UCSanDiegoX DSE200x
Final Project

Sentiment analysis to determine whether Amazon Kindle products are likely to have a high or low rating

Megan A Holborn

Abstract

Using customer review data from the Amazon Kindle Store collected between May 1996 and July 2014, this analysis aimed to determine whether sentiment analysis of textual product reviews could be used to predict whether Kindle products received high or low ratings. A Naïve Bayes classifier model was utilised and rating polarities were successfully predicted at a high accuracy. The results suggest that sentiment analysis of textual reviews can effectively be used to categorise the rating levels of Amazon Kindle products.

Motivation

- Online product reviews provide valuable insight into customer sentiment that can be utilised by businesses to make informed decisions about product purchasing and advertisement, and reputation management.

- The Amazon Kindle store provides a popular range of e-book and e-reader product offerings, attracting a large amount of customer reviews.

- Understanding the customer sentiments associated with positive and negative ratings in Kindle reviews, could assist e-book/e-reader buyers in making informed purchasing decisions. Additionally, it can provide feedback to Amazon that aids in the selection of recommended products and highlights key product issues.

- This analysis aimed to determine whether the sentiments expressed in Amazon Kindle product reviews could serve as indicators for a product’s overall rating.

Dataset

- Amazon Kindle Store customer reviews collected between May 1996 and July 2014 were used for this analysis.

- The data is available at:

  https://www.kaggle.com/datasets/bharadwaj6/kindle-reviews/versions/3?resource=download

- The dataset was retrieved in Json format and consists of 982 619 review entries and 9 features, prior to filtering.

- The dataset provides information on (i) customer reviews, including reviewer ID, review text, review summary, and the review time, (ii) the reviewed product ID, and (iii) the overall rating of a product.

Data Preparation and Cleaning

The dataset was prepared for sentiment analysis as follows:

1. Entries with null values and duplicate entries were located and removed if present.
2. Relevant dataset features, namely, the review text and overall rating of a product, were selected for the analysis.
3. Review entries with a corresponding rating greater than 3, equal to 3 and less than 3 were categorised as positive, neutral, and negative, respectively.
4. The dataset consisted of 982 483 review entries. Random samples of 20 000 positive and 20 000 negative reviews were selected for sentiment analysis.
5. In preparation for sentiment analysis, review text was converted to lowercase and tokenized using NLTK. Stopwords and punctuation were removed. Negation handling was also performed to ensure that words preceded by negations would not be misinterpreted.

Research Question

Can sentiment analysis on textual reviews be used to predict which Amazon Kindle products received high or low ratings?

Methods

1. Reviews with a corresponding rating greater than 3, equal to 3 and less than 3 were categorised as positive, neutral, and negative, respectively.

2. Using a bag-of-words model, words associated with each positive or negative review were collected.

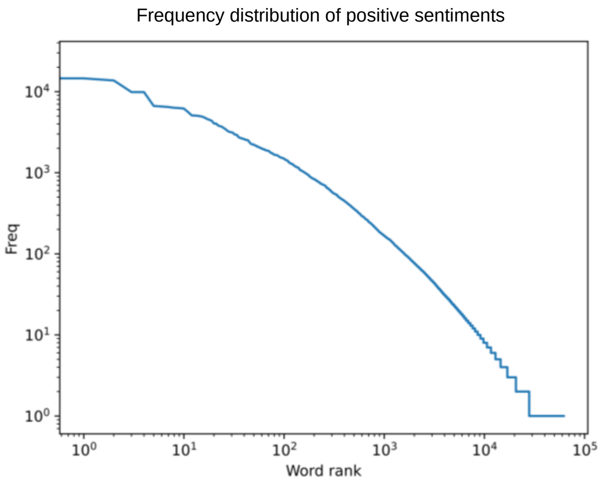
3. Log-log plots were used to visualize the frequency distribution of all positive and negative words. Additionally, word clouds were used to visually represent the most common positive and negative words found in these reviews.

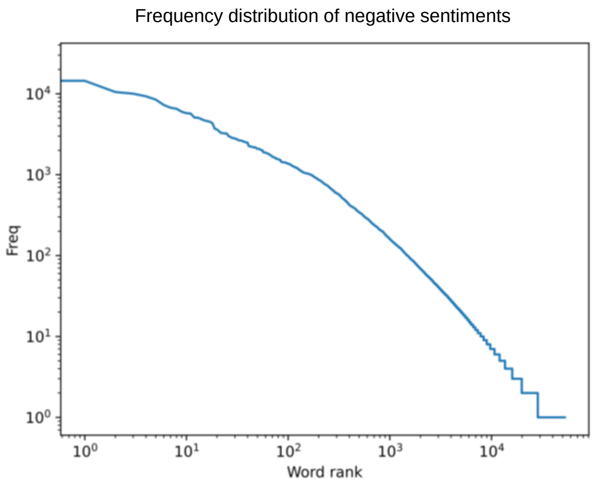
4. To predict which reviews were likely to have a positive (high) or negative (low) rating, a Naïve Bayes machine learning classifier was trained on the features associated with positive and negative reviews. The classifier was trained on 80% of the data. The remaining data was used for testing purposes.

