Predicting Helpful Online Product Reviews

Daniel Miller, Hector Castillo, Justin Stoner, Kelsie Box
Computer Science Department
University of New Mexico
Alubquerque, NM, USA
{dmiller23, hcastillomartinez, jstoner, boxk} @unm.edu

ABSTRACT

The rise of online retail over the last twenty years has caused consumers to increasingly rely on the product reviews of others to make informed shopping choices. This development combined with the massive volume of product reviews that are being posted online has led to several issues in the online shopping ecosystem. A vast majority of the reviews posted online are not helpful to consumers for a several reasons. Many lack new and relevant information related to the product being reviewed; others are highly opinionated and lack subjectivity; some reviews are fake and often factually inaccurate which harms the consumer.

We are interested in creating methods to better regulate this growing online space, so that consumers can save time and money while shopping online. In this paper, we use the Amazon Customer Reviews dataset to analyze specific features that a helpful review is likely to contain. We also propose a method of assigning reviews an initial helpfulness rating to combat the biased compounding effects of Amazon's review listing system. Our model was able to predict the number of helpful votes a review is likely to receive with 70% accuracy.

KEYWORDS

big data; predicting review helpfulness; spark

1 INTRODUCTION

The rise of online retail over the last twenty years has led to rapid growth in the number of product reviews posted on the internet. Consumers increasingly rely on the product reviews of others to make informed shopping choices. However, the high volume of reviews being posted makes it difficult for consumers to get an accurate picture of the product they are researching. Consumers and businesses both benefit when methods of determining review quality improve.

One possible way to determine the quality of a review is rating how helpful it is to other users. Many shopping sites display reviews in order of how many helpful votes a review receives from other users. However, the number of helpful votes a review is likely to get is largely a function of time, and many reviews that are potentially useful to customers get buried at the bottom of the page. Additionally, older reviews contain content that may not be relevant any more. This causes the online shopping experience to be confusing for consumers because it is difficult to know which reviews are factually accurate and best represent the product being researched. This causes consumers to spend exorbitant amounts of time sifting through reviews instead of shopping. Online marketplaces are in need of good metrics for determining the helpfulness of their product reviews so that reviews can be ranked, saving valuable time for their customers.

There have been several successful attempts at predicting review helpfulness. One way of determining review helpfulness is to identify what structural factors (e.g. length and readability) contribute to a review's helpfulness. For example, Mudambi and Schuff [5] analyzed 1,600 reviews from Amazon.com across six products and found that review extremity and depth were contributing factors to review helpfulness. While this research gives valuable insight into what structural factors make a review helpful, its limited use of review data could prove inaccurate for the entire review population.

Another popular approach in predicting review helpfulness involves detecting the emotion of the review. This method is termed emotion-based helpfulness prediction. Martin and Pu [3] extracted the emotionality from the review text and used supervised machine learning methods to derive a prediction of emotion-based review helpfulness. They found that emotion-based methods outperformed the structure-based approach by up to 9%.

In this paper, we combine these two approaches with big data techniques in order to cluster reviews into helpfulness tiers. Reviews are clustered based on metrics that quantify both review structure and emotionality to determine what aspects contribute to its perceived helpfulness. We also use a supervised machine learning algorithm to predict how many helpful votes a given review is likely to receive based solely on its text body. We then use the predicted number of helpful votes determined by the neural network to assign reviews an initial helpfulness rating that can be used to offset the bias in favor of older reviews that is currently present in the online product review ecosystem.

2 MOTIVATION

Sifting through many reviews is time consuming for consumers, but it is necessary for establishing trust in the online marketplace. In this research, we attempt to develop methods that reduce the amount of time consumers spend reading reviews while keeping the same level of trust in their online purchases.

3 DATA

We used the AWS Amazon Review dataset which was compiled by Amazon from 1998 to 2015[1]. This dataset contains over 130 million product reviews from over one million Amazon users. It has 15 columns containing information such as the product title and id, review body, rating, and metadata such as the date. Most reviews from the data set have no helpful votes and most of the users have only voted once. This makes sense because of how people generally review and rate content, where the majority of people do not rate

