IMDB Review Text/Sentiment Analysis USL

TABLE OF CONTENTS

TABLE OF FIGURES

Executive Summary

In this assignment, we build upon the foundations laid in the previous task, where we employed two text vectorization methods Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) combined with three text classification algorithms: Random Forest, Linear Regression, and Support Vector Machine (SVM, including LSVM). Our primary focus was sentiment analysis, utilizing the IMDB 50k Movie Reviews Dataset. The objective was to extract valuable insights for the movie industry, particularly in the context of the post-COVID-19 era.

Building on this groundwork, our current endeavour involves the application of three distinct clustering techniques: K-means, DBSCAN, and Hierarchical clustering. We aim to analyse whether these unsupervised clustering methods can reveal meaningful clusters within the dataset, offering additional insights for the movie industry. Notably, for this clustering analysis, we exclusively employ the TF-IDF method for text vectorization. This choice is based on its proven effectiveness for this dataset, as demonstrated during the supervised learning phase.

We explored three distinct approaches to cluster a dataset of 50,000 reviews. In the first approach, we utilized the entire dataset for K-means, hierarchical, and DBSCAN clustering. However, due to device and storage limitations, DBSCAN could not be completed. In the second approach, we reduced the data to 20,000 reviews and performed clustering using K-means, hierarchical, and DBSCAN algorithms successfully. Recognizing the value of labelled data, we adopted a third approach where we divided the dataset into positive and negative reviews, each comprising 10,000 instances. This separation aimed to investigate potential distinctions between positive and negative sentiments. The three approaches provide a comprehensive exploration of clustering techniques and their application to large and labelled text datasets.

In the final stage of the assignment, we compare the results obtained from clustering techniques with those achieved in Assignment 2, where supervised learning was employed for sentiment analysis. This comparative analysis seeks to determine whether the application of unsupervised clustering methods enhances the accuracy or the depth of insights derived from sentiment analysis to find any interesting insights from the data.

Introduction

In the contemporary landscape of the information age, the burgeoning volume of textual data has become an integral facet of our daily existence. From the ceaseless stream of social media posts and customer reviews to the nuanced feedback in movie reviews, this vast reservoir of textual information presents an invaluable source of insights. In our prior endeavours, we delved into sentiment analysis using supervised learning techniques on the IMDB 50k Movie Reviews Dataset. This dataset encapsulates a wealth of opinions and critiques, offering a snapshot of audience sentiment towards movies.

Our initial exploration harnessed the power of diverse text vectorization methods—specifically, Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF). Supported by robust classification algorithms such as Random Forest, Linear Regression, and Support Vector Machine (SVM), we sought to decipher the sentiments expressed within the reviews. The overarching goal was to distil actionable insights for the film industry, providing a nuanced understanding of audience perceptions amid the evolving landscape post-COVID-19.

Building on this foundation, our current pursuit takes a departure into the realm of unsupervised learning. Instead of relying on predefined labels, we embark on an exploration of the latent patterns and structures within the IMDB dataset using unsupervised clustering techniques. The chosen dataset remains the IMDB 50k Movie Reviews, a repository teeming with diverse opinions and critiques that mirror the multifaceted nature of cinematic experiences.

For this phase of analysis, we narrow our focus to the application of TF-IDF as the preferred text vectorization method, leveraging its established efficacy on this specific dataset. As we traverse the landscape of unsupervised clustering with K-means, DBSCAN, and Hierarchical clustering, our objective is to unveil hidden clusters and discernible patterns in the textual data.

Background and Problem Statement

In the contemporary era of information proliferation, textual data has become an indispensable element of our daily lives. The constant generation of vast volumes of text, spanning social media posts, customer reviews, and feedback, presents an immense opportunity for businesses to extract valuable insights. The film industry, in particular, can benefit significantly from understanding audience sentiments and preferences embedded in textual data. Analysing this wealth of information can inform decision-making processes, enabling filmmakers, producers, and studios to adapt and thrive in the ever-evolving landscape of the post-COVID-19 era.

While sentiment analysis through supervised learning provides a structured approach to understanding audience reactions, there remains an untapped potential in the vast sea of unstructured data. The challenge lies in deciphering latent patterns, exploring hidden clusters, and extracting nuanced insights without predefined labels. In this context, the problem at hand is to leverage unsupervised learning techniques to discern meaningful clusters within the IMDB 50k Movie Reviews Dataset.