Findings

Frequency distribution of positive and negative sentiments

- The gradual decline of the distribution curves indicated a broad vocabulary usage in both positive and negative reviews.
- Positive reviews contained more words than negative reviews (62137 vs 52560 words).



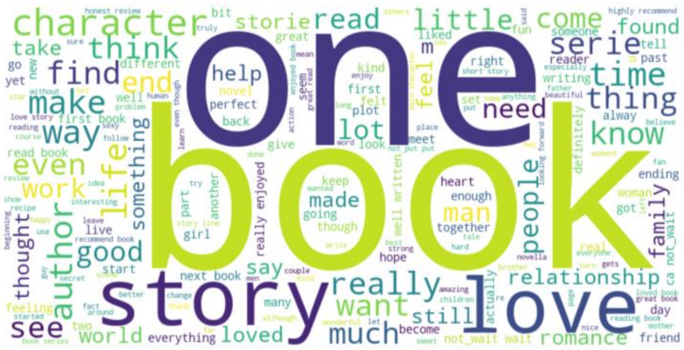


Findings

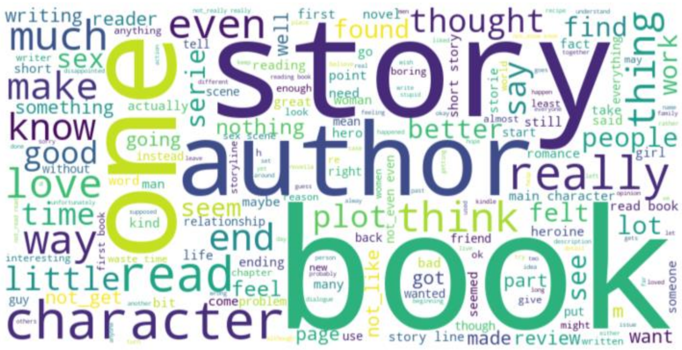
Common positive and negative sentiments

- Several of the most common words were shared between positive and negative reviews.
- In the positive reviews, the most frequent words, in descending order, were book (n=25096), story (n=14574) and one (n=9870). Similarly, in the negative reviews, the most frequent words, in descending order, were book (n=25452), story (n=14471) and one (n=9307).
- The shared common words are unlikely to be informative in distinguishing between positive and negative reviews.

Common positive sentiments



Common negative sentiments



Findings

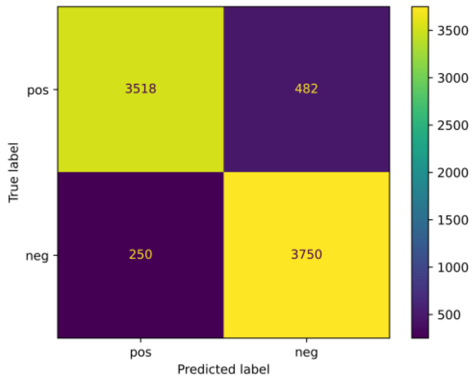
Naïve Bayes classification of textual reviews

- A Naïve Bayes classifier, trained on 80% of the review data, predicted which Amazon Kindle products received positive (high) or negative (low) ratings with 90% accuracy.
- According to the confusion matrix, of the 8000 reviews tested using the classifier, 7268 were correctly classified as positive or negative, while 732 were incorrectly classified.

Accuracy report

Train accuracy	Test accuracy
95.09%	90.02%

Confusion matrix



Findings

Informative features for classification

- The classification model was trained using the most informative features, which were words that were likely to occur in positive reviews but not in negative reviews, and vice versa. The top 5 features are listed in the table below.

Informative features

Feature	Occurrence ratio
not_disappoint	pos : neg = 104.3 : 1
4.5	pos : neg = 73.0 : 1.0
Deleted	neg : pos = 62.1 : 1.0
not_impressed	neg : pos = 52.3 : 1.0
not_finish	neg : pos = 44.7 : 1.0

Limitations

- For this analysis, a single machine learning algorithm was employed. No other algorithms were tested to determine which would be most suitable.

- Many words were shared between both negative and positive reviews. In future analyses, it may be useful to group words in phrases to aid in distinguishing between sentiments expressed in positive and negative reviews.

