on the raw frequencies of adjectives, adverbs and negations (Balasubramanian, 2018). After that, we tried training the model on other features to increase the accuracy. Firstly, we used tf-idf weighting instead of normal weighting for the unigrams because it does not only take a word's frequency in a review into account, but also it's frequency across the entire corpus.

Secondly, we trained a model based on the frequencies of co-occurrences within the corpus. We found a list of co-occurring words and the raw frequencies of these words within a span of 3 words to use as a features for our classifier. We then weighted these co-occurrences with their PLMI scores in a new Naive Bayes model. To increase accuracy even more, we also tried to multiply the frequency with the PLMI scores, to take both the scores and the frequencies into account.

The final model we created used average SentiWordNet positivity and negativity scores per word per review as its features.

6 Results & Findings

The features of the first model were the frequencies of words that appeared 10 or more times in the entire data set. This gave an accuracy of 65%. We wanted to improve accuracy and tried different features and different weighting schemes. We used this accuracy metric as our baseline accuracy. Notably, all models we tested improve on the chance accuracy of 0.2, which is the estimated accuracy of random assignment with five classes.

The tf-idf weighting with the raw frequency of adjectives, adverbs and negations reduced our accuracy to 57%. This can be explained by the definition of tf-idf. The adjectives, adverbs and negation that we used as our target words probably occur across most, if not all, documents in our dataset. However, some might occur more often in one rating class than the other. Those word frequencies could still be useful to distinguish classes. However, tf-idf will still reduce the weight of these distinguished words, since they occur in all documents. Therefore tf-idf could still reduce the overall accuracy of our classifier.

The model based on the frequencies of co-occurrences gave an accuracy of 77%, precision of 65% and recall of 70%. We tried to improve these results using the PLMI scores of the co-occurrences. We first used the PLMI score alone, instead of the co-occurrence raw frequencies, in the feature matrix. Then we tried to use the PLMI score as a weighting for the co-occurrence frequencies by multiply it by the frequencies in the feature matrix. Both gave a lower accuracy. Respectively, the accuracy was 71% and 72%, which is lower than using the raw frequencies alone. However, the precision of both models was higher: 72%. So even although the accuracy decreases, the precision does increase, meaning that we had fewer false positives when using PLMI as a weighting.

The model based on the positivity and negativity scores of a review did not increase our accuracy any further. We had plotted the probability distribution of the scores per class and found that although you could distinguish the different classes easily, there was not an adequate level of difference across the classes for this to work as a proper feature (see figure 1 and figure 2, pictured to the right). We trained Naive Bayes model on these two features and got an accuracy of 61%. This is still better than chance, however we already found better models to predict the stars of reviews with.

