# 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 propose a project where will be studying and mining on the data-set containing product reviews posted on Amazon.com. It can be found at https://www.kaggle.com/cynthiarempel/amazon-us-customer-reviews-dataset. Our github repository with the code used to implement the methods given below can be found at https://github.ncsu.edu/agadodi/engr-ALDA-fall2021-P42

## 1 Background and Introduction

The interpretation of text is a highly complex and specialized field that continues to expand rapidly. For our project, we chose to focus on understanding text at a sentimental level, i.e. the base emotion felt by the author of that piece of text. A place we would easily find such a corpus would be in reviews where reviewers would rarely ever have a neutral view on the product, and instead would have a very specific view on it, be it either positive or negative. It is our goal to create a model to detect, with a decent accuracy, the degree of the emotion felt by the author.

We chose to use a kaggle based Amazon Product Review Dataset as it is a vast dataset with many datapoints consisting of the kind of data we need. This would allow us to carry out various different techniques for analysis and classification and in the end generate a good model. Other 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.
It has approximately 113M reviews encompassing 36 product categories. The reviews are stored in separate tsv files based on the product category.
There are 15 columns in each file which are:

- 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

Keeping in mind the limited computing resources available to us, we Selected 1M rows from the files and have based our preliminary studies on this sample dataset.

# 2 Method

## 2.1 Preprocessing All Data

Some steps taken while loading the data:

- 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 continous 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 be attempting both these techniques from various packages and use the one that fits best.

## 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 word2Vec and bag of words representations as these allow the meaning of the word to have some value which can be useful in our task of judging the sentiment and helpfulness of a 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 this representation to analyse the words present in reviews using sentiment analysis to predict the ratings given to each product.

### 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, evaluations, attitudes, and emotions of a text.
We used a trained NLTK lexicon called **VADER** (Valence Aware Dictionary for Sentiment Reasoning) which is a model used for text sentiment analysis that is sensitive to both polarity (positive/negative) and intensity (strength) of emotion. It is available in the NLTK package and can be applied directly to unlabeled text data. It relies on a dictionary that maps lexical features to emotion intensities known as sentiment scores. The sentiment score of a text can be obtained by summing up the intensity of each word in the text.
We won't try to determine if a sentence is objective or subjective, fact or opinion. Rather, we care only if the text expresses a positive, negative or neutral opinion.

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).

## 3 Experiment Setup

### 3.1 Text Preprocessing

The pandas package is used as it provides an excellent data structure along with functions that allow us to manipulate the data in simple ways such as selecting only the required set of attributes or removing null values.

We also use the NLTK package to tokenize our sentences and to remove the stop words to get more accurate results. It is important to remove stop words, i.e. words commonly found in English that do not have any real impact on the topic discussed in the corpus. If we run our model using these words, they will essentially act as noise and give us faulty results.

We will also try to use the gensim package for normalization of words.

### 3.2 Text Representation

We will be using the gensim package to generate our word2vec representation. For this midway report we are only using the the 'amazon_reviews_us_Electronics_v1_00.tsv' file which consists of over 3 million reviews.

## 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 below:

| 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 1: Study of Product Categories

### 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**
Really cool design and it goes up really easy. The amount of adjustment is really nice as well. It's been up for almost two months with our 55 inch TV and hasn't moved as all unless I'm the one to move it.

**Stop words removed sentence**
Really cool design goes easy. The adjustment nice well. It's months 55 inch TV hasn't moved I'm it.

### 4.3 Text Representation

The below table shows the three most similar words to five out of twenty words that are commonly found in the reviews under consideration after preprocessing.

| Word | Similar word 1 | Distance | Similar word 2 | Distance | Similar word 3 | Distance |
|---|---|---|---|---|---|---|
| **great** | 'fantastic' | 0.848 | 'awesome' | 0.836 | 'good' | 0.82 |
| **sound** | 'sounds' | 0.77 | 'sounding' | 0.672 | 'imaging' | 0.625 |
| **good** | 'decent' | 0.896 | 'great' | 0.82 | 'excellent' | 0.731 |
| **works** | 'worked' | 0.8 | 'performs' | 0.653 | functioned' | 0.628 |
| **quality** | 'fidelity' | 0.758 | 'quality..' | 0.659 | 'quality.' | 0.618 |

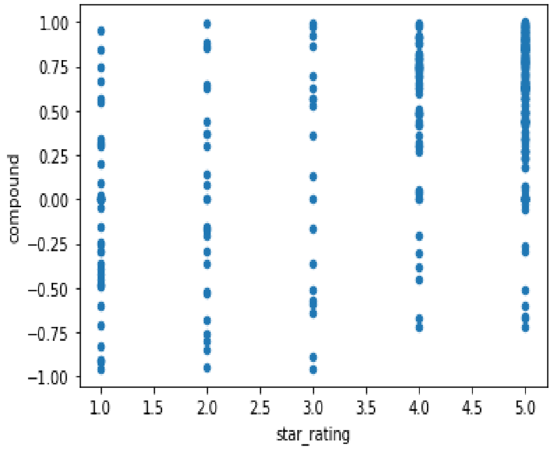
We can see from the data that the word2vec model works extremely well in finding words with similar meaning.

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 Results after performing Sentiment Analysis

After performing the sentiment analysis, we get the negative, positive, neutral and compound polarity scores for all the records. For e.g. below given table is a small part of the results that we received after performing the sentiment analysis.

| 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 |



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 basically the sum of the positive, negative and neutral polarity that has been normalized.

## 5 Conclusion

The text representation technique used yielded results that are less that satisfactory. A lot of noise such as some stop words, punctuation, different forms of the same word and gibberish due to html formatting remained. This adversely affected our conversion of the data into the word2vec form.

Also, the model that we used has detected the positive reviews correctly when matched to the ratings however for the ratings less than 3 the polarity has not been computed accurately. This shows that the model that we used is not as tuned as we’d like it to be.

Therefore our future goal is to develop a more robust system for conversion of text into word2vec form and a more accurate model is needed that can detect and classify all the reviews correctly with respect to their ratings.

## 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 Sentiment 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/*