- It is also worth noting that the classifier did not predict the exact rating of a product, but rather whether a product received a high or low rating. It would be beneficial to incorporate more fine-scale predictions of product rating in future analyses.

Conclusions

Despite the presence of a wide range of vocabulary and shared common words between positive and negative reviews for Amazon Kindle products, a Naïve Bayes classifier model was successfully utilised to accurately predict which Amazon Kindle products received high or low ratings using textual review data. This was made possible due to the utilisation of informative features, which were words that were likely to occur in positive reviews but not in negative reviews, and vice versa.

Acknowledgements

The data used in this analysis was obtained from:
https://www.kaggle.com/datasets/bharadwaj6/kindle-reviews/versions/3?resource=download

This data was originally compiled by Julian McAuley:
http://jmcauley.ucsd.edu/data/amazon/

References

Taparia, Ankit and Bagla, Tanmay, Sentiment Analysis: Predicting Product Reviews' Ratings using Online Customer Reviews (May 15, 2020). Available at SSRN: https://ssrn.com/abstract=3655308 or http://dx.doi.org/10.2139/ssrn.3655308

Final Project

As the final project for this course, I will be analysing customer product review data from the Amazon Kindle Store to answer the following research question: **Can sentiment analysis be used to predict which Amazon Kindle products received high or low ratings?**

The data used in this project was retrieved from Kaggle: https://www.kaggle.com/datasets/bharadwaj6/kindle-reviews/versions/3?resource=download.

Import libraries

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import nltk
import string
from wordcloud import WordCloud
from collections import Counter, OrderedDict
from nltk.classify import NaiveBayesClassifier
import random
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatri
```

Import data

In [ ]:
```
amazon_reviews = pd.read_json("../Data/kindle_reviews.json", lines=True)
```

Conduct an initial exploration of the data

What is the shape of the data?

In [ ]:
```
amazon_reviews.shape
```

Out[ ]:
```
(982619, 9)
```

The dataset has 9 columns and 982619 rows. What information is contained in the columns?

In [ ]:
```
amazon_reviews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 982619 entries, 0 to 982618
Data columns (total 9 columns):
 #   Column          Non-Null Count   Dtype 
---  ------          --------------   ----- 
 0   reviewerID      982619 non-null  object
 1   asin            982619 non-null  object
 2   reviewerName    978820 non-null  object
 3   helpful         982619 non-null  object
 4   reviewText      982619 non-null  object
 5   overall         982619 non-null  int64 
 6   summary         982619 non-null  object
 7   unixReviewTime  982619 non-null  int64 
 8   reviewTime      982619 non-null  object
dtypes: int64(2), object(7)
memory usage: 67.5+ MB
```

How many missing values are present?

In [ ]:
```
amazon_reviews.isna().sum()
```

Out[ ]:
```
reviewerID           0
asin                 0
reviewerName      3799
helpful              0
reviewText           0
overall              0
summary              0
unixReviewTime       0
reviewTime           0
dtype: int64
```

How many unique products are present?

In [ ]:
```
len(amazon_reviews.asin.unique())
```

Out[ ]:
```
61934
```

How many unique reviewers were there?

In [ ]:
```
len(amazon_reviews.reviewerID.unique())
```

Out[ ]:
```
68223
```

Select dataset features

Select the reviewText, summary and overall dataset columns for further analysis.

In [ ]:
```
amazon_reviews_subset = amazon_reviews[["reviewText", "summary", "overall"]].copy(
len(amazon_reviews_subset)
```

Out[ ]:
```
982619
```

Remove duplicate reviews

In [ ]:
```
amazon_reviews_subset = amazon_reviews_subset.drop_duplicates()
len(amazon_reviews_subset)
```

Out[ ]: 982483

Divide reviews into groups corresponding to rating.

Reviews with a rating of less than 3, equal to 3, and greater than 3 will be negative, neutral and positive, respectively.

In [ ]:
```
conditions = [
    amazon_reviews_subset["overall"] < 3,
    amazon_reviews_subset["overall"] == 3,
    amazon_reviews_subset["overall"] > 3
]

choices = ["negative", "neutral", "positive"]

amazon_reviews_subset["sentiment"] = np.select(conditions, choices)

amazon_reviews_subset.head(5)
```

Out[ ]:

	reviewText	summary	overall	sentiment
0	I enjoy vintage books and movies so I enjoyed ...	Nice vintage story	5	positive
1	This book is a reissue of an old one; the auth...	Different...	4	positive
2	This was a fairly interesting read. It had ol...	Oldie	4	positive
3	I'd never read any of the Amy Brewster mysteri...	I really liked it.	5	positive
4	If you like period pieces - clothing, lingo, y...	Period Mystery	4	positive

Count the number of positive and negative reviews

In [ ]:
```
len(amazon_reviews_subset[amazon_reviews_subset["sentiment"] == "positive"])
```

Out[ ]: 829147

In [ ]:
```
len(amazon_reviews_subset[amazon_reviews_subset["sentiment"] == "negative"])
```

Out[ ]: 57148

Take a sample of 40000 reviews (20000 positive, 20000 negative) for testing

In [ ]:
```
pos_sample = amazon_reviews_subset[amazon_reviews_subset["sentiment"] == "positive'
neg_sample = amazon_reviews_subset[amazon_reviews_subset["sentiment"] == "negative'