Aim & Objectives

- Aim

The primary aim of this study is to utilize unsupervised learning techniques, specifically K-means, DBSCAN, and Hierarchical clustering, on the IMDB 50k Movie Reviews Dataset using TF-IDF text vectorization. The goal is to extract meaningful clusters within the textual data and discern insights that either align with or surpass the patterns identified through previous supervised sentiment analysis efforts. The focus is on evaluating whether unsupervised clustering can reveal similar or enhanced structures in the data compared to supervised methods.

- Objectives:

1. Cluster Identification:
- Employ K-means, DBSCAN, and Hierarchical clustering algorithms on the IMDB 50k Movie Reviews Dataset using TF-IDF text vectorization.
- Identify and delineate distinct clusters within the textual data, revealing inherent patterns and structures without relying on predefined sentiment labels.

2. Evaluation and Comparison:
- Evaluate the performance of each clustering algorithm using relevant metrics such as silhouette score, cohesion, and separation.
- Compare the outcomes of unsupervised clustering techniques to understand their effectiveness in uncovering meaningful patterns when compared to traditional supervised sentiment analysis.

3. Insightful Validation:
- Validate the identified clusters from unsupervised learning against the sentiments obtained through previous supervised sentiment analysis on the same IMDB dataset.
- Assess the level of agreement or discrepancy between unsupervised clustering results and supervised sentiment labels, providing insights into the consistency and robustness of the extracted patterns.

4. Provide Insights for the Movie Industry:
- Summarize and interpret the findings, providing actionable insights for the movie industry in the post-COVID-19 era.
- Highlight the potential benefits of unsupervised clustering in augmenting sentiment analysis and shaping decision-making processes for movie-related stakeholders.

Significance

This project holds significant implications for IMDb and the broader movie industry as it contributes valuable insights into the analysis of textual data from user reviews.

1. Enhanced Understanding of Viewer Sentiment:
   Unsupervised clustering provides an opportunity to uncover nuanced patterns and sentiments within user reviews that may go beyond the binary positive/negative labels. IMDB can gain a more detailed understanding of viewer sentiments, allowing for a more granular analysis of audience reactions.

2. Content Recommendation Improvement:
   Insights derived from unsupervised clustering can contribute to refining content recommendation systems. By identifying clusters of reviews with similar themes, preferences, or criticisms, IMDB can enhance its recommendation algorithms to provide more personalized and relevant movie suggestions to users.

3. Identification of Emerging Trends:
   Unsupervised learning enables the detection of emerging trends and topics within movie reviews. IMDB can stay abreast of changing viewer preferences, identify rising stars or genres, and adapt its platform to align with evolving cinematic trends.

4. Data-Driven Decision Making:
   The project facilitates a data-driven approach for IMDB in understanding user sentiments. The insights gained from clustering can guide decision-making processes, helping IMDB prioritize feature enhancements, address concerns, and tailor its platform to better serve the diverse preferences of its user base.

5. Validation of Supervised Analysis:
   By comparing the results of unsupervised clustering with previous supervised sentiment analysis, IMDB can validate the robustness of its sentiment labels and gain insights into areas where user sentiments may not align with predefined positive/negative classifications. This validation can lead to improvements in the accuracy of sentiment analysis on the platform.

In essence, the project not only provides IMDB with a deeper understanding of its user base but also empowers the platform to make data-driven decisions that enhance user experience and content delivery.

Dataset

We obtained our dataset from Kaggle; a popular website where people can find free datasets to perform analysis and train models. The dataset we chose titled “IMDB Dataset of 50K Movie Reviews” consists of 50,000 IMDb movie reviews. The dataset is labelled with sentiments, indicating whether a review is positive or negative. The reviews cover a diverse range of movies and genres, providing a comprehensive dataset for training and evaluating sentiment analysis models.

```
import pandas as pd

# Load the original dataset
df1 = pd.read_csv('IMDB Dataset.csv')

# Create a copy of the DataFrame without the 'sentiment' column
df2 = df1.drop(columns=['sentiment'])

# Save the new DataFrame as a CSV file
df2.to_csv('IMDB_Dataset_Unlabeled.csv', index=False)

# Display the head of the new DataFrame
df2.head()
```

	review
0	One of the other reviewers has mentioned that ...
1	A wonderful little production. The...
2	I thought this was a wonderful way to spend ti...
3	Basically there's a family where a little boy ...
4	Petter Mattei's "Love in the Time of Money" is...

