Amazon Kindle Books Reviews Analysis

Big Data Course Project Proposal

Hien Pham Thi Thu, $^{\mathrm{SID}}$ 2022120203
Dept. Computer Science
UCLab, Chung-Ang University
Seoul, South Korea

***Abstract*—Buying goods online nowadays has become an essential part of the modern lifestyle. As one of the big pioneers in the domain, Amazon is aware of customers' demands and hence provides a helpful rating system that helps buyers get more clear about products and services before making any decision. This project will propose a solution that could enhance the quality of the rating system by trying to get customers to approach more helpful and genuine reviews as possible.**

***Index Terms*—Natural Language Processing, Classification, Sentiment Analysis**

I. Introduction

Amazon is renowned for being the largest e-Commerce platform all over the world. There is a fact that nearly 90% of Amazon shoppers check product reviews before they purchase, which even gets higher when it comes to the COVID-19 pandemic. Since customers are not able to go to the shop to physically check the product, they have to depend on reviews of other users who already bought it. Although the Amazon review system works quite well, sometimes it still displays poor-quality reviews on the top of the forums that can easily affect customers and make them give up on the product. In which, a poor-quality review can either be unhelpful or a spam review. To tackle the problem, this proposal will answer the below questions:

- What factors make people rate a review as helpful or unhelpful?
- How to know a review is helpful or not?

Once answering these questions, Amazon could find a way to show all genuine and helpful reviews on top so that would be more convincing people to buy their products and eventually increase their sales volume.

II. Dataset

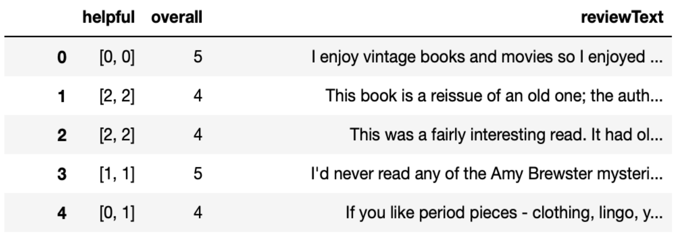
For this project, I will use the data set provided by Kaggle platform, which is named *Amazon reviews: Kindle Store Category* [1].

- The data set size: 550MB.
- The data set format: csv, json.

This dataset is taken from Amazon product data, Julian McAuley, UCSD website [2].

III. Proposed Solution

Considering this data set, there would be three columns that we should focus on, which are: *helpful*, *overall*, and *reviewText*. Before diving in deeper, let's take a brief look at the value of these columns:



	helpful	overall	reviewText
0	[0, 0]	5	I enjoy vintage books and movies so I enjoyed ...
1	[2, 2]	4	This book is a reissue of an old one; the auth...
2	[2, 2]	4	This was a fairly interesting read. It had ol...
3	[1, 1]	5	I'd never read any of the Amy Brewster mysteri...
4	[0, 1]	4	If you like period pieces - clothing, lingo, y...

Fig. 1. Raw data.

A. Data Exploration

As you can see in the data, the *helpful* column gives us the number of helpful votes over the total number of votes. Let's say for a particular review, if this ratio is greater than 0.5, this will be a helpful review. Therefore, I add one more column named *is_helpful* with the value of **1** indicating that a review is helpful and the value of **0** for the reverse case.

After that, I remove all reviews that have less than 10 votes in total since they do not make a lot of sense in our big data set. Then I plot a bar chart to see if there any correlation between the helpfulness and the stars. "Fig. 2" suggests that reviews with higher overall rate are likely to be more helpful than their counterparts with lower rate. Moreover, our data set is quite unbalanced since the majority of considered reviews are helpful.

B. Data Pre-processing

The field *reviewText* obviously is the most important factor to the review's helpfulness. However, they are text that are written by people without any constraint. Therefore, in order to deal with this kind of data, we first must pre-process it to get proper text reviews on which analysis can be performed.

I will perform basis of Natural Language Processing (NLP) on these data following the workflow in "Fig. 3"

- Step 1. Drop Null/Duplicates: Drop all the *reviewText* that is null/na using *dropna()* function. Drop all the duplicated records using *distinct()* function.

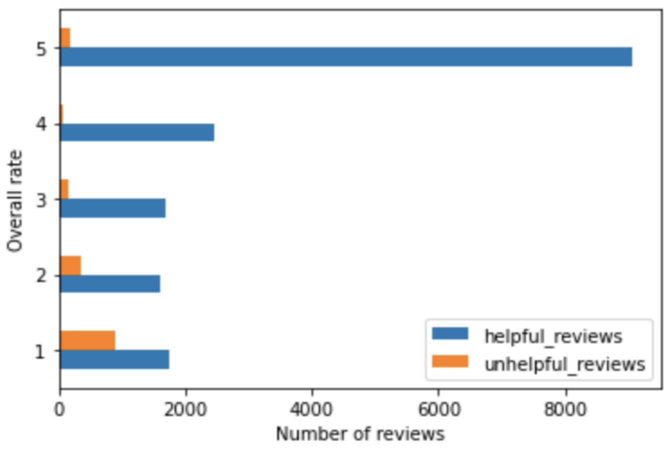


Fig. 2. Helpfulness by overall stars rate.

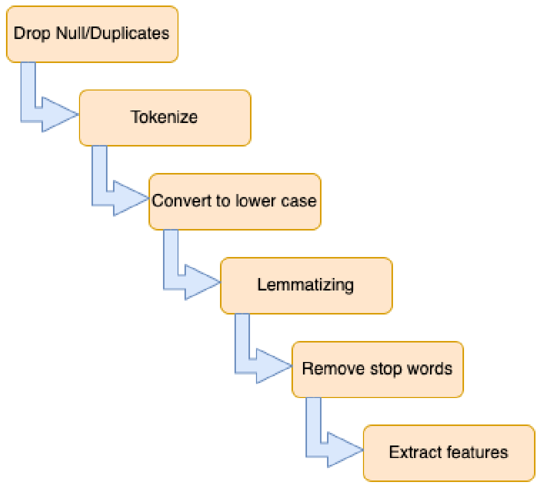


Fig. 3. Text data pre-processing flow.

- Step 2. Tokenize: Break the whole *reviewText* into individual words using *Tokenizer* annotator (which is then used in spark pipeline).
- Step 3. Convert to lower case: All the words should be normalized into lowercase by using *Normalizer* annotator. This will help to reduce the vector space size when put into machine learning models.
- Step 4. Lemmatizing: This is to return the base or root form of a word. I choose to use lemmatizing over stemming because it is more accurate although it is computationally expensive. This process will need a dictionary, so I use the pre-trained model supported by *spark-nlp* library called *LemmatizerModel*.
- Step 5. Remove stop words: Words such as *I*, *the*, *a*, *has*, *etc* do not carry much information and thus should be removed from the analysis. This will be done by using *StopWordsCleaner* annotator.
- Step 6. Extract features: Words need to be converted to numeric vectors to be able to feed into a machine learning model. There are actually multiple ways to do it, I choose to apply TF-IDF Vectorizer, which is a statistical measure used to find how important a word is to document in a collection or corpus.

All of these steps will be supported mostly by the *spark-nlp*, which is a library for Natural Language Processing built on top of Apache Spark.

C. Models

Coming back to our main problem, which is to answer those questions given in the introduction part.

For the first question, we already knew that the rating stars do have an influence on helpfulness. So this information and the vectorized representation of *reviewText* will be features fed into models to answer the second question.

To predict if a review is helpful or not, I will use a binary classification model. I will perform both *Naive Bayes* and *Logistic Regression* to see which algorithm would give a better result. Data will be split as 80% for training and 20% for testing.

To increase the accuracy of the model, some noise from our data set should be removed. Assuming that there must be some fake reviews in our data, which will worsen our model's performance. I will try to find these by doing as follow:

- Classify score ratings of reviews as Positive, Negative, and Neural.
- Perform sentiment analysis on the review text to get the overall tone and compare it to the tone based on score ratings. Let's say a mismatch between these results will tell us that a review is not genuine and thus being labelled as false.
- Do some statistics to find out which users have the most false reviews and then manually analyze these to decide whether they should be dropped from our data set or not.
- Retrain the model whenever dropping some data.
- Repeat the process until getting a better result.

The overall flow will start from *Drop fake news* to *Model Training* to *Model Testing* and then repeat it until getting what we expect.

IV. Conclusion

The above is my proposed solution to the given topic of the Amazon review system. I will perform all of the tasks mainly using *spark-nlp* and *mllib* built on top of Apache Spark. Along with the process of doing the project, there is a slight chance that I may try different algorithms to get the best result in case those that I proposed do not give me what I expect. But in general, the main topic will not change and so do my purpose is to answer those questions that might bring some insights into the Amazon business.

References

[1] https://www.kaggle.com/datasets/bharadwaj6/kindle-reviews
[2] http://jmcauley.ucsd.edu/data/amazon/