**Report: Analysis of Amazon Food Reviews**

# **Overview**

This project aims to analyze Amazon food reviews using text analysis techniques. It aims to gain any insights possible into the sentiment of reviews and further identify any patterns or trends in the data. This sentiment analysis is performed using the Natural Language Toolkit library (NLTK). We use this method since it provides us a quick, efficient and effective method with which we can assess and analyze large number of reviews.

# **Text Data**

The dataset in this project is from the Stanford Network Analysis Project. It consists of over 500,000 food reviews from Amazon. It is sourced as a csv file and contains reviews from Oct 1999 to Oct 2012. It contains over 260 users who have more than 50 reviews. The Reviews.csv file is pulled from the corresponding SQLite table named Reviews in database.sqlite. Database.sqlite contains the table Reviews. Reviews include various information from product and user information to ratings and plain text review. It also includes reviews from all other Amazon categories. There’s a total of 256,059 users reviewing over 74,000 products.

The dataset contains various columns, especially 'Score', which happens to be the rating given by the reviewer, and 'Text', which contains the actual text of the review.

We begin the analysis by reading the data into a Pandas dataFrame. Since the dataset is quite large, we have decided to take a smaller subset. A subset of 50,000 reviews is selected for analysis.

The 'Score' feature is represents the distribution of ratings. A majority of the reviews are 5 stars and we can see this from the bar graph that we obtain (*Fig 1*).

We then examine the ‘Text’ feature by extracting a sample review from it and then tokenizing it using the NLTK library. This process of tokenization involves breaking down the text into individual words or ‘tokens’. These resulting tokens that we obtain are then further used to perform a frequency distribution analysis. This allows us to identify the most common words. But since there contain many words with insignificant meanings, we further remove stopwords. Stopwords are commonly used words that carry no significant meaning. Thus after removing stopwords, we obtain a more meaningful frequency distribution (*Fig 2*).

We further perform a process called stemming which allows us to chop words to their root word. For example, ‘cook’, ‘cooked’ and ‘cooking’ come from the root word ‘cook’. Stemming allows us to process words and bring them down to their root word. Furthermore, we perform lemmatization to reduce these words to their base word or linguistically correct lemmas. For example, ‘better’ and ‘good’ are similar enough words to be grouped or identified together.

We generate POS tags or Part of Speech tags to further understand what kind of words are present. We can then take those tags and put them into entities.

# **Method**

The sentiment analysis is performed using the SentimentIntensityAnalyzer from the NLTK library. This approach uses a bag-of-words approach to determine the sentiment of a given text. Using positivity, negativity, neutral and compound scores, the SentimentIntensityAnalyzer assigns a sentiment polarity score to each text, that represents an overall sentiment.

Each review of the dataframe is processed through the SentimentIntensityAnalyzer so that we can analyze the entire dataset. We store the results of the positivity, negativity, neutrality, and compound scores which are calculated into a dictionary for further analysis. This is then merged into the original dataset by first being converted into a Pandas dataframe.

# **Evaluation and Findings**

Since the sentiment analysis results are merged with the original dataset, this allows us to perform a comprehensive analysis of the reviews. We find a relationship between the review scores and sentiment scores which is then visualized using bar plots (*Fig 3, 4, 5, 6*).

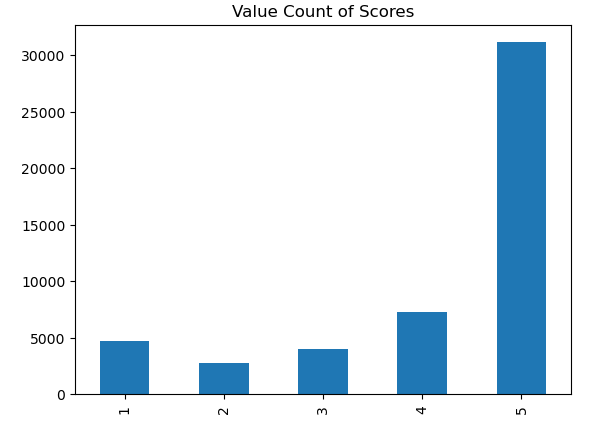
The bar plot of compound sentiment scores against the review scores shows a general trend of higher sentiment scores for higher review scores. This indicates a positive correlation between the overall sentiment of the reviews and the rating given by the reviewers (*Fig 3*).

Further analysis is performed by examining the positivity, negativity, and neutrality scores against the review scores. The bar plots demonstrate that as the review scores increase, the positivity scores tend to increase, while the negativity scores decrease. This suggests that higher-rated reviews generally have a more positive sentiment, while lower-rated reviews tend to have a more negative sentiment (*Fig 4, 5*).

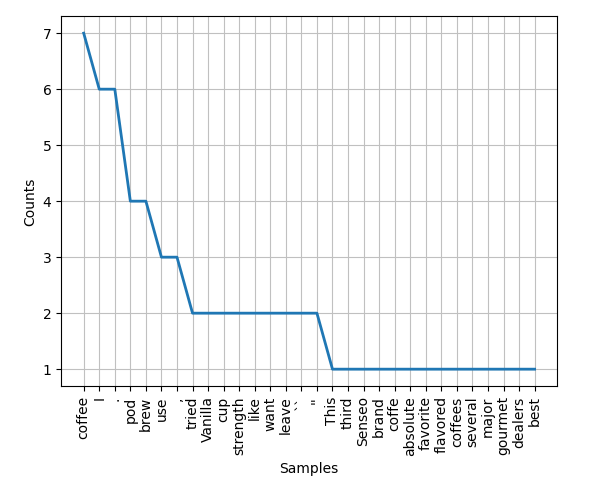
In conclusion, the analysis of Amazon food reviews using sentiment analysis techniques reveals interesting patterns and insights. Higher-rated reviews tend to have more positive sentiment, while lower-rated reviews have a higher negativity score. This demonstrates the effectiveness of sentiment analysis in capturing the sentiment expressed in the text data. These findings can provide valuable insights for businesses and consumers in understanding the sentiment associated with different review ratings.

The analysis conducted in this project showcases the potential of text analysis techniques to extract valuable information from large datasets of textual data. By leveraging the power of NLTK and sentiment analysis, businesses can gain deeper insights into customer sentiment and make informed decisions to improve their products and services based on customer feedback.

# APPENDIX



*Fig 1*



*Fig 2*

A picture containing text, screenshot, diagram, rectangle

Description automatically generated

*Fig 3*

A picture containing text, screenshot, rectangle, diagram

Description automatically generated

*Fig 4*

A picture containing text, screenshot, diagram, rectangle

Description automatically generated

*Fig 5*

A picture containing text, screenshot, colorfulness, rectangle

Description automatically generated

*Fig 6*

**Text Analysis Project**

Project to analyse amazon food reviews

Step 1: Setup and import libraries

#Setup and import libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import nltk

from nltk.sentiment import SentimentIntensityAnalyzer

Read the reviews from the reviews dataset.

#Read the data in

df = pd.read\_csv('Reviews.csv')

We can now check whether the data has been read in, and how it looks.

#Check read data

df.head()

We still do not know how big the dataset is.

print(df.shape)

We could probably decrease our sample size to 50000 for now

df = df.head(50000)

print(df.shape)

df.head()

Explore the data to see how it looks like

df['Score']

This is our what the score looks like which is a fundamental feature. Perhaps we could also check how many times each value appears on Score.

value\_c = df['Score'].value\_counts()

value\_c

Most of the values listed are 5 stars.

We can plot this on a bar graph to visualize how the data is broken down.

#Sort the scores

sorted\_scores = value\_c.sort\_index()

sorted\_scores

bar\_graph = sorted\_scores.plot(title = 'Value Count of Scores', kind='bar')

bar\_graph

We can now start looking at the subject of what the reviewers are writing. Let's have a look at one of the texts.

text1 = df['Text'][9473]

print(text1)

This seems to be a very positive review. We can now try to extract the words and tokenize it using NLTK.

tokens = nltk.word\_tokenize(text1)

Using these tokens we can now identify what part of speech they belong to using NLTK POS.

Let's have a look at how the tokens variable looks and then further into what it contains using POS.

tokens

Frequency Distribution

Here, we can check for frequency of our tokens.

from nltk.probability import FreqDist

freq\_dist = FreqDist(tokens)

print(freq\_dist)

This gives us an object that contains information of samples with total outcomes. Let's now see what the most common words are.

freq\_dist.most\_common(5)

Let's plot and visualize the frequency distribution.

freq\_dist.plot(30, cumulative = False)

plt.show()

The most common words seem to be very insignificant/neutral words that don't give us much information. Such as "a", "is", "the", or "I".

Let's identify the stopwords from the text.

from nltk.corpus import stopwords

stop\_words = set(stopwords.words("english"))

print(stop\_words)

Let's now remove stopwords from the data

filtered = []

for token in tokens:

if token not in stop\_words:

filtered.append(token)

print("Tokens: ", tokens[:10])

print("Filtered: ", filtered[:10])

We can see how much our data has been filtered in this process. Let's perform our frequency distribution again.

freq\_dist = FreqDist(filtered)

print(freq\_dist)

freq\_dist.most\_common(5)

freq\_dist.plot(30, cumulative = False)

plt.show()

We now have a better frequency distribution that has removed some of the insignificant words from our text analysis.