amazon_reviews_sample = pd.concat([pos_sample, neg_sample])
amazon_reviews_sample.head(5)
```

Out[ ]:

	reviewText	summary	overall	sentiment
848734	I loved this new series by Christina Tetreault...	OMG another awesome book	5	positive
823954	Yes! This was a hot read, so hot that I HAD to...	Freaky	5	positive
831623	I was gifted this book in exchange for an hone...	Amazing	5	positive
503383	This short story was very hot. The tension and...	Very hot	5	positive
372736	It was almost painful as the reader to see how...	So Much Pain	5	positive

Convert reviews to lowercase

In [ ]:

```
amazon_reviews_sample["reviewText"] = amazon_reviews_sample["reviewText"].astype(st
amazon_reviews_sample.head(5)
```

Out[ ]:

	reviewText	summary	overall	sentiment
848734	i loved this new series by christina tetreault...	OMG another awesome book	5	positive
823954	yes! this was a hot read, so hot that i had to...	Freaky	5	positive
831623	i was gifted this book in exchange for an hone...	Amazing	5	positive
503383	this short story was very hot. the tension and...	Very hot	5	positive
372736	it was almost painful as the reader to see how...	So Much Pain	5	positive

Tokenize text

Split the text into individual words

In [ ]:

```
amazon_reviews_sample["tokenized_review"] =  amazon_reviews_sample.reviewText.apply
amazon_reviews_sample.head(5)
```

Out[ ]:

	reviewText	summary	overall	sentiment	tokenized_review
848734	i loved this new series by christina tetreault...	OMG another awesome book	5	positive	[i, loved, this, new, series, by, christina, t...
823954	yes! this was a hot read, so hot that i had to...	Freaky	5	positive	[yes, !, this, was, a, hot, read, ,, so, hot, ...
831623	i was gifted this book in exchange for an hone...	Amazing	5	positive	[i, was, gifted, this, book, in, exchange, for...
503383	this short story was very hot. the tension and...	Very hot	5	positive	[this, short, story, was, very, hot, ., the, t...
372736	it was almost painful as the reader to see how...	So Much Pain	5	positive	[it, was, almost, painful, as, the, reader, to...

Remove stop words and punctuation

Remove punctuation and words that are not relevant

In [ ]:

```
# Retrieve punctuation and common english stop words to be removed
removed_words = nltk.corpus.stopwords.words("english") + list(string.punctuation)

# Drop negations from the removed_words list. Negations will be handled later in th
for word in ['not', 'n\'t', 'no', 'never']:
    if word in removed_words:
        removed_words.remove (word)

# Add some stop words

removed_words.extend(["\'s", "''", "``", "..."])
```

In [ ]:

```
amazon_reviews_sample["without_stop_words"] = amazon_reviews_sample["tokenized_revi
amazon_reviews_sample.head(5)
```

Out[ ]:

	reviewText	summary	overall	sentiment	tokenized_review	without_stop_words
848734	i loved this new series by christina tetreault...	OMG another awesome book	5	positive	[i, loved, this, new, series, by, christina, t...	[loved, new, series, christina, tetreault, lov...
823954	yes! this was a hot read, so hot that i had to...	Freaky	5	positive	[yes, !, this, was, a, hot, read, ,, so, hot, ...	[yes, hot, read, hot, get, part, 2, immediatel...
831623	i was gifted this book in exchange for an hone...	Amazing	5	positive	[i, was, gifted, this, book, in, exchange, for...	[gifted, book, exchange, honest, review.this, ...
503383	this short story was very hot. the tension and...	Very hot	5	positive	[this, short, story, was, very, hot, ., the, t...	[short, story, hot, tension, desire, nearly, p...
372736	it was almost painful as the reader to see how...	So Much Pain	5	positive	[it, was, almost, painful, as, the, reader, to...	[almost, painful, reader, see, much, pain, two...

Handle negation

Words preceded by negations should not be interpreted as positive

In [ ]:

```
def handle_negation(tokenized_list):

    negation_handled_list = []

    for word in tokenized_list:
        # next_word = tokenized_list[tokenized_list.index(word) + 1]
        if word in ['not', 'n\'t', 'no', 'never']:
            if tokenized_list.index(word) < len (tokenized_list) - 1:
                next_word = tokenized_list[tokenized_list.index(word) + 1]
                negation_handled_list.append("not_" + next_word)
        else:
            negation_handled_list.append(word)

    # negation_handled_list = ', '.join(negation_handled_list)
    return negation_handled_list
```

In [ ]:

```
amazon_reviews_sample["handled_negation"] =  amazon_reviews_sample.without_stop_wor
amazon_reviews_sample.head(5)
```

Out[ ]:

	reviewText	summary	overall	sentiment	tokenized_review	without_stop_words	handled
848734	i loved this new series by christina tetreault...	OMG another awesome book	5	positive	[i, loved, this, new, series, by, christina, t...	[loved, new, series, christina, tetreault, lov...	[loved, r christina
823954	yes! this was a hot read, so hot that i had to...	Freaky	5	positive	[yes, !, this, was, a, hot, read, ,, so, hot, ...	[yes, hot, read, hot, get, part, 2, immediatel...	[yes, hot g im
831623	i was gifted this book in exchange for an hone...	Amazing	5	positive	[i, was, gifted, this, book, in, exchange, for...	[gifted, book, exchange, honest, review.this, ...	[gi exchan rev
503383	this short story was very hot. the tension and...	Very hot	5	positive	[this, short, story, was, very, hot, ., the, t...	[short, story, hot, tension, desire, nearly, p...	[short, tens
372736	it was almost painful as the reader to see how...	So Much Pain	5	positive	[it, was, almost, painful, as, the, reader, to...	[almost, painful, reader, see, much, pain, two...	[almc reader,

Visualize the most common negative and positive sentiments

Extract and group all positive and negative words. Construct a WordCloud for the positive and negative groups.

In [ ]:

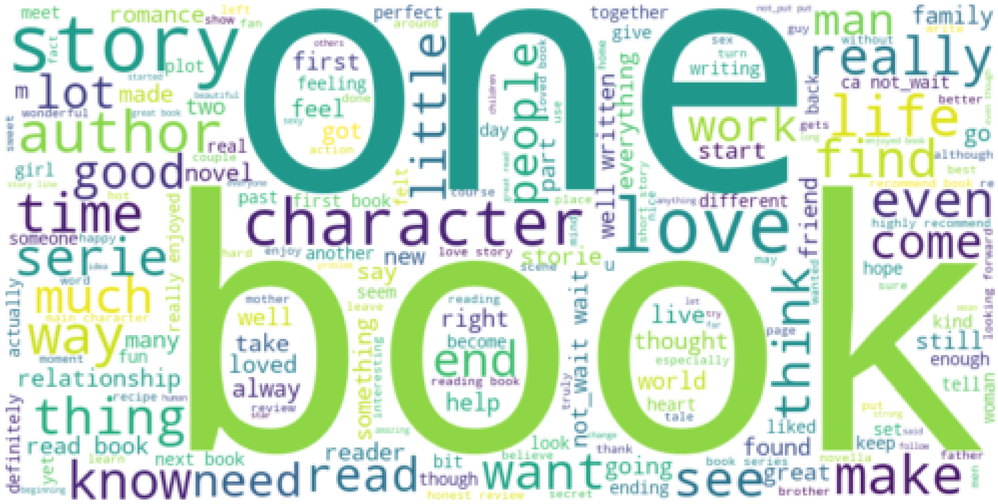
```
positive_reviews =  amazon_reviews_sample[amazon_reviews_sample["sentiment"] == "po
negative_reviews = amazon_reviews_sample[amazon_reviews_sample["sentiment"] == "neg
```

In [ ]:

```
positive_words = WordCloud(background_color='white', width=800, height=400).generat
```

In [ ]:

```
plt.imshow(positive_words, interpolation="bilinear")
plt.axis("off")

# Save the figure
plt.savefig("../Results/Figures/Positive_wordcloud.png", dpi=1200)

plt.show()
```




```
In [ ]: negative_words = WordCloud(background_color='white', width=800, height=400).generat
```

```
In [ ]: plt.imshow(negative_words, interpolation="bilinear")
        plt.axis("off")

        # Save the figure
        plt.savefig("../Results/Figures/Negative_wordcloud.png", dpi=1200)

        plt.show()
```

Get the counts and frequencies of the positive and negative sentiments

Create lists with all positive and negative words

How many positive words were used?