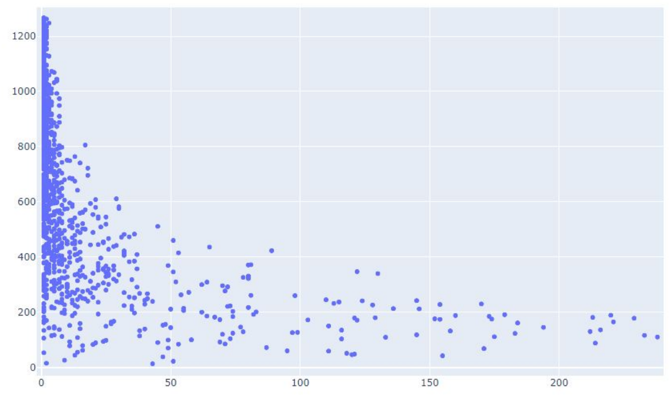

Figure 1: x axis: number of helpful votes, y axis: number of reviews with that many helpful votes

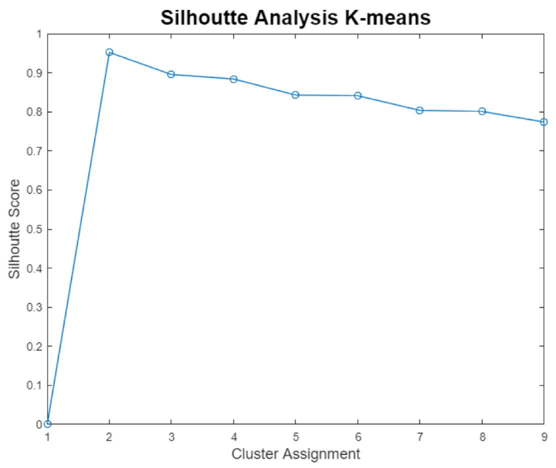

Figure 2: Silhouette Analysis

or review products. Unfortunately, users also do not appear to vote on helpful reviews very often. The vast majority of reviews have no helpful votes. Reviews also generally gain more helpful votes as the age because more people see them.

Given that we use the helpful votes column as a the label to the review body for our neural net, this creates a problem as most of the data is effectively unlabeled. In some cases, the lack of helpful votes means can mean that the review is truly unhelpful, so we could not simply discard them. The skewing was so bad that the distribution of helpful votes appeared to be exponential. This heavily biased our initial results towards reviews with no votes.

As illustrated in figure 1, the vast majority of reviews have few or no helpful votes. Unfortunately for us, we are interested in the learning the characteristics of the outliers in this graph, which are few in number.

4 METHODOLOGIES

Our methodologies for this project were centered around predicting how helpful a review is likely to be. However, this turned out to be more difficult than expected. The data varied greatly and posed a challenge for us in figuring out a clustering approach and a neural net that could accurately account for that variety. Additionally, because of the large amount of reviews with no helpful votes, both the clustering and neural net were skewed towards zero.

4.1 Clustering: Approach 1

In order to get the data into a form in which clustering over the set could be performed, a method of parsing the reviews needed to be developed. The data was parsed into a matrix in which each row of the matrix corresponded to the data of one review. The values from the matrix used for clustering were helpful votes, sentiment rating, subjectivity rating, and word count. To get the sentiment and subjectivity rating, we used the Textblob library [14]. Other values in the matrix include sentences, adjectives, nouns, pronouns, verbs, and adverbs.

Our first attempt at clustering was to use the scikit learn k-means clustering library [15]. This method of clustering turned out to be unfeasible as it took over twenty hours for the algorithm to finish over the smallest data file in our set.

We then switch over to using the PySpark MLlib library [2]. Additionally, in an attempt to reduce the amount of time the algorithm had to run, we used only a sample of our data set in which each row of our data set had a 35% chance of being included in the sample set. With the PySpark MLlib library, we implemented a k-means clustering algorithm that used the silhouette method to understand how well each point was assigned to its cluster. Out metric for how well the point was assigned was it's closeness to a neighboring cluster. Figure 2 shows the silhouette score plotted against the number of clusters. The peak of the plot indicates what the value of k should be in the k-means clustering algorithm. In our case, the peak presented at two, so we knew that two clusters would be ideal for our data.

The clusters were formed using the helpful votes, sentiment rating, subjectivity rating, and word count of each review. The newly formed clusters raised an important problem with our data set that we had not noticed previously—approximately 98% of our data had no helpful votes at all. This caused our results to be skewed and for the majority of the data points to be clustered next to zero, while the second cluster accounted for the outliers with 500 or more helpful votes.

4.2 Clustering: Approach 2

In an attempt to produce results that weren't skewed, the probability of a row being included in the sample set needed to be modified. Two probability functions were used. The first probability function was the following:

$$P = \tanh\left(\log_{10}(1) + 0.15\right) + \frac{r}{R} \tag{1}$$

with r being the number of rows in the current data file and R being the total number of rows in all the data files. This probability function was used for any of the reviews that had no helpful votes. The second probability function was used for all other votes that

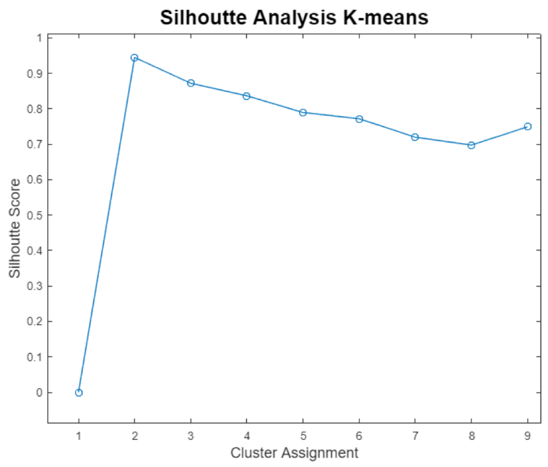

Figure 3: Silhouette Analysis with Probability Functions

did have helpful votes. The equation is as follows:

$$P = \tanh\left(\log_{10}(n) + 0.15\right) + \frac{r}{R} \tag{2}$$

with n being the number of helpful votes the review had. The use of these probability functions effectively normalized the distribution of helpful votes among the sample set. Using the silhouette method again, we found the ideal number of clusters to be two. Figure 3 depicts the silhouette analysis using the new probability functions.

The resulting clusters were still skewed, but less than they were previously. We then decided to try clustering with three clusters in an attempt to further analyze the reviews. By using three clusters, we found that the reviews with no helpful votes were being clustered with the outliers. An outlier in this case is defined as a review with more than 1000 helpful votes. Further analysis of the reviews revealed that the timestamps for those reviews with no helpful votes were from 2015. These reviews were too new at the time Amazon sampled them to gather any helpful votes. Once we understood this, we were better able to analyze our resulting clusters.

4.3 Neural Net Methodology

With clustering, we were attempting to define what type of reviews Amazon buyers found helpful. Because re-clustering after a new review is added is prohibitively expensive, we decided to build and train a neural net to classify the reviews are part of one of a few tiers defined by how many helpful votes a review has. At first however, we wanted to make the neural net preform regression and makes guesses as to how many helpful votes a review has.

However, the first step in creating the neural net was transforming the data into something the neural net could take in. Our first attempt at doing this was the bag of words algorithm. This algorithm works by converting text into an array in which a one is present in the array when the corresponding word at that index is present in the sentence, and a zero is present otherwise. This method of extracting features from the body of the review presented two problems: the growth of the array is relative to the variety of the words and the context the words were used in was lost. These problems caused our design to become inaccurate and to take an abhorrent amount of time to train, so we decided to move on to a new method.

The next attempt at getting the Amazon data into a format the neural net could take in was to use word embedding and convolution layers. Word embedding works by iteratively assigning vectors in a multitude of dimensions to words with the idea that similar words will be assigned similar vectors. This method has three settings we can tune in order to get the best embedding fit for our data: embedding dimensions, max length of a sequence, and word count. The ability to customize these settings allowed us to get the data into a better form for the neural nets than the bag of words method allowed.

Now that the data was into a form that the neural net could handle, a new problem arose. The sum of the vocabulary across all of the reviews was so large and diverse that regression was difficult and the accuracy almost nonexistent. The first part of our solution to this was to focus the neural net on only one product category file at a time. So one set of weights and a tokenizer were produced for each file processed. The idea being that the way the English language was used in a single category would be more specific to that topic. This would allow the neural net to better understand the input without blowing up the dimensions need to produce the embeddings.

The second part of the solution to the accuracy problem was to switch from regression to classification. Various tiers based on the number of helpful votes a review had were created. The number and size of the tiers shifted around as we tuned the neural network. Six tiers worked fairly well because they roughly reflected the distribution of the data. A new tier would start every five helpful votes until the count exceeded 20, at which point all further values would be assigned to the same tier. At first glance, we thought this method had been successful as the accuracy of the neural nets increased into the 90% range. However, upon further examination of these classifications, we realized that the large number of tier zero reviews had skewed the training, causing the neural net to classify everything as a tier zero review.

This proved to be a massive problem. There were far too many reviews with no helpful votes relative to everything else. To negate this effect, we first reduced the number of tiers to three (less than or equal to 4 = tier 0, less than or equal to 16 = tier 1, everything else = tier 2), the goal being that would could achieve better control over the amount of reviews in each tier for the purpose of balancing. Second, we limited the number of reviews to no more than three times the number of tier 2 reviews; which helped in reducing the skewing.

In addition to this, we threw out the convolution layers and replaced them with 128 units of lstm and a dense layer with 50 neurons (Figure 4). While the problem was the data, these changes helped negate this and gave about three zones of accuracy.

5 RESULTS

We had some success in predicting the number of helpful votes a review should have. The clustering allowed us to get a template of what an ideal review should look like, which is described in the tables below in section 5.1. Additionally, we were hoping to find

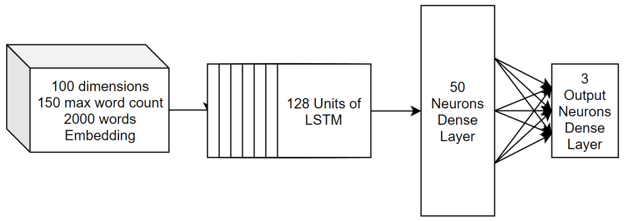

Figure 4: Final neural net design

some sort of correlation between the ideal number of clusters and the number of nodes used by the neural network for classification. This turned out not to be the case and we ended up using two clusters and three nodes for the neural network.

5.1 Clustering Results

We found that the reviews that Amazon customers found the most helpful were those that were longer in length, neutral, and semi-objective. This result is demonstrated in the following table, which contains an overview of what the ideal helpful review looks like when two clusters were used.

Table 1: Two Clusters

	Helpful Votes	Sentiment Rating	Subjectivity Rating	Word Count	Sentence Count
avg	27.783	0.142	0.494	670.752	29.367

The ideal helpful review has approximately 670 words, 29 sentences, and 27 helpful votes. Additionally, as we increased the size of the clusters, we found that the number of helpful votes inversely proportional to both the sentiment rating and subjectivity rating. When three clusters were used, the ideal helpful review looks as follows in the table below.

Table 2: Three Clusters

	Helpful Votes	Sentiment Rating	Subjectivity Rating	Word Count	Sentence Count
avg	43.122	0.138	0.489	1121.853	47.228

The average review length found when using three clusters was approximately 1122 words, almost double the word length from Table 1. The number of helpful votes from Table 1 to Table 2 is also close to doubled. The difference in the number of helpful votes, word count, and sentence count from Table 1 to Table 2 demonstrates that Amazon users tended to think that longer reviews were more helpful than their shorter counterparts. Additionally, the slight decrease in the Sentiment Rating and the Subjectivity Rating from Table 1 to Table 2 demonstrates that users preferred neutral, semi-object reviews.

5.2 Neural Net Results

With our neural net, we produced a set of weights along with a tokenizer for each file in the data set. Most of the weights either had approximately 71% or approximately 80% accuracy. Only one file, mobile electronics, gave us an even lower accuracy of approximately 67% accuracy. Additionally, we developed a program to load the weights that were produced by the training of the neural net. The values from the following table are outputs from the program that represent the training accuracy of three different files.

Neural Net Accuracy

	Major Appliances	Gift Cards	Mobile Electronics
Validation Accuracy	0.7532	0.7740	0.6674

6 DISCUSSION

This research uses a combination of structural and emotionality analysis to determine what features helpful reviews are likely to contain. We found that helpful reviews tend to be longer in word and sentence count, with neutral subjectivity and slightly positive sentiment ratings. This suggests that consumers find thorough reviews with slightly positive phrasing to be the most helpful when researching prospective products online.

We believe that these models can be extended from product reviews to reviews for all online services, in order to increase the efficiency of the current review ecosystem. The models used in this research need to be updated to take different parameters if they are to be used in novel situations, but we believe that this application is significant and merits further investigation.

It is important to note that our results may not generally apply to users who don't usually vote on reviews. This phenomenon is comparable to social media, where users generally lie on one end of the spectrum in terms of how many posts they like. Future work should sample the entire population of online shoppers in order to get a clearer picture of the space. However, since machine learning based approaches require substantial amounts of data, novel methods may need to be developed to accurately depict the entire online product review space.

