Text Mining

Natural Language Toolkit

https://www.nltk.org/

```
In [ ]: # Load up libraries
        import pandas as pd
        import string
        import nltk
        from nltk.corpus import stopwords
        from nltk.tokenize import word_tokenize
        from nltk.stem import WordNetLemmatizer
        from nltk.sentiment import SentimentIntensityAnalyzer

        # Download NLTK resources
        nltk.download('punkt')
        nltk.download('stopwords')
        nltk.download('wordnet')
        nltk.download('vader_lexicon')
```

```
In [ ]: # Load the dataframe
        review = pd.read_csv("bartini_reviews.csv")

        # remove na values

        review = review.dropna()


        # change Date to date value

        review["Date"] = pd.to_datetime(review["Date"])
        review
```

Tokenize

Splitting the text into individual words or "tokens"

```
In [ ]: review['tokens'] = review['Review'].apply(word_tokenize)
        review
```

Stopwords

Removing common words that do not carry significant meaning

```
In [ ]: stop_words = set(stopwords.words('english'))
        review['tokens'] = review['tokens'].apply(lambda x: [word for word in x if word.lower() not in s
        review
```

Remove Punctation

```
In [ ]: review['tokens'] = review['tokens'].apply(lambda x: [word for word in x if word not in string.pu
```

Lemmatization

Reducing words to their base or root form

```
In [ ]: lemmatizer = WordNetLemmatizer()
        review['tokens'] = review['tokens'].apply(lambda x: [lemmatizer.lemmatize(word) for word in x])
```

Basic Analysis

```
In [ ]: !pip install textblob
```

```
In [ ]: # load some libraries

        from collections import Counter
        import matplotlib.pyplot as plt
        from textblob import TextBlob
```

Word Frequency Analysis

```
In [ ]: word_freq = Counter(word for sublist in review['tokens'] for word in sublist)
        top_words = word_freq.most_common(10)
        print("Top 10 frequent words:", top_words)
```

```
In [ ]: # visualize

        # Visualization of Word Frequencies
        plt.bar(*zip(*top_words))
        plt.xlabel('Words')
        plt.ylabel('Frequency')
        plt.title('Top 10 Frequent Words')
        plt.xticks(rotation=45)
        plt.show()
```

Vader Sentiment Analyzer

https://vadersentiment.readthedocs.io/

```
In [ ]: # Initialize the VADER sentiment analyzer
        sid = SentimentIntensityAnalyzer()
```

```
In [ ]: # Function to analyze sentiment and return sentiment label
        def analyze_sentiment(tokens):
            text = ' '.join(tokens)
            scores = sid.polarity_scores(text)
            if scores['compound'] >= 0.05:
                return "Positive"
```

```
    elif scores['compound'] <= -0.05:
        return "Negative"
    else:
        return "Neutral"
```

In [ ]:
```
# Apply sentiment analysis to each row of the DataFrame
review['Opinion'] = review['tokens'].apply(analyze_sentiment)
```

In [ ]:
```
# Print the DataFrame with sentiment analysis results
print(review[['Review', 'Opinion']])
```

In [ ]:
```
#bar plot for opinion

import seaborn as sns

sns.countplot(data = review, x = "Opinion")

# Add title and labels
plt.title('Opinion Sentiment')
plt.xlabel('')
plt.ylabel('')

# Show plot
plt.show()
```

In [ ]:
```
# Apply sentiment analysis to each row of the DataFrame
review['Opinion'] = review['tokens'].apply(analyze_sentiment)

# Aggregate sentiment over time
review['month'] = review['Date'].dt.to_period('M')
sentiment_over_time = review.groupby(['month', 'Opinion']).size().unstack(fill_value=0).reset_in

# Convert period to datetime
sentiment_over_time['month'] = sentiment_over_time['month'].dt.to_timestamp()

# Melt the DataFrame for seaborn compatibility
sentiment_melted = sentiment_over_time.melt(id_vars='month', value_vars=['Positive', 'Negative',

# Plot the sentiment trends over time using seaborn
plt.figure(figsize=(12, 6))
sns.scatterplot(data=sentiment_melted, x='month', y='Count', hue='Sentiment', style='Sentiment',
plt.title('Sentiment Analysis Over Time')
plt.xlabel('Month')
plt.ylabel('Number of Reviews')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

In [ ]: