CSE508 Information Retrieval

ASSIGNMENT-3 Report

Divyansh Mishra
2021042

In order to account for the computational power required for this assignment we used kaggle.
Reading the json files:

```
import pandas as pd
import gzip
import feather
import json

def parse(path):
  with open(path, 'r') as g:
      for l in g:
        yield json.loads(l)

def getDF(path):
  i = 0
  df = {}
  for d in parse(path):
    df[i] = d
    i += 1
  return pd.DataFrame.from_dict(df, orient='index')

df = getDF('/kaggle/input/amazon-rev-data/amazon-review-data/Electronics_5.json/Electronics_5.json')
```

We chose the product console for this assignment.

```
df_cat = df_meta[df_meta['title'].str.contains('console', case=False)]
df_cat.drop_duplicates(subset=['asin'],inplace=True)
print(len(df_cat))
# df_headphones.head
df_filtered_reviews = df[df['asin'].isin(df_cat['asin'])]
print(len(df_filtered_reviews))
# df_merged = df_filtered_reviews.merge(df_cat, on='asin', how='inner')
# df_filtered_reviews=df_merged
# df_filtered_reviews = pd.merge(df, df_cat, on='asin', how='inner')
# print(df_filtered_reviews.head)
```

```
658
5836
```

The descriptive statistics that we got were:

```
Number of reviews: 5836
Average Rating Score: 4.3658327621658675
Number of Unique Products: 145
Number of Good Ratings: 5255
Number of Bad Ratings: 581

Number of Reviews corresponding to each Rating:
overall
1.0     368
2.0     213
3.0     344
4.0     902
5.0    4009
Name: count, dtype: int64
```

We then preprocessed the review text as follows:

```python
# Apply preprocessing functions
df_filtered_reviews['reviewText'] = df_filtered_reviews['reviewText'].apply(remove_html_tags)
df_filtered_reviews['reviewText'] = df_filtered_reviews['reviewText'].apply(remove_accented_chars)
df_filtered_reviews['reviewText'] = df_filtered_reviews['reviewText'].apply(remove_special_characters)
df_filtered_reviews['reviewText'] = df_filtered_reviews['reviewText'].apply(lemmatize_text)
df_filtered_reviews['reviewText'] = df_filtered_reviews['reviewText'].apply(normalize_text)
# df_filtered_reviews['reviewText'] = df_filtered_reviews['reviewText'].apply(ch)
```

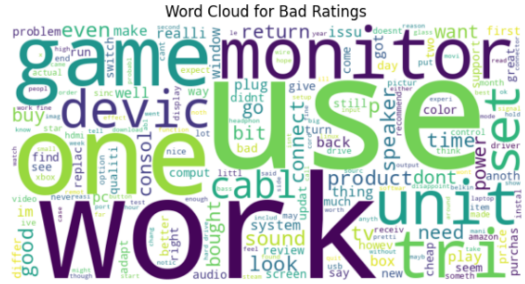
Rating count over 5 years:

```
Count of Ratings for Most Positively Reviewed Console Over 5 Consecutive Years:
Year 2014: 749 ratings
Year 2015: 1144 ratings
Year 2016: 1296 ratings
Year 2017: 1201 ratings
Year 2018: 769 ratings
```

Word clouds:

Word Cloud for Good Ratings

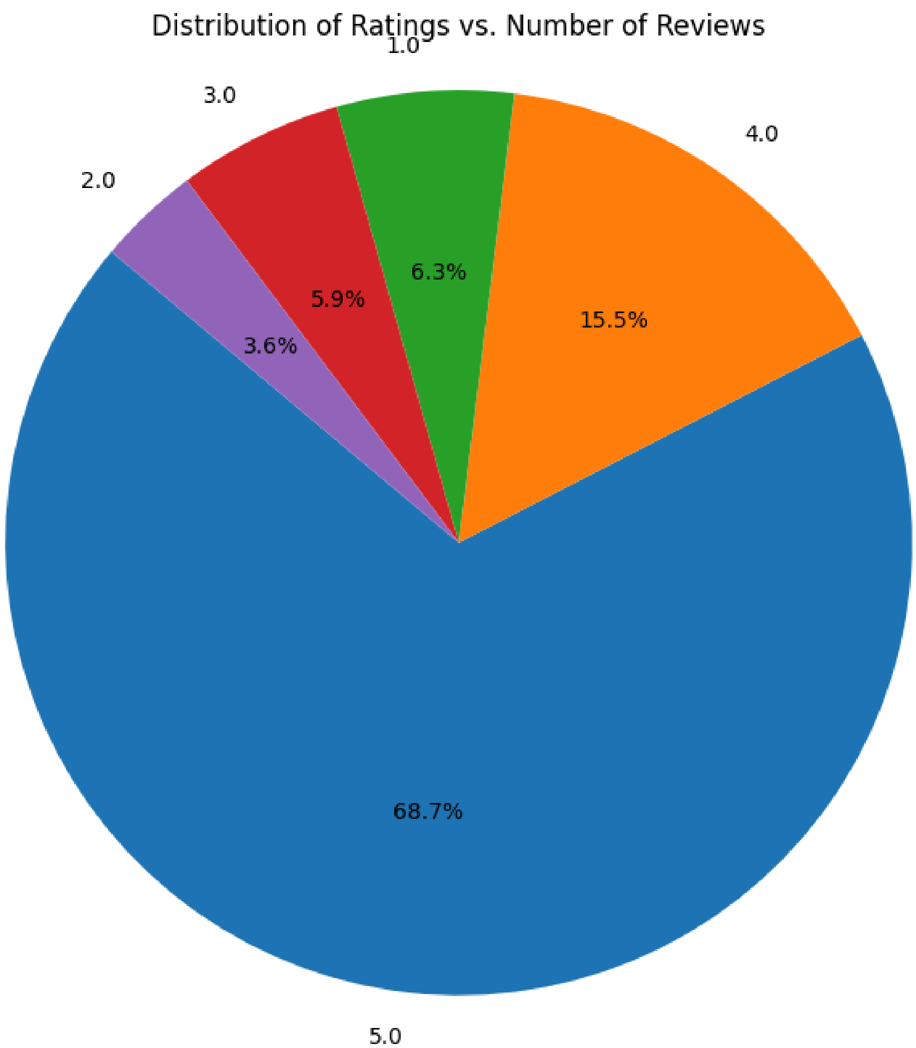
Word Cloud for Bad Ratings



The EDS that we performed gave us these values: and the product ID of the most positively reviewed product on the basis of average overall for each asin.

```
Top 20 Most Reviewed Brands:
brand
Plugable                   606
ORIA                       604
BenQ                       576
Logitech                   374
Panlong                    271
Belkin                     242
Cellnorth Electronics      237
by\n    \n    TOMSENN      186
Cisco                      181
Asunflower                 172
HDE                        169
StarTech                   139
Fosmon                     129
UGREEN                     129
Alienware                  114
hossen                     109
Creative                    84
E-sds                       84
Edimax                      81
Pyle                        80
Name: count, dtype: int64
Top 20 Least Reviewed Brands:
brand
TRENDnet                     6
IOGEAR                       6
InterlinkAuckland            6
MOCREO                       6
ULBRE                        6
Bose                         5
Stuff4                       5
Hausbel                      5
Rocketfish                   5
2gig                         5
Crosley                      5
LiKe                         5
Inland                       5
Importer520                  5
Zendure                      5
Marantz                      5
Hercules                     5
Abrams                       5
PowerMonkey                  4
elexa consumer products      4
Name: count, dtype: int64
ASIN: B01EAYYJAS
      overall vote  verified   reviewTime      reviewerID        asin style  \
5799      5.0  NaN      True  07 15, 2018  A2O8EYXZZ1PPYL  B01EAYYJAS   NaN
5800      5.0  NaN      True  06 23, 2018  A2VS5SQROVJK41  B01EAYYJAS   NaN
```

Distribution:




```
# Group the reviews by 'reviewYear' and count the number of reviews for each year
reviews_per_year = df_filtered_reviews['reviewYear'].value_counts()

# Find the year with the maximum number of reviews
max_reviews_year = reviews_per_year.idxmax()
max_reviews_count = reviews_per_year.max()

# Report the year with the maximum reviews
print(f"The year with the maximum reviews is {max_reviews_year} with {max_reviews_count} reviews.")
```

```
The year with the maximum reviews is 2016 with 1296 reviews.
```

```
Year with the Highest Number of Customers (Considering Verified Reviews Only):
Year: 2016, Number of Customers: 1163
```