Ensuring that high quality data is used to develop future models is paramount to their success. We realized near the tail end of our project that while the Amazon Customer Reviews dataset is massive, with over 130 million reviews from over one million customers, we did not have enough reviews that received high numbers of helpful votes to feed into our algorithms. We also found out that the reviews were not evenly distributed through time. The dataset was skewed towards a larger number of recent reviews. Since the number of helpful votes reviews get is largely a function of time, many reviews in the dataset had not accumulated enough votes to reflect their true helpfulness. This means that the majority of our data was unlabeled because there was no way for our neural network to know whether a review had no votes because it was truly unhelpful, or it simply had not been posted for long enough to accumulate votes.

We see now that using helpful votes as a metric of review helpfulness is not unbiased. We essentially tried to solve a problem by leveraging data that was biased by the same problem we were trying to solve. This skewed our results. Future work should explore new ways of assessing review helpfulness. For example, many forums take into account the credibility of posters when determining the quality of their posts. We believe that this idea of author credibility can be extended to the product review ecosystem because a reviewer that has reviewed many products is likely to be more trustworthy and credible than a user that has posted few reviews. However, this approach is not without fault and shouldn't be used exclusively. Many reviewers are compensated for reviewing products for new online suppliers because it is necessary for sellers to build a review base before a consumers trust can be gained. We believe that if used properly, assessing reviewer credibility when determining review helpfulness could be used to improve future research in this space.

It is also important to note that, just like other automated processes, human due-diligence must be used to ensure that the systems of interest are being modeled accurately without introducing biases into solutions. This is difficult when using machine learning algorithms because it is difficult to understand why these algorithms learn the information that they do. It is important to understand how human biases can affect experimental results, so that we can verify the correctness of our solutions. Developing algorithms that are free of bias is an area of active research and the research in this paper will benefit immensely from advancements in the aforementioned field.

7 ACKNOWLEDGMENT

We would like to express our gratitude for the guidance given to us by Dr. Estrada of the University of New Mexico Computer Science department.

REFERENCES

[1] Amazon customer reviews dataset. URL https://s3.amazonaws.com/amazon-reviews-pds/readme.html.

[2] Nikolaos Korfiatis, Elena GarcíA-Bariocanal, and Salvador SáNchez-Alonso. Evaluating content quality and helpfulness of online product reviews: The interplay of review helpfulness vs. review content. *Electronic Commerce Research and Applications*, 11(3):205–217, 2012.

[3] Lionel Martin and Pearl Pu. Prediction of helpful reviews using emotions extraction. In *Twenty-Eighth AAAI conference on artificial intelligence*, 2014.

[4] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[5] Susan M Mudambi and David Schuff. What makes a helpful review? a study of customer reviews on amazon. com. *MIS quarterly*, 34(1):185–200, 2010.

[6] Jyoti Prakash Singh, Seda Irani, Nripendra P Rana, Yogesh K Dwivedi, Sunil Saumya, and Pradeep Kumar Roy. Predicting the "helpfulness" of online consumer reviews. *Journal of Business Research*, 70:346–355, 2017.

[7] Stephen Skalicky. Was this analysis helpful? a genre analysis of the amazon. com discourse community and its "most helpful" product reviews. *Discourse, Context & Media*, 2(2):84–93, 2013.

[8] Anon. 10.7. glob - Unix style pathname pattern expansion. Retrieved December 11, 2019 from https://docs.python.org/2 /library/glob.html

[9] Anon. Clustering. Retrieved December 11, 2019 from https://spark.apache.org/dc
clustering.html

[10] Anon. Keras: The Python Deep Learning library. Retrieved December 11, 2019 from https://keras.io/

[11] Anon. NumPy. Retrieved December 11, 2019 from https://numpy.org/

[12] Anon. pickle - Python object serialization. Retrieved December 11, 2019 from https://docs.python.org/3/library/pickle.html

[13] Anon. Python Data Analysis Library. Retrieved December 11, 2019 from https://pandas.pydata.org/

[14] Anon. Simplified Text Processing. Retrieved December 11, 2019 from https://textblob.readthedocs.io/en/dev/

[15] Anon. sklearn.cluster.KMeans. Retrieved December 11, 2019 from https://scikit-learn.org/stable/modules/generated/sklearn .cluster.KMeans.html