```
In [ ]: positive_words = []
        for list in positive_reviews:
            positive_words.extend(list)

        len (positive_words)
```

```
Out[ ]: 1128424
```

Many words were used in the positive reviews. How many of these words are unique?

In [ ]:
```
len (set(positive_words))
```

Out[ ]:
```
62137
```

How many negative words were used?

In [ ]:
```
negative_words = []
for list in negative_reviews:
    negative_words.extend(list)

len (negative_words)
```

Out[ ]:
```
1058692
```

Less words were used in the negative reviews in comparison to the positive reviews. How many of these words are unique?

In [ ]:
```
len (set(negative_words))
```

Out[ ]:
```
52560
```

Count the frequencies of the positive and negative words

In [ ]:
```
pos_word_counter = Counter(positive_words)
```

In [ ]:
```
neg_word_counter = Counter(negative_words)
```

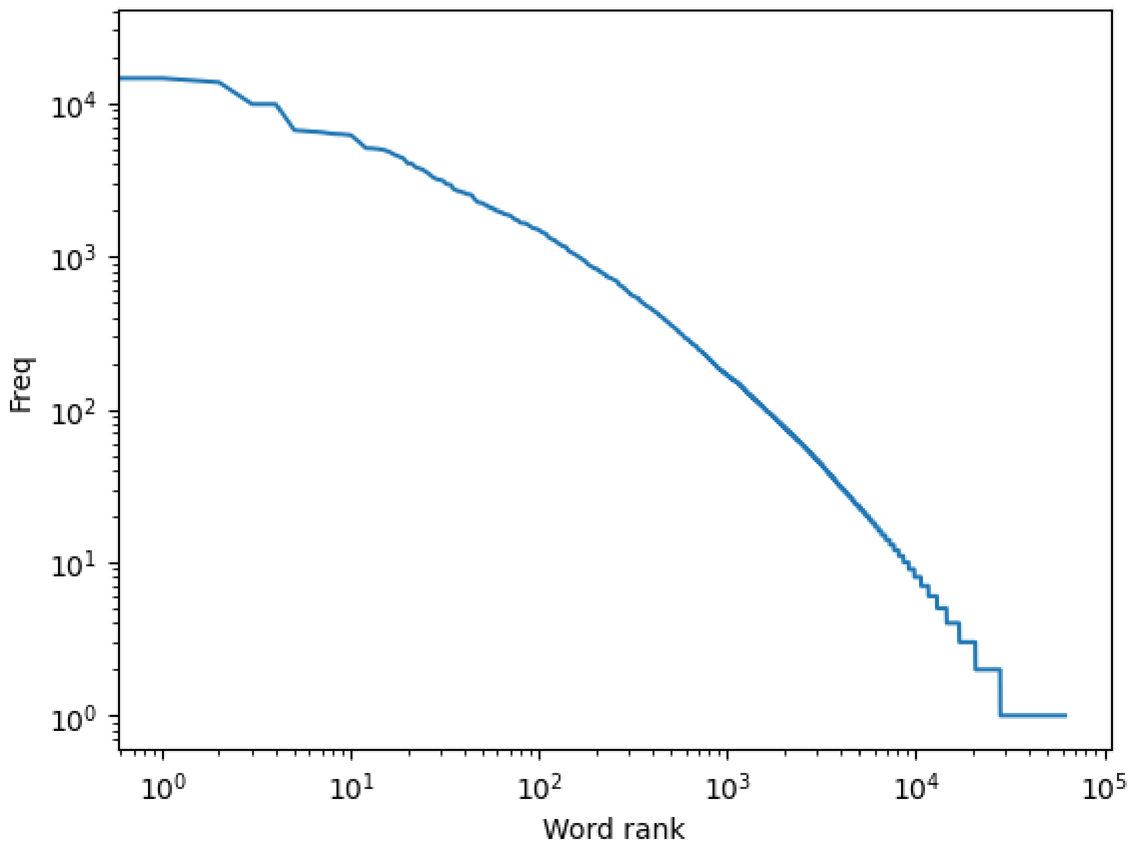
Plot the frequencies

Sort the numerical data

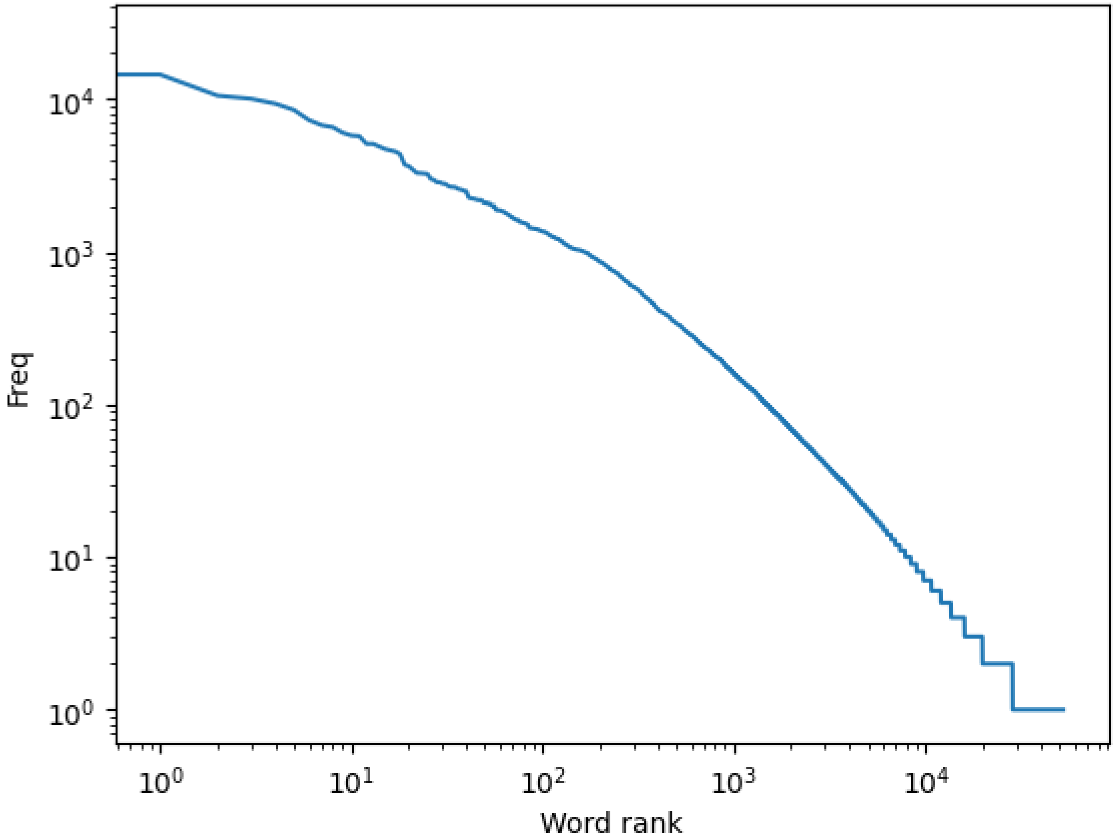
In [ ]:
```
pos_sorted_word_counts = sorted(pos_word_counter.values(), reverse=True)
neg_sorted_word_counts = sorted(neg_word_counter.values(), reverse=True)
```

Plot the positive word count frequencies

In [ ]:
```
plt.loglog(pos_sorted_word_counts)
plt.ylabel("Freq")
plt.xlabel("Word rank")