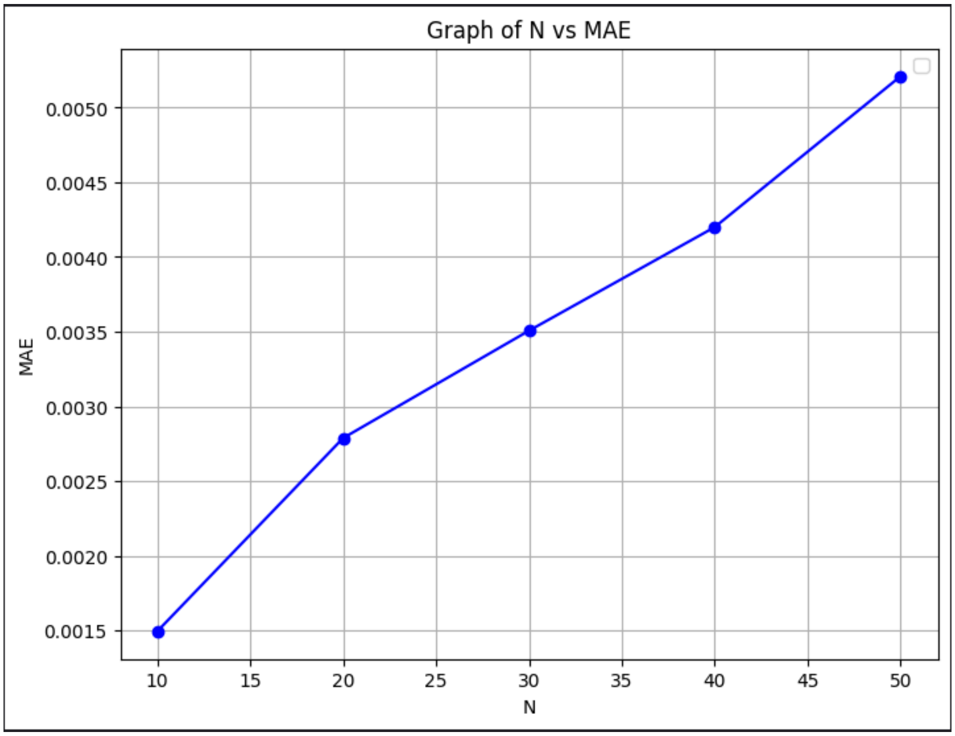
We the the TF-IDF for feature engineering

The ML models that we used were: "Logistic Regression",   "Random Forest",   "Support Vector Classifier",   "Multinomial Naive Bayes",   "Gradient Boosting Classifier"