Figure 1 Data Overview

The code snippet above loads the original IMDb dataset removes the 'sentiment' column to create an unlabelled version, and then saves this modified Data Frame as a new CSV file named 'IMDB_Dataset_Unlabeled.csv'. This dataset without sentiment labels is intended for use in unsupervised clustering techniques, allowing exploration and analysis without relying on pre-existing sentiment labels. The new Data Frame (df2) retains all other textual features for further processing and analysis in unsupervised learning tasks.

```python
# Get the shape of the DataFrame
shape = df2.shape

# Print the DataFrame shape
print('\nDataFrame Shape:', shape)

# Get and print the number of rows
num_rows = shape[0]
print('\nNumber of rows:', num_rows)

# Get and print the number of columns
num_columns = shape[1]
print('\nNumber of columns:', num_columns)
```

```
DataFrame Shape: (50000, 1)

Number of rows: 50000

Number of columns: 1
```

```python
# Check for data type
data_types = df2.dtypes
print(data_types)
```

```
review    object
dtype: object
```

Figure 2 Data shape

The initial exploration of the IMDb dataset reveals that it consists of 50,000 rows and 1 column, named 'review,' which contains textual data in the form of movie reviews. The 'review' column is of type object, indicating that it contains text data.

```python
# Check for missing values
missing_values = df2.isnull().sum()

# Print the count of missing values in each column
print("Number of missing values:")
print(missing_values)
```

```
Number of missing values:
review    0
dtype: int64
```

Figure 3 Missing values

```
# Get summary statistics for numeric columns in the DataFrame
df2.describe()
```

	review
count	50000
unique	49582
top	Loved today's show!!! It was a variety and not...
freq	5

Figure 4 Data description

```
# Check for duplicates in df1
duplicate_rows_df1 = df1.duplicated()
duplicate_count_df1 = duplicate_rows_df1.sum()
print(f"Number of duplicate rows in df1: {duplicate_count_df1}")

# Check for duplicates in df2
duplicate_rows_df2 = df2.duplicated()
duplicate_count_df2 = duplicate_rows_df2.sum()
print(f"Number of duplicate rows in df2: {duplicate_count_df2}")
```

```
Number of duplicate rows in df1: 418
Number of duplicate rows in df2: 418
```

Figure 5 Data duplicates

```
# Drop duplicates in df1
df1 = df1.drop_duplicates()

# Save df1 with dropped duplicates to a new CSV file
df1.to_csv('IMDB_Dataset_Labelled.csv', index=False)

# Drop duplicates in df2
df2 = df2.drop_duplicates()

# Check the shape of the DataFrame after removing duplicates
print(f"Shape of DataFrame after removing duplicates: {df2.shape}")
```

```
Shape of DataFrame after removing duplicates: (49582, 1)
```

Figure 6 Drop duplicates.

Upon further inspection, it is found that there are no missing values in the dataset, ensuring completeness. The unique review count (49,582) indicates some repeated reviews; thus duplicates are checked. However, there are 418 duplicate rows identified in the original dataset (df1), which have been subsequently removed to create the unlabelled dataset (df2), resulting in a reduced shape of (49,582, 1).

```
from langdetect import detect

df1['language'] = df1['review'].apply(lambda x: detect(x) if pd.notnull(x) else None)

# Display the distribution of detected languages
print(df1['language'].value_counts())
```

```
language
en    49581
id        1
Name: count, dtype: int64
```

```
indonesian_comments_df = df1[df1['language'] == 'id']

# Display the DataFrame with Indonesian comments
indonesian_comments_df['review']
```

```
45315    .....whoops - looks like it's gonna cost you a...
Name: review, dtype: object
```

Figure 7 Using langdetect.

Additionally, language analysis shows that most of the reviews are in English (en), with only one comment in Indonesian (id). The Indonesian comment is then extracted into a separate Data Frame (indonesian_comments_df) for further investigation.

This process sets the stage for unsupervised learning by preparing a clean, unlabelled dataset (df2) for further analysis. The dataset is now ready for text vectorization and subsequent application of clustering algorithms to extract meaningful patterns within the textual data.