Stemming

Now let's perform some Stemming here. This involves the chopping of words to their root word. This is done by chopping off their derivational affixes.

from nltk.stem import PorterStemmer

from nltk.tokenize import sent\_tokenize, word\_tokenize

stemmer = PorterStemmer()

stems = []

for word in filtered:

stems.append(stemmer.stem(word))

print("Filtered: ", filtered)

print("Stemmed: ", stems)

We can notice how a lot of our words have lost a few letters in the ending because of stemming. This allows us to group words of similar roots into one.

But we can still do better by lemmatization which would reduce these words to their base word or linguistically correct lemmas.

from nltk.stem.wordnet import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

lem\_words = []

for word in filtered:

lem\_words.append(lemmatizer.lemmatize(word))

print(lem\_words)

tags = nltk.pos\_tag(tokens)

tags

We can now take these tags and put them into entities.

entities = nltk.chunk.ne\_chunk(tags)

print(entities)

Sentiment Analysis in a Bag of Words Approach

Create a Sentiment Analyzer object.

sia = SentimentIntensityAnalyzer()

sia

We can run this on any text to check the sentiment of the words. Let's test it before we proceed.

sia.polarity\_scores('This is a beautiful garden')

From the results, we can see how negativity is set to 0 while positivity is detected to be pretty high. We can also now see what the key value pairs of this dictionary look like for us to analyze our results later.

sia.polarity\_scores('This is a horrible garden')

This is another example on how our Analyzer seems to do a pretty good job in identifying the positivity and negativity of the text. Let's now run it on the review we extracted from the dataset earlier. The review was pretty positive, meaning we need to expect a positive result from the analyzer.

sia.polarity\_scores(text1)

The result seems to be aligning towards our estimations. We can now run this over our complete dataset for better analysis.

results = {}

for index, row in df.iterrows():

positivity = sia.polarity\_scores(row['Text'])['pos']

negativity = sia.polarity\_scores(row['Text'])['neg']

neutral = sia.polarity\_scores(row['Text'])['neu']

compound = sia.polarity\_scores(row['Text'])['compound']

results[index] = {}

results[index]['Positivity'] = positivity

results[index]['Negativity'] = negativity

results[index]['Neutral'] = neutral

results[index]['Compound'] = compound

results

Let's put these results into a Dataframe.

sentiment\_res = pd.DataFrame(results).T.reset\_index()

sentiment\_res.rename(columns = {'index': 'Id'})

sentiment\_res

We can now merge this with our original dataframe so we have all of this sentiment analysis data along with the original data.

merged\_df = sentiment\_res.merge(df, how = 'left')

merged\_df.head()

Let's extract another text from the dataframe to confirm if we have aligned the values correctly

text2 = df['Text'][5476]

print(sia.polarity\_scores(text2))

print(merged\_df['Positivity'][5476])

The values are matching. We now have a new dataframe with both the original data and the analysis results.

We can now make some assumptions based on the score from the original data. Since the score is the number of stars given in a review, the higher the number of stars should mean the higher the positivity factor of the review. Thus we can assume that the positivity value of each review should be directly proportional to the score.

Lets make a bar plot of the results with x value as the Score from the original dataset and y value being the compound from the analysis.

import seaborn as sns

bar\_plot = sns.barplot(data = merged\_df, x = 'Score', y = 'Compound')

bar\_plot.set\_title('Compound vs Star Score')

plt.show()

We can dive a bit deeper and analyse how positivity, negativity and neutral look like over score.

Positivity

bar\_plot\_p = sns.barplot(data = merged\_df, x = 'Score', y = 'Positivity')

bar\_plot\_p.set\_title('Positivity vs Star Score')

plt.show()

Negativity

bar\_plot\_neg = sns.barplot(data = merged\_df, x = 'Score', y = 'Negativity')

bar\_plot\_neg.set\_title('Negativity vs Star Score')

plt.show()

Neutral

bar\_plot\_neg = sns.barplot(data = merged\_df, x = 'Score', y = 'Neutral')

bar\_plot\_neg.set\_title('Neutral vs Star Score')

plt.show()

Clearly, as the star review goes higher, positivity keeps going higher in the sentiment analysis of the text. The negativity goes lower.

Inversely, as the star review goes higher, negativity keeps dropping in the sentiment analysis of the text. The positivity goes higher.

# BIBLIOGRAPHY

1. Stanford Network Analysis Project (2013). Amazon Fine Food Reviews [Dataset]. Retrieved from <https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews>

2. McAuley, J., & Leskovec, J. (2013). From Amateurs to Connoisseurs: Modeling the Evolution of User Expertise through Online Reviews.