Amazon Review Data Sentiment Analysis - P42

Anant Gadodia
200364928
NC State University
agadodi@ncsu.edu
github.ncsu.edu/agadodi

Gabriel Francis
200370013
NC State University
gfranci@ncsu.edu
github.ncsu.edu/gfranci

Shreya Someswar Karra
200422257
NC State University
sskarra@ncsu.edu
github.ncsu.edu/sskarra

Abstract

We have implemented a project where will be studied and mined on the dataset containing product reviews posted on Amazon.com. The data we used was pre-processed in order to make the textual part of the dataset more easy to handle by our models. We then performed a preliminary analysis to get an idea of the trends and relations present. This information was then used to perform sentiment analysis using a variety of techniques and then we introduce a new method that gave positive results. It can be found here.
Our github repository with the code used to implement the methods given below can be found **here**

1 Background and Introduction

Reviews are an integral part of the buying process. The enable the buyer to make an informed decision on whether or not to buy a particular product without having to directly interact with said product. An online shopping platform such an Amazon.com, relies heavily on reviews and ratings and these can be the difference between a customer choosing one product over another extremely similar product. At times a review may hold a sentiment more true to the reviews emotions on the product as opposed to just the star rating provided. We attempt to create a classifier that can accurately predict the sentiment present in a review.

Attributes such as the average the number of helpful votes received and whether it was a verified purchase can also help determine the degree of emotion.

In order to perform the above analysis and then create a model, chose to use a kaggle based Amazon Product Review Dataset. The dataset has approximately 113M reviews encompassing 36 product categories. The reviews are stored in separate tsv files based on the product category. For our analysis and modelling we will be using various samples of the data that are at most consisting of 3M reviews. There are 15 columns in each file which are mentioned in Table 1.
Keeping in mind the limited computing resources available to us, we subsets of 360K, 1M and 3M rows from the files and have based our results on these sample datasets.

2 Method

2.1 Preprocessing All Data

Some steps taken while loading the data:

Field	Description
marketplace	2 letter Country Code of the marketplace where the review was written
customer_id	Random identifier unique for each author
review_id	Unique id of each review
product_id	Unique identifier for each product for which reviews are written
product_title	Title of the product
product_category	The broad category used to classify products
star_rating	Rating given by the reviewer on a scale of 1-5
helpful_votes	The number of helpful votes received by the review
total_votes	The total votes of the review
vine	Flag to show if the review was a part of the Vine program
verified_purchase	Flag to show if it reviewer actually purchased the product
review_headline	The title of the review
review_body	The content of the review
review_date	Date the review was written

Table 1: Data Fields

- Fill null values: The product category column had some null values which were replaced with the category they belonged to
- Converting review_date to a date data type.
- Some entries had some escape characters in the text which prevented python from reading them correctly. We skipped these entries.
- We checked all dates to make sure we did not have any garbage values
- We checked all numeric columns namely star_rating, total_votes, helpful_votes to make sure the values were all within the range.
- We made sure that there were no duplicate reviews.

2.2 Preprocessing Textual Data

2.2.1 Stop word removal

As a part of preprocessing textual data, we have to remove stop words. Stop words are those words that are commonly used in the English language such as a, an, the, but etc. These words while useful when communicating, they do not hold much information about the topic being discussed and thus can be removed.

2.2.2 Normalization: Stemming and Lemmatization

Normalization is the process of reducing a word to it's root, i.e. without any tense applied to it. This process is useful as instead of having multiple words having the same meaning but just being of different tense, we have one word to represent them all. This allows our model to be more easily trained and reduces the probability of over-fitting.

Stemming is when normalization is done simply using rudimentary rules such as removing the 'ing' at the end of words in a continuous tense. However, this process sometimes results in incorrect values as words like 'sing' would have the letters 'ing' removed from it even though it is not in a continuous tense.

Lemmitization is a more advanced form of normalization where contextual information along with a predefined dictionary is used to reduce a word to it's root.

We shall focus mainly in lemmitization as that has shown to be the better method for normalization.[9]

2.3 Text Representation

There are many methods for the representation of textual data such as 1-Hot encoding, n-gram models, vector semantics and bag of words. For our project we have decided to focus on using bag of words and word2Vec representations as these allow the meaning of the word to have some value which can be useful in our task of judging the sentiment of the review.

2.3.1 Bag of words

The bag of words representation is one of the simplest representations of text. It is simply a set of all the words present in the corpus, be it a sentence or a large collection of documents.