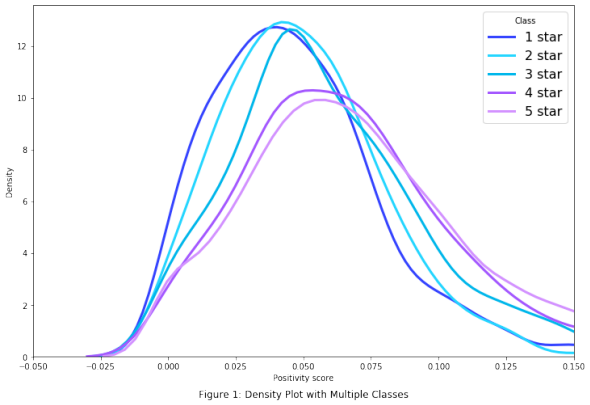

Figure 1: Density Plot with Multiple Classes

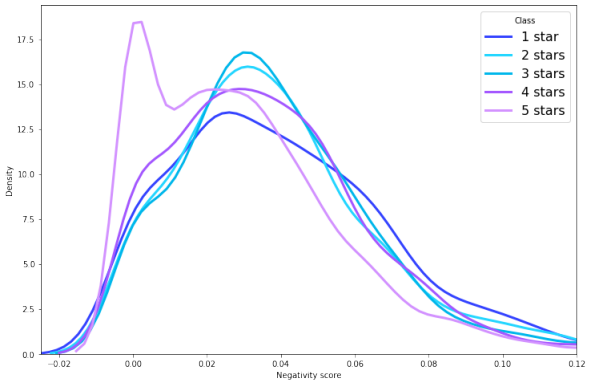

Figure 2: Density Plot with Multiple Classes

7 Conclusions

In terms of further research, several sources we looked at found SVM to be superior classifiers for this task than Naive Bayes Classifiers. We were not familiar with SVM classifiers, so we did not implement one for this project, but with more time and the opportunity to familiarize ourselves with their algorithms, we would be interested to see how an SVM classifier could potentially improve our accuracy.

In particular, using a SVM could solve the issue we had where the feature independence assumption necessary for Naive Bayes classification. In our scenario, this assumption is highly unlikely, because words do not exist in isolation, and their contexts usually have implications on other words and phrases, which acted as our features.

Ultimately, the most effective feature for classification was the raw frequencies of co-occurrences in the reviews. We think this is because co-occurrences demonstrate a stronger relationship between words than simple raw frequencies, because this adds more contextual information to our model and does not underfit the data as much as a unigram model. We thought about additionally trying a trigram model to account for this, but we believed that the amount of data that we were capable of working with given our laptop capacity was too sparse to create a good trigram model. We are not certain why weighting the co-ocurrences with the PLMI scores decreased the accuracy rather than increasing it; we hypothesized that this was a symptom of the Naive Bayes assumption that features are independent, but we are not certain. Average positivity and negativity per

class were too similar per class, as can be seen in our density plot, to be an effective feature.

Our best accuracy was a score of 0.77, which is roughly in line with the accuracies found by the researchers in our papers that used Naive Bayes Classifiers (Rathora et al, 2017). When we used Naive Bayes classifiers in Machine Learning and Information Lab assignments, we managed to get accuracies of around 0.9, and we had some hypotheses as to why our accuracy was not closer to this value. First, in the dataset, 260 users wrote more than 50 reviews, indicating that many reviewers write multiple reviews for different products. Since people's language tends to be distinctly individualistic even when writing reviews about very different products, it is possible that the language between reviews authored by the same users is quite similar. If we were to continue working on this project in the future, we would also extract the user ids from the csv file and, when filtering our slice of the dataset, ensure than no two reviews have the same user id. We also had a very difficult time handling negation in the text (meaning that reviews containing statements like "it was not very good" or "I will not buy it again" were potentially misclassified). We thought about using a trigram model for negation, or some if tests to check if there is negation, but we did not find a solution where data sparsity did not become an issue or that handled all cases of negation.

As a whole, we discovered that there are many reviews which are difficult to classify because the language of online reviews tends to be quite generic and non-product-specific. For example, people tend to say "great flavor" or "bad packaging" across a spectrum of star ratings, additionally without tying these compliments and complaints to specific aspects about the product itself. We do not believe this is an issue that can be fixed on our end; rather, Amazon could encourage users to write reviews that are more specific. Perhaps, rather than just having one large text box that users fill in, Amazon can split reviews into separate boxes (for example, a box that says "What is the best feature about this product?", followed by a box that says "What do you dislike about this product?". This would encourage users to put more informative language within the body of the review. We could also implement a more aggressive pre-processing pipeline, if our laptops had a greater capacity for running notebooks. Ultimately, this task was much more complicated than we anticipated, but we believe with some tweaks, this classifier could become very informative to users and manufacturers alike.

References

1.AS Rathora, A Agarwalb and P. Dimric. (2018). Comparative Study of Machine Learning Approaches for Amazon Reviews. *International Conference on Computational Intelligence and Data Science (ICCIDS 2018)* pp. 1552-1561.

2. S Balasubramanian. (2017). Msc. Thesis: Text Mining on Amazon Reviews to Extract Feature-Based Feedback *Department of Computer Science, California State University Sacramento.*

3. Q Yang, P Feng and Z Cheng (2015). Clothing Product Reviews Mining Based on Machine Learning *International Journal of Online and Biomedical Engineering, 11*(9), 71-76.