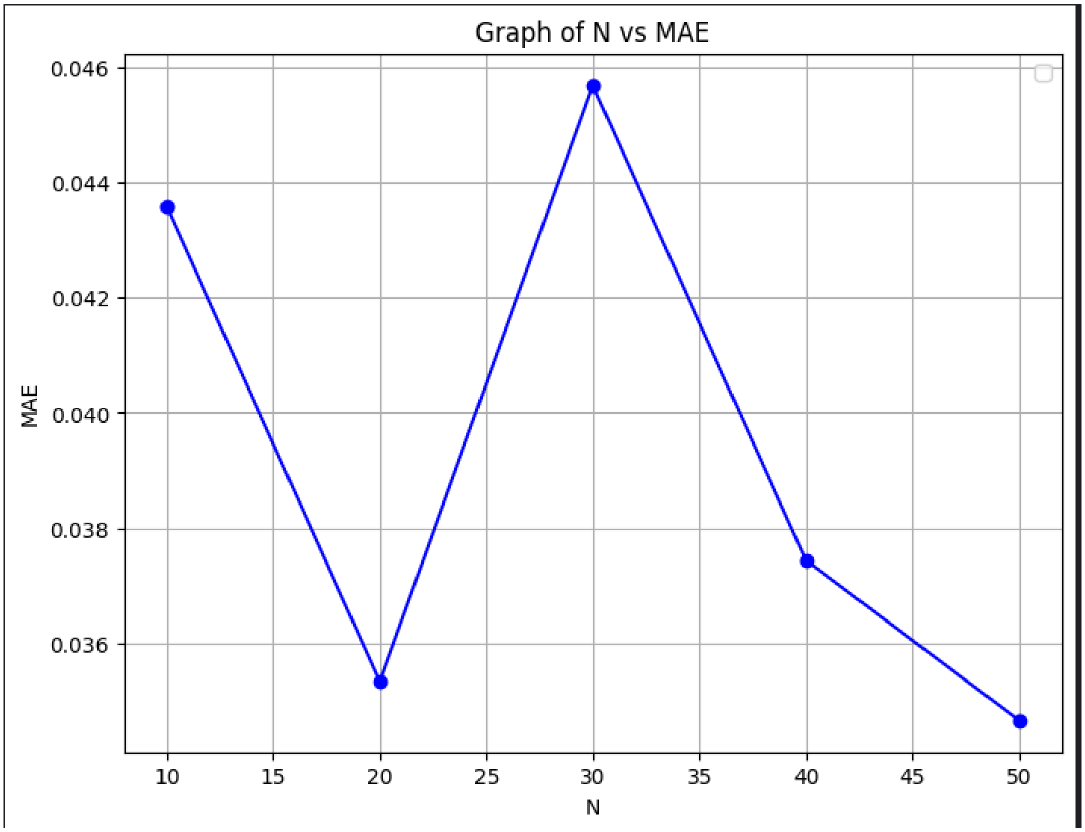
```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

     Average       0.00      0.00      0.00        99
         Bad       0.78      0.31      0.45       143
        Good       0.86      1.00      0.93      1217

    accuracy                           0.86      1459
   macro avg       0.55      0.44      0.46      1459
weighted avg       0.80      0.86      0.82      1459

======================================================================
Training Random Forest...

Random Forest Classification Report:
              precision    recall  f1-score   support

     Average       0.29      0.02      0.04        99
         Bad       0.83      0.21      0.34       143
        Good       0.85      0.99      0.92      1217

    accuracy                           0.85      1459
   macro avg       0.66      0.41      0.43      1459
weighted avg       0.81      0.85      0.80      1459

======================================================================
Training Support Vector Classifier...

Support Vector Classifier Classification Report:
              precision    recall  f1-score   support

     Average       0.50      0.01      0.02        99
         Bad       0.76      0.43      0.55       143
        Good       0.88      0.99      0.93      1217

    accuracy                           0.87      1459
   macro avg       0.71      0.48      0.50      1459
weighted avg       0.84      0.87      0.83      1459

======================================================================
```

```
======================================================================
Training Multinomial Naive Bayes...

Multinomial Naive Bayes Classification Report:
              precision    recall  f1-score   support

     Average       0.00      0.00      0.00        99
         Bad       1.00      0.01      0.01       143
        Good       0.83      1.00      0.91      1217

    accuracy                           0.83      1459
   macro avg       0.61      0.34      0.31      1459
weighted avg       0.79      0.83      0.76      1459

======================================================================
Training Gradient Boosting Classifier...
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1
ed and being set to 0.0 in labels with no predicted samples. Use `zero_divis
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1
ed and being set to 0.0 in labels with no predicted samples. Use `zero_divis
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1
ed and being set to 0.0 in labels with no predicted samples. Use `zero_divis
  _warn_prf(average, modifier, msg_start, len(result))

Gradient Boosting Classifier Classification Report:
              precision    recall  f1-score   support

     Average       0.67      0.02      0.04        99
         Bad       0.68      0.28      0.40       143
        Good       0.86      0.99      0.92      1217

    accuracy                           0.86      1459
   macro avg       0.74      0.43      0.45      1459
weighted avg       0.83      0.86      0.81      1459

======================================================================
```

```
MAE AT N= 10 is 0.0014940755322140561
MAE AT N= 20 is 0.002787501995090178
MAE AT N= 30 is 0.0035063424866803978
MAE AT N= 40 is 0.004200063842885699
MAE AT N= 50 is 0.005209761881237032
```



```
MAE for Item-Item Recommender System with N=10: 0.043581357877376055
MAE for Item-Item Recommender System with N=20: 0.03534285149688765
MAE for Item-Item Recommender System with N=30: 0.045681991492682136
MAE for Item-Item Recommender System with N=40: 0.03744580955059169
MAE for Item-Item Recommender System with N=50: 0.034672660804268354
```




```
# Calculate the sum of ratings for each product across all users
product_sum_ratings = df_filtered_reviews.groupby('asin')['overall'].sum()

# Sort the products based on sum of ratings in descending order
top_10_products = product_sum_ratings.sort_values(ascending=False).head(10)

# Print the top 10 products by sum of ratings
print("Top 10 Products by User Sum Ratings:")
for rank, (product, rating_sum) in enumerate(top_10_products.items(), 1):
    print(f"{rank}. Product ID: {product}, Sum of Ratings: {rating_sum}")
```

```
Top 10 Products by User Sum Ratings:
1. Product ID: B01E16J6RQ, Sum of Ratings: 2762.0
2. Product ID: B00AQM8586, Sum of Ratings: 2465.0
3. Product ID: B007HSKSMI, Sum of Ratings: 1458.0
4. Product ID: B00KXVBB3Q, Sum of Ratings: 1171.0
5. Product ID: B0151K2AB0, Sum of Ratings: 1114.0
6. Product ID: B00JPBFC8U, Sum of Ratings: 842.0
7. Product ID: B005SN3INA, Sum of Ratings: 803.0
8. Product ID: B00RORBPCO, Sum of Ratings: 564.0
9. Product ID: B0006U3ACY, Sum of Ratings: 517.0
10. Product ID: B00KMRVGFO, Sum of Ratings: 496.0
```