# Save the figure
plt.savefig("../Results/Figures/Positive_frequency.png", dpi=1200)

plt.show()
```



In [ ]:
```
plt.loglog(neg_sorted_word_counts)
plt.ylabel("Freq")
plt.xlabel("Word rank")

# Save the figure
plt.savefig("../Results/Figures/Negative_frequency.png", dpi=1200)

plt.show()
```



Train a classifier for sentiment analysis

Create a bag of words to house the positive and negative words contained in each review.

In [ ]:
```
# Create a function to build a bag of words for each review

def build_bag_of_words(review):
    return {word:1 for word in review}
```

In [ ]:
```
# Build a list of positive/negative words contained in each review. Associate the t

negative_features = [
    (build_bag_of_words(value), "neg") for value in negative_reviews.values
]

positive_features = [
    (build_bag_of_words(value), "pos") for value in positive_reviews.values
]
```

Classify the data using a Naive Bayes supervised machine learning classifier. Train the data on 80% of the data. Use the remaining 20% for testing purposes.

Determine how many records make up 80% of the data.

In [ ]:
```
pos_split = len (positive_features) * 0.8
round(pos_split)
```

Out[ ]: 16000

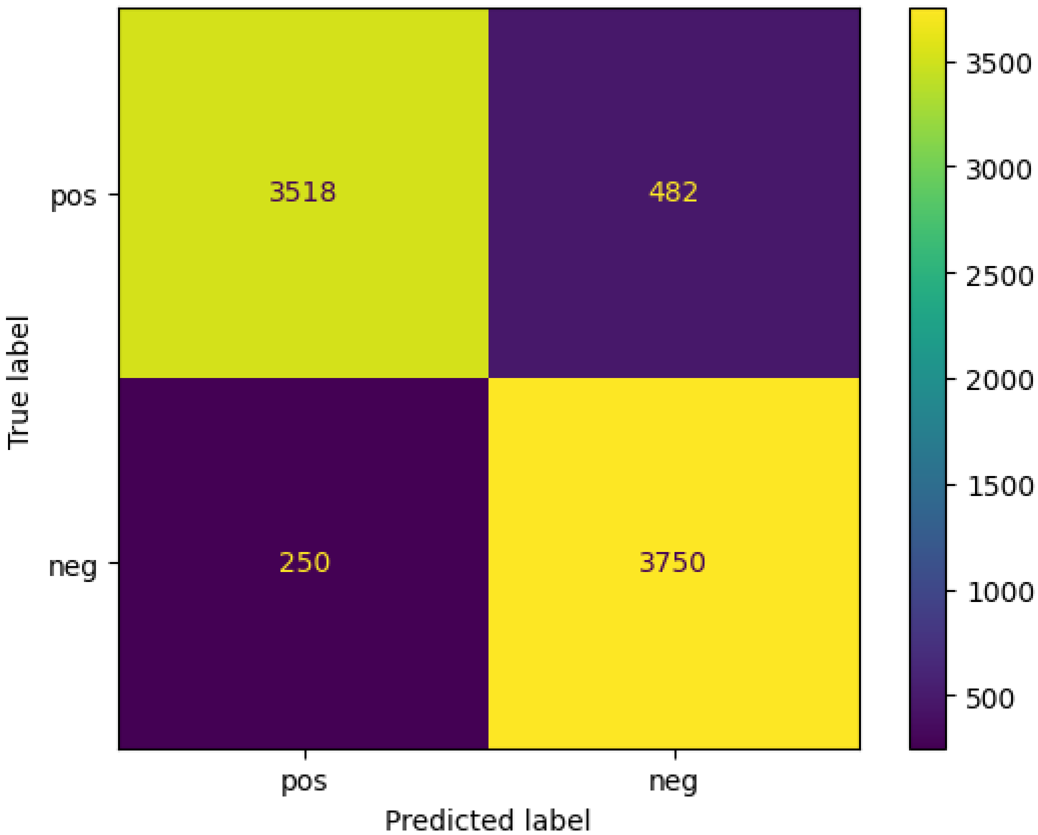
In [ ]:
```
neg_split = len (negative_features) * 0.8
round(neg_split)
```

Out[ ]: 16000

Train the classifier. Check the accuracy of the training set. We expect a high percentage as the classifier algorithm already saw this data. Check the accuracy using the test data. The classifier algorithm has not seen this data.

In [ ]:

```
accur_report = pd.DataFrame()

# Create a random sample of positive and negative training data

pos_train = positive_features[:16000]
neg_train = negative_features[:16000]

# Get the remaining test data

pos_test = positive_features[16000:]
neg_test = negative_features[16000:]

# Train the classifier

classifier = NaiveBayesClassifier.train(pos_train + neg_train)

# Test the accuracy

train_accuracy = nltk.classify.util.accuracy(classifier, pos_train + neg_train)*10
test_accuracy = nltk.classify.util.accuracy(classifier, pos_test + neg_test)*100

# Create a classification report

observed_result = []
actual_result = []
pos_and_neg_set = pos_test + neg_test

for i in range(len(pos_and_neg_set)):
    observed_result.append(classifier.classify(pos_and_neg_set[i][0]))
    actual_result.append(pos_and_neg_set[i][1])

clas_report = pd.DataFrame(classification_report(actual_result, observed_result, ou

# Create a confusion matrix for each measurement

conf_matrix = confusion_matrix(actual_result, observed_result)
conf_matrix_display = ConfusionMatrixDisplay(conf_matrix, display_labels=['pos','ne
conf_matrix_display.plot()

# Save and show the figure
plt.savefig("../Results/Figures/Confusion_matrix.png", dpi=1200)

plt.show()

# Show the most informative features for each measurement

classifier.show_most_informative_features()

# Save the results

accur_report["train_accuracy"] = [train_accuracy]
accur_report["test_accuracy"] = [test_accuracy]
```




```
Most Informative Features
          not_disappoint = 1              pos : neg    =    104.3 : 1.0
                     4.5 = 1              pos : neg    =     73.0 : 1.0
                 deleted = 1              neg : pos    =     62.1 : 1.0
           not_impressed = 1              neg : pos    =     52.3 : 1.0
              not_finish = 1              neg : pos    =     44.7 : 1.0
                  refund = 1              neg : pos    =     44.6 : 1.0
                  weaves = 1              pos : neg    =     43.0 : 1.0
                    lame = 1              neg : pos    =     42.2 : 1.0
                deleting = 1              neg : pos    =     41.7 : 1.0
           uninteresting = 1              neg : pos    =     38.3 : 1.0
```

In [ ]:

```
accur_report
```

Out[ ]:

	train_accuracy	test_accuracy
0	95.09375	90.015848

In [ ]:

```
clas_report
```

Out[ ]:

	precision	recall	f1-score	support
neg	0.903539	0.862537	0.882562	3019.0
pos	0.897758	0.929118	0.913169	3922.0