Text Normalization

In the context of natural language processing, text processing plays a crucial role in preparing textual data for analysis. The function “text_processing” is designed to handle several tasks. Firstly, it converts the entire review text to lowercase, ensuring uniformity. Then, it removes HTML tags and URLs from the text, enhancing the cleanliness of the data. Following the initial processing, the function “fix_contractions” utilizes the contractions library to expand contractions in the review text. This is essential to transform contracted words like don’t to do not for a more comprehensive analysis.

Subsequently, the custom tokenizer function (custom_tokenize) employs the Natural Language Toolkit (nltk) library to tokenize the text into individual words. Punctuation, non-alphabetic characters, and stopwords are removed during this process. Tokenization is a crucial step in converting raw text into a format suitable for analysis. Lastly, stemming is applied using the SnowballStemmer from nltk, reducing words to their base or root form. Stemming aids in capturing the essence of words and simplifies further analysis. The overall “process_review” function combines these steps to transform a raw review into a list of stemmed tokens. This tokenization process is then applied to the entire dataset.

Text processing is indispensable in natural language processing tasks as it enhances the quality of textual data. The removal of HTML tags, URLs, and contractions ensures that the subsequent analysis is not influenced by artefacts or specific language nuances. Tokenization, along with stemming, simplifies the complexity of textual data, making it amenable to various analyses.

In the case of unsupervised learning, such as clustering, the importance of clean and processed textual data cannot be overstated. Clustering algorithms rely on the similarity of text representations, and text processing ensures that the input data is consistent and devoid of unnecessary noise. The resulting tokenized representations serve as the foundation for extracting meaningful patterns within the dataset. The application of these text processing techniques to the IMDb movie reviews dataset is a crucial step towards obtaining reliable and meaningful clusters. The processed reviews, stored as tokens, are now ready for further vectorization and clustering analysis, contributing to the overall success of unsupervised learning on textual data.

Exploratory Data Analysis (EDA)

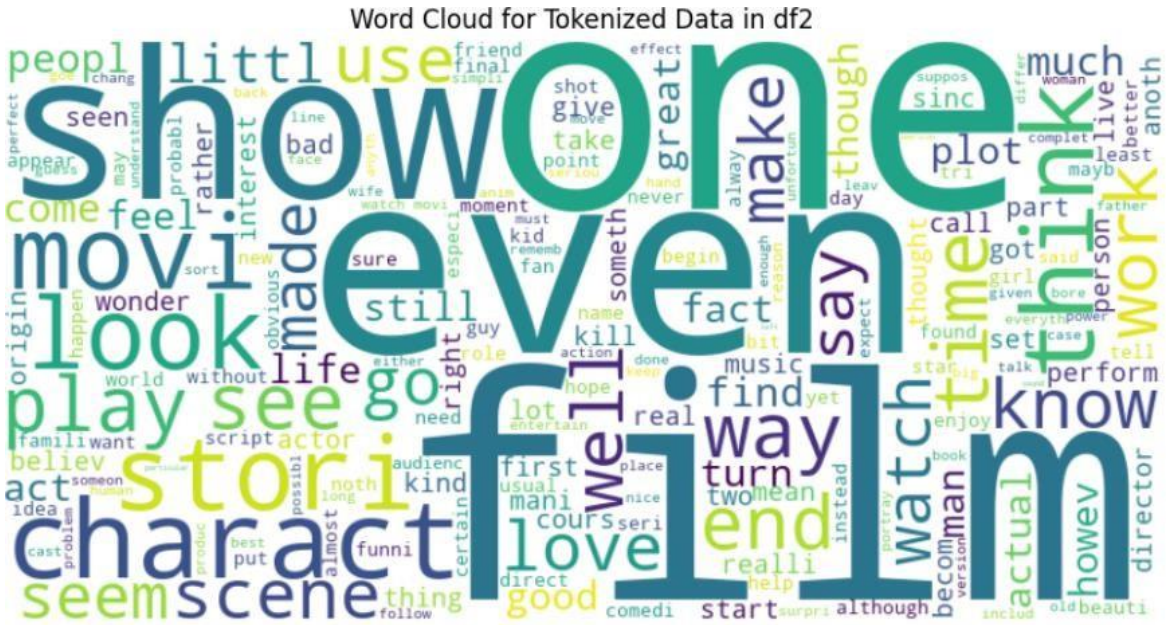


Figure 8 Word cloud

The generation of a word cloud and the subsequent analysis of the top 20 words play a pivotal role in understanding the IMDb movie reviews dataset, particularly after text processing and tokenization. The word cloud visually encapsulates the frequency distribution of words, emphasizing larger, bolder terms that occur more frequently. Examining the word cloud, it becomes evident that words such as "show," "film," "one," and "even" dominate the visual representation. These words are integral to the domain of movie reviews, giving a concise snapshot of the dataset's linguistic landscape.