2.3.2 Word2vec

In this form of text representation, each word is represented by a vector of some length based on the data set we use. Each word has a corresponding position in n-dimensional space such that as we move away from that position words become less and less similar to the word in consideration. That is, each word in the space is surrounded by words with a similar meaning. This allows us to easily conduct basic arithmetic operations such as addition and subtraction on words to get a new word. For example, if we take the word 'Boy' and subtract the word 'child' we could get the word 'male' or 'man'.

We will use both representations to analyse the sentiment of a review.

2.4 Sentiment Analysis

Sentiment analysis is a text analysis method which is used to detect the polarity i.e whether the given text is positive, negative or neutral. It is used to measure the attitude, sentiments, and emotions of a text.

2.4.1 VADER

VADER (Valence Aware Dictionary and sEntiment Reasoner)[5][6] is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media.
A sentiment lexicon is a list of lexical features (e.g., words) which are generally labeled according to their semantic orientation as either positive or negative.
VADER not only tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is.
VADER's ***SentimentIntensityAnalyzer()*** takes in a string and returns a dictionary of scores in each of four categories: negative, neutral, positive, compound (computed from the other three).
The Compound score is a metric that calculates the sum of all the lexicon ratings which have been normalized between -1(most extreme negative) and +1 (most extreme positive). It uses a dictionary of terms that it can evaluate.

- Negations - a modifier that reverses the meaning of a phrase ("not great").
- Contractions - negations, but more complex ("wasn't great").
- Punctuation - increased intensity ("It's great!!!").
- Slang - variations of slang words such as "kinda".

It's even able to understand acronyms and emojis.
The scoring is a ratio of the proportion for text that falls into each category. Language is not black and white, so it is rare to see a completely positive or a completely negative score.

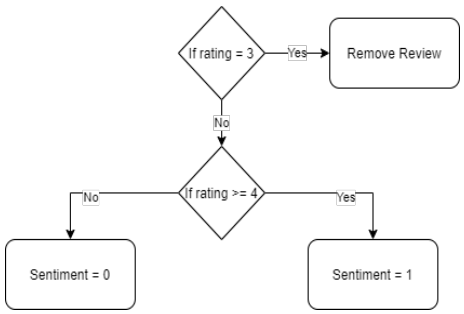

Figure 1: Binary Logistic Regression Model Pre-processing

2.4.2 Logistic Regression

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. For our project we used the Binary Logistic Regression model. In this model, the categorical response has only two 2 possible outcomes. We use the flowchart in Figure 1 to assign a sentiment to each review.

As the data is unbalanced and heavily skewed towards positive reviews, thus we balance the data using Random Undersampling, a method that randomly selects and removes samples from the majority class till the number of samples in both the classes are equal.
The Logistic Regression Model is then applied on both datasets.

2.4.3 Word Similarity using Word2Vec

In the word2vec representation of our reviews, we know that words with similar meaning are closer together as compared to words with different meanings.

The similarity of words in the review to positive and negative words may be used to determine whether a review has a positive or negative sentiment. We will see operations can be done on these similarity values to get a better result and then determine if it was a success or not.

3 Experiment Setup

3.1 Text Preprocessing and Representation

The pandas package is used as it provides an excellent data structure and provides the functions to manipulate that data structure in the required ways.

We use the NLTK package to tokenize and lemmitize our reviews and use the gensim package to remove stop words and generate our word2vec representation.

3.1.1 Sentiment Analysis using Word Similarity with Word2Vec

We first use random under sampling to balance our data. After that, the textual data (the review body) is run through our preprocessing pipeline and a word2vec representation of our corpus is created.

Three methods of classification using the word2vec representation are used; each a slight variation of the same idea in order to find the optimal result.

Maximum Average Similarity

The similarity of each word in the review from a positive word and a negative word is taken. The average of both these similarities are taken for the whole sentence and then if the average similarity

Product Category	# of products	Avg Rating	# of Reviews
Apparel	66,226	3.94	99,997
Automotive	65,164	4.28	100,000
Baby	30,406	4.23	100,000
Beauty	50,018	4.23	99,998
Books	65,055	4.21	123,938
Camera	32,188	4.17	100,000
Digital_Ebook_Purchase	49,111	4.31	99,995
Digital_Music_Purchase	66,625	4.71	99,995
Digital_Software	2,983	3.55	99,992
Digital_Video_Download	12,248	4.22	48,292
Music	2,628	4.45	43,422

Table 2: Study of Product Categories

to positive word is greater than the average similarity to negative word, then we classify the review as positive and else as negative.

Maximum Similarity Count

Each word in a sentence is checked to see whether it is closer to the positive word or the negative word. A count is kept of the number of words closer to either side and then the side with the maximum number of words attached to it is chosen.

Similarity Ratio with Threshold

The similarity of each word in the review from a positive word and a negative word is taken. The average of both these similarities are taken and the ratio of the average positive word similarity and the average negative word similarity.

3.1.2 Metrics

As all the models we will design are classifiers or different sorts, we shall be using classification accuracy as our main metric. No class takes precedent over the other and both have equal importance making accuracy the best metric for our case.

4 Results

4.1 Basic Statistics

The average rating on these reviews was $4.14/5$

The lowest star rating was given to Digital Software Category(3.55).
The highest star rating was given to Digital Music Category(4.71)

5 Star reviews made up 63% of the total reviews we studied while 4 star reviews accounted for 16% of the total reviews.

A basic study of the categories below shows us the number of products, reviews and average rating for each category in the table 2 below. We also check the review with ratings 1-3 for the Electronics product category to find out the most commonly used words and they were: *Sound, Work, Quality, Battery, Cable, Time*
We also checked the review lengths with product categories and found that the ones with the longest lengths were products that were subjective like *books and apparel* while those with shorter lengths were products where the user has a fair idea of the product before purchase like *music, movies, gift cards*.

Word	Similar word 1	Distance	Similar word 2	Distance	Similar word 3	Distance
good	'decent'	0.757	'great'	0.735	'well'	0.717
bad	'terrible'	0.745	'horrible'	0.691	'awful'	0.654
music	'song'	0.859	'track'	0.808	'piano'	0.799
quality	'construction'	0.628	'workmanship'	0.619	expectation'	0.600
best	'worst'	0.658	'well'	0.655	'amaze'	0.642

Table 3: Word Similarity Comparison

star_rating	review_body	scores
1	Very bad quality not like in pictures	neg': 0.541, 'neu': 0.459, 'pos': 0.0, 'compound': -0.7098
2	Broke after not much use.	neg': 0.412, 'neu': 0.588, 'pos': 0.0, 'compound': -0.4215
3	Works really well once it's paired. The problem is with connecting	neg': 0.071, 'neu': 0.866, 'pos': 0.063, 'compound': -0.079

Table 4: Snapshot of Sentiment Analysis Results

4.2 Text Preprocessing

Stop words have been successfully removed using NLTK. Below is a comparison of the 101th sentence of our corpus before and after removing stop words.

Raw sentence
I don't know if the problem is with my bose speakers or this cable, the sound quality is not that great.

Stop words removed sentence
know problem bose speaker cable sound quality great

4.3 Text Representation

Table 3 shows the three most similar words to five out of twenty words that are commonly found in the reviews under consideration after preprocessing and we can see from the data that the word2vec model mostly works extremely well in finding words with similar meaning with a few exceptions.

However, we can also see that words are similar to just slight modifications of itself. This makes the model unnecessarily complicated and thus must be removed using normalization.

4.4 Sentiment Analysis

4.4.1 VADER

After performing the sentiment analysis, we get the negative, positive, neutral and compound polarity scores for all the records. We have shown a few examples of this in Table 4. Similarly 1M records from the 'amazon_reviews_us_Electronics_v1_00.tsv' file were classified.
After generating the polarity index, we generated a scatter plot for the star_ratings and the compound polarity which is a combination of the positive, negative and neutral polarity.

4.4.2 Logistic Regression

After performing sentimental analysis using Binary Logistic regression on the unbalanced model the accuracy achieved is **0.931**.
The accuracy achieved for the balanced data using Binary Logistic regression is **0.542**. The confusion matrix for the balanced data is given in Figure 3.

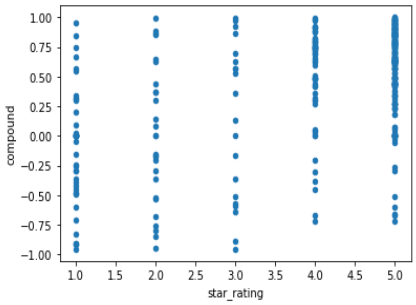

Figure 2: Star Rating vs Review Sentiment

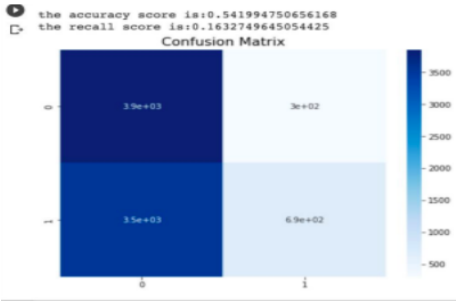

Figure 3: Confusion Matrix for Logistic Regression using Balanced Data

4.4.3 Word Distance with Threshold

Positive Word	Negative Word	Mixed DataSet (99K+) Acc %	Electronics Dataset (1M+) Acc %
Good	Bad	66.3	**73.1**
Best	Worst	56.7	64.8
Positive	Negative	54.1	60
Awesome	Terrible	**66.9**	70.9

Table 5: Accuracies for two samples using different positive and negative words using Maximum Average Similarity method

Positive Word	Negative Word	Mixed DataSet (99K+) Acc %	Electronics Dataset (1M+) Acc %
Good	Bad	62.3	**68.9**
Best	Worst	55.1	61.7
Positive	Negative	55.2	59.8
Awesome	Terrible	**63.4**	66.3

Table 6: Accuracies for two samples using different positive and negative words using Maximum Similarity Count method

Based on Table 5 and Table 6, we see that for large dataset we get best results when we use the words 'good' and 'bad' as the positive and negative words respectively. We then worked on our next technqiue which gave us the results found in Table 7.

Threshold	Mixed Dataset (99K) Acc %	Electronics Dataset (3M) Acc %
0.5	52.3	64.7
0.7	56.6	68.4
0.9	63.8	71.7
1	66.9	73
1.1	69.1	74
1.15	69.8	74.3
1.2	70.1	74.4
1.25	**70.4**	**74.5**
1.3	70.2	74.4
1.35	69.9	74.2
1.4	69.3	74
1.45	68.7	73.6

Table 7: The table show the accuracy for two samples in consideration using 'good' as the positive word and 'bad' as the negative word by using Similarity Ratio with Threshold method for different thresholds

5 Conclusion

The text preprocessing used has been shown to be good as it gives us just the lemmitized words that hold some meaning of the review without any other noise.

As the corpus we are using is large, the word2vec representation of words present is large and accurate. This enables us to predict the sentiment of reviews with decent accuracy.

After comparing all the methods of sentiment analysis that we implemented, we find that using the word2vec representation to calculate the ratio of similarities of words of a sentences with the positive word 'good' and the negative word 'bad' and then taking all sentences below the threshold of **1.25** to be negative and anything above that to be positive gives us the best accuracy of **74.5%**

Sentiment analysis using an LSTM[7] proved to be too difficult and thus could not be implemented.

6 References

[1] Vaisakh Nambiar (2019) Text Analysis of Amazon Customer Reviews, *https://medium.com/analytics-vidhya/text-analysis-of-amazon-customer-reviews-b4fcf0663216*.

[2] Shubham Singh (2019) NLP Essentials: Removing Stopwords and Performing Text Normalization using NLTK and spaCy in Python, *https://www.analyticsvidhya.com/blog/2019/08/how-to-remove-stopwords-text-normalization-nltk-spacy-gensim-python/*

[3] Aditya Beri (2020) SENTIMENTAL ANALYSIS USING VADER interpretation and classification of emotions, *https://towardsdatascience.com/sentimental-analysis-using-vader-a3415fef7664*

[4] C. Hutto, Eric Gilbert (2014) VADER: A Parsimonious Rule-Based Model for Sen timent Analysis of Social Media Text, *https://ojs.aaai.org/index.php/ICWSM/article/view/14550/14399*

[5] Aryan Bajaj (2021) Can Python understand human feelings through words? – A brief intro to NLP and VADER Sentiment Analysis, *https://www.analyticsvidhya.com/blog/2021/06/vader-for-sentiment-analysis/*

[6] Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

[7] Samarth Agrawal (2019) Sentiment Analysis using LSTM (Step-by-Step Tutorial), *https://towardsdatascience.com/sentiment-analysis-using-lstm-step-by-step-50d074f09948*

[8] Lin Zhang, Kun Hua, Honggang Wang, Guanqun Qian, Li Zhang, (2014) Sentiment Analysis on Reviews of Mobile Users, Procedia Computer Science, Volume 34, 2014, Pages 458-465, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2014.07.013. *https://www.sciencedirect.com/science/article/pii/S1877050914008680*

[9] Balakrishnan, Vimala & Ethel, Lloyd-Yemoh. (2014). Stemming and Lemmatization: A Comparison of Retrieval Performances. Lecture Notes on Software Engineering. 2. 262-267. 10.7763/LNSE.2014.V2.134.