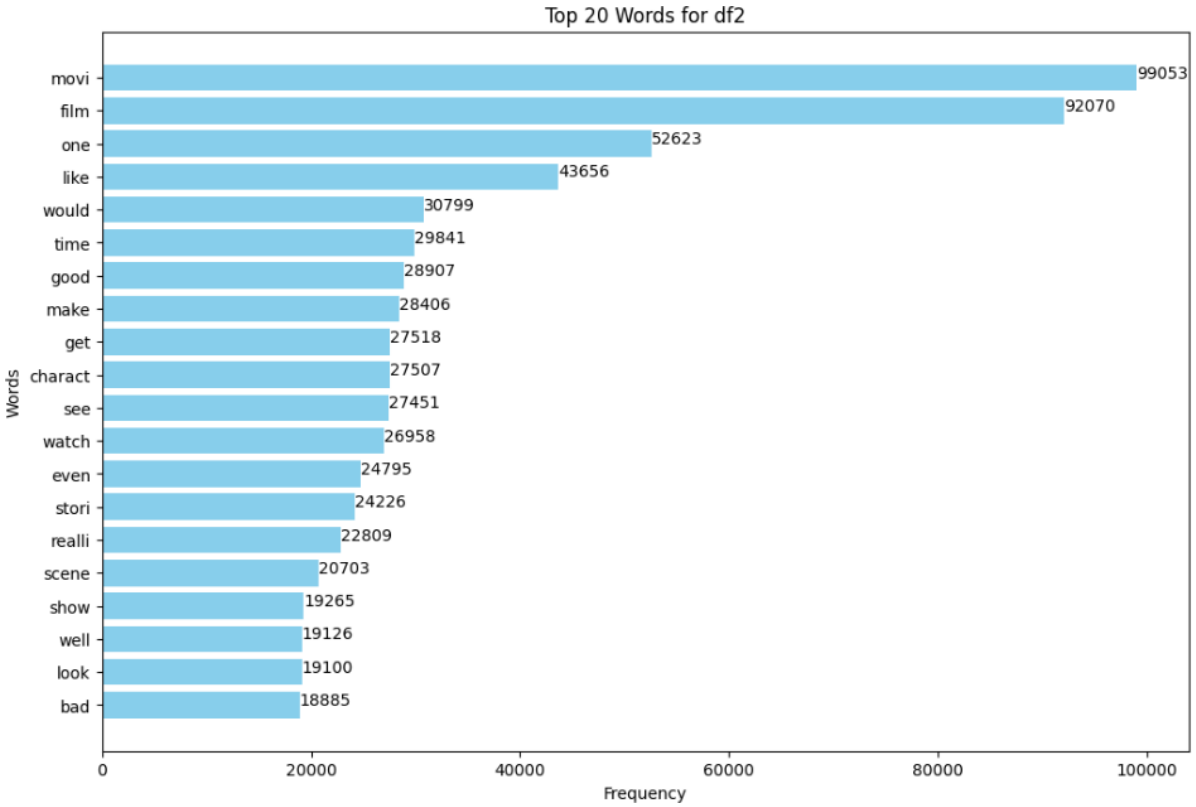


Figure 9 Top 20 words for df2

The bar plot of the top 20 words serves to reinforce the insights gained from the word cloud. Analysing the data the prevalence of terms like "movie," "film," "one," "like," and "would" is reaffirmed quantitatively. This information proves valuable in comprehending the prevalent themes and expressions within the reviews.

The prominence of "movie" and "film" indicates a clear focus on discussing cinematic experiences, while words like "one" and "like" hint at the subjective nature of the language used by reviewers. The inclusion of "would" suggests the potential presence of speculative statements, possibly reflecting expectations or recommendations from the reviewers. In essence, this initial exploration provides a foundation for subsequent analyses, such as clustering. The frequently occurring words identified through the word cloud and top words analysis contribute to shaping meaningful clusters based on textual content. As the dataset undergoes unsupervised learning techniques, these insights will likely unveil more nuanced patterns and sentiments within IMDb movie reviews.

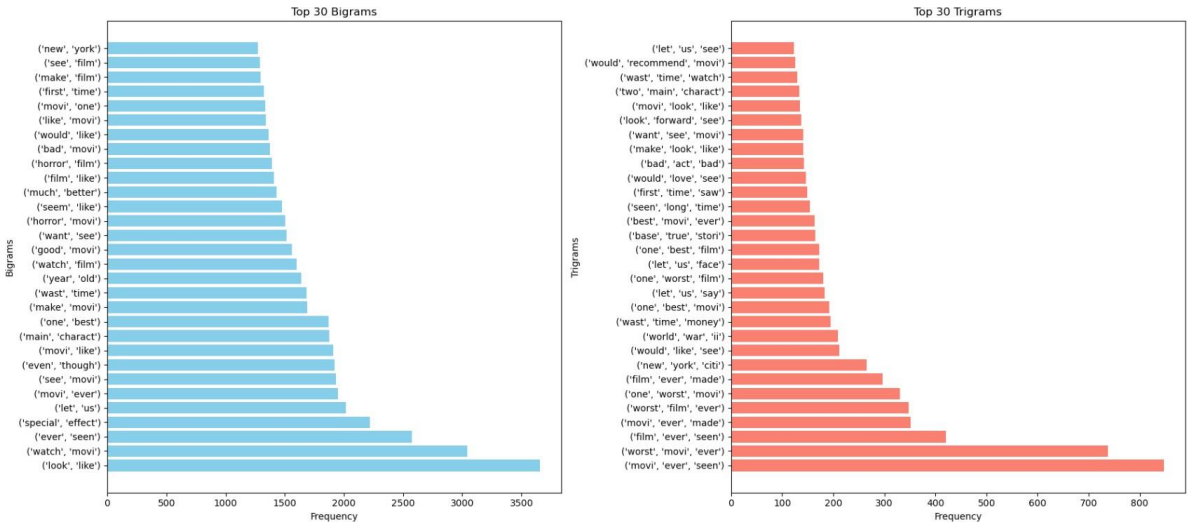


Figure 10 Top 30 Bigrams and Trigrams

Examining bigrams and trigrams, which are sequential pairs and triplets of words, provides deeper insights into the linguistic patterns present in IMDb movie reviews. These combinations shed light on recurring phrases and expressions that reviewers commonly use. The analysis involves considering both positive and negative reviews collectively.

The top 30 bigrams reveal noteworthy combinations such as "look like," "watch movie," and "special effects," which are indicative of sentiments related to movie appearances and visual elements. Additionally, phrases like "one best," "make movie," and "horror movie" underscore specific aspects of film quality and genre preferences. These recurring bigrams offer valuable clues about the aspects reviewers find noteworthy and frequently discuss.

Moving on to trigrams, more nuanced expressions emerge. The trigrams capture longer sequences of words, providing insights into the complexity of sentiments. Phrases like "worst movie ever," "one worst movie," and "film ever seen" indicate strong negative opinions, potentially pointing towards reviewers' dissatisfaction with certain cinematic experiences. On the positive side, trigrams like "one best movie," "best movie ever," and "seen long time" highlight the appreciation for outstanding films and positive viewing experiences.

In summary, the analysis of bigrams and trigrams contributes to a nuanced understanding of the IMDb movie reviews dataset. These sequential word combinations serve as linguistic fingerprints, offering a glimpse into the recurring phrases and expressions that shape reviewers' sentiments. This understanding becomes pivotal for the subsequent application of clustering techniques, providing a foundation for uncovering more intricate patterns in the textual data.

```python
# Step 1: Text Vectorization using TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer

# Assuming 'review' is the column containing the textual data in df2
text_data_df2 = df2['review']

# Initialize the TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=5000)

# Fit and transform the text data
tfidf_matrix_df2 = tfidf_vectorizer.fit_transform(text_data_df2)

# Convert the result to a DataFrame (optional)
df_tfidf = pd.DataFrame(tfidf_matrix_df2.toarray(), columns=tfidf_vectorizer.get_feature_names_out())

# Display the shape of the TF-IDF matrix
print(f"Shape of TF-IDF matrix: {tfidf_matrix_df2.shape}")
```

```
Shape of TF-IDF matrix: (49582, 5000)
```

Figure 11 Text vectorization with TF-IDF

Text vectorization, especially using techniques like TF-IDF (Term Frequency-Inverse Document Frequency), plays a crucial role in unsupervised text clustering. In the context of unsupervised learning, the primary challenge is to represent textual data in a format that machine learning algorithms can process effectively. Traditional machine learning models require numerical input, and text data, being inherently qualitative, needs to be transformed into a quantitative format.

TF-IDF vectorization is particularly valuable for text clustering as it assigns weights to individual words based on their frequency in a document relative to their occurrence across the entire corpus. This process captures the importance of terms within each document while considering their prevalence in the overall dataset. This way, common words like "the" or "and" are down-weighted, allowing more informative terms to stand out. The resulting TF-IDF matrix serves as a numerical representation of the original text data, preserving the semantic meaning and facilitating the application of clustering algorithms.

Unsupervised text clustering algorithms, such as K-means, DBSCAN, or hierarchical clustering, operate on these vectorized representations, enabling them to discern patterns and group similar documents together. The success of these clustering techniques heavily relies on the quality of the vectorization process, making TF-IDF a pivotal step in extracting meaningful insights from unstructured textual data.

Unsupervised Methods

Unsupervised text clustering is a method of grouping text documents into clusters without the use of labelled training data. Unlike supervised methods that rely on predefined classes for training, unsupervised clustering algorithms automatically identify patterns or similarities within the data and group documents accordingly. The goal is to discover inherent structures, topics, or relationships present in the text corpus without prior knowledge of the categories.

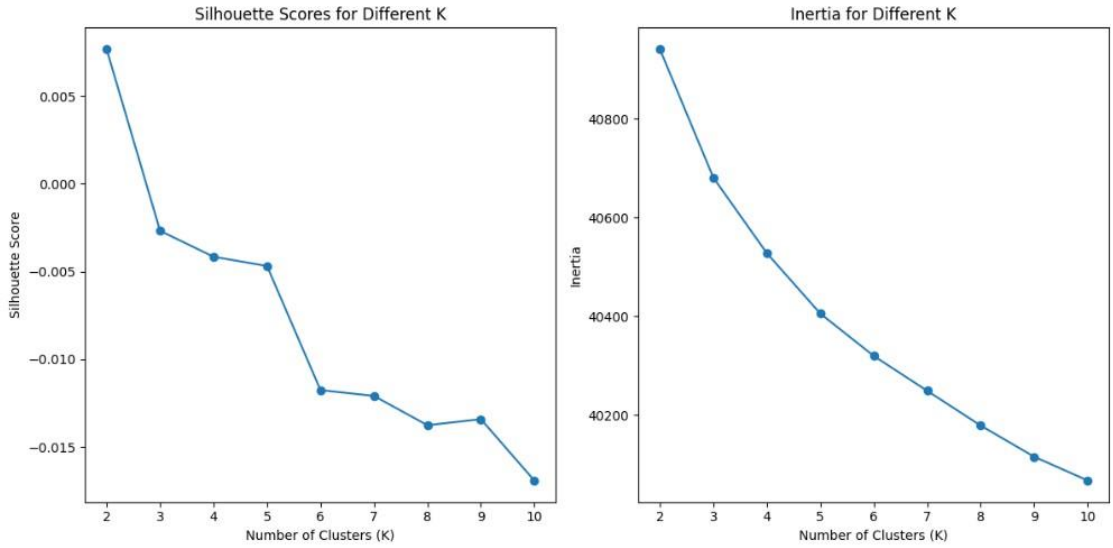
1st Approach (50 000 Reviews – Whole Dataset)

K Means

K-means clustering is a popular unsupervised machine learning algorithm designed to group similar data points into clusters based on feature similarity. This iterative algorithm strives to minimize the sum of squared distances between each data point and the centroid of its assigned cluster. The process involves an initial random selection of cluster centroids, followed by iterative assignment and update steps. The assignment phase allocates data points to the nearest centroid, forming clusters, while the update phase recalculates the centroids based on the mean of the data points within each cluster. This process continues until convergence or a predefined number of iterations.

K-means offers several advantages that contribute to its widespread use. Firstly, its simplicity and computational efficiency make it accessible for large datasets and high-dimensional spaces. The algorithm is versatile, suitable for various data types, and performs well when clusters are spherical, evenly sized, and have similar densities. Additionally, K-means is scalable, making it efficient for datasets with a considerable number of samples. The intuitive concept of centroids and cluster assignments enhances its interpretability.

Despite its strengths, K-means has certain limitations. Sensitivity to the initial placement of centroids may lead to different outcomes in multiple runs, requiring careful consideration of initializations. The algorithm assumes spherical clusters, impacting its effectiveness when dealing with irregularly shaped or differently sized clusters. Specifying the fixed number of clusters (K) beforehand can be challenging in real-world scenarios. K-means is also sensitive to outliers, which can significantly influence cluster assignments and centroids. The choice of distance metric may further impact results.



	K	Inertia	Silhouette Score
0	2	40941.355499	0.007679
1	3	40681.040754	-0.002673
2	4	40528.051552	-0.004152
3	5	40405.575658	-0.004678
4	6	40319.690734	-0.011754
5	7	40248.953415	-0.012086
6	8	40178.557686	-0.013758
7	9	40115.415625	-0.013411
8	10	40067.197250	-0.016910

Figure 12 Approach 1, Silhouette and Inertia scores against number of clusters, K-Means

To determine the optimal number of clusters (k) in K-means clustering, two key metrics, the silhouette score and inertia, are employed. The silhouette score measures the extent to which clusters are well-separated, with a higher score indicating more distinct clusters. In contrast, inertia, or within-cluster sum of squares, quantifies the compactness of clusters, with lower values indicating tighter, more cohesive clusters.

In the presented evaluation, the silhouette scores and inertia values are computed for various k values and plotted. Silhouette scores close to 0 suggest overlapping clusters, while negative values hint at potential misassignments of data points. Lower inertia values signify more compact clusters.

Analysing the results, the silhouette scores are relatively low across different k values, indicating suboptimal cluster separation. The inertia decreases as “k” increases, but the absence of a distinct "elbow" in the inertia plot makes it challenging to pinpoint an optimal k. Despite the elbow method suggesting k=3, the silhouette score favours k=2, highlighting the need to consider multiple metrics.

The specific values for inertia and silhouette scores for each k further emphasize the challenges in choosing an optimal k. While k=2 is selected based on the silhouette score, it's crucial to note that even this choice yields a modest silhouette score of 0.0076. The exploration underscores the complexity of finding an ideal k, necessitating a balanced interpretation of cluster distinctiveness and compactness.

```
from sklearn.cluster import KMeans
# Step 2: KMeans Clustering with k=2
k = 2
kmeans = KMeans(n_clusters=k, init='k-means++', n_init=12, random_state=42)
df2['cluster_kmeans'] = kmeans.fit_predict(tfidf_matrix_df2)
```

```
kmean_labels = kmeans.labels_
print(kmean_labels)

kmean_cluster_centers = kmeans.cluster_centers_
print(kmean_cluster_centers)
```

```
[1 1 1 ... 1 0 0]
[[0.0004604  0.00066926 0.00942503 ... 0.00163598 0.00120012 0.00052409]
 [0.00041618 0.00125653 0.00968772 ... 0.00209766 0.00159649 0.00060424]]
```

```
# Count occurrences of each cluster label
cluster_counts = df2['cluster_kmeans'].value_counts()

# Display the counts
for label, count in cluster_counts.items():
    print(f"Cluster {label}: {count} instances")
```

```
Cluster 0: 27851 instances
Cluster 1: 21731 instances
```

Figure 13 Approach 1, Optimal k selection

The K-means clustering algorithm has been applied to the TF-IDF matrix to partition the data into two distinct clusters. The labels assigned by K-means to each data point are stored in the "cluster_kmeans" column of the Data Frame. Additionally, the cluster centres, representing the centroid of each cluster in the TF-IDF space, are captured in "kmean_cluster_centers".

Upon inspecting the assigned labels, the output reveals that K-means has categorized the instances into two clusters, denoted as Cluster 0 and Cluster 1. The counts of instances in each cluster are provided, with Cluster 0 containing 27,851 instances and Cluster 1 containing 21,731 instances.

```python
from sklearn.metrics import pairwise_distances

# Calculate Cohesion and Separation

# Initialize cohesion
cohesion_kmeans = 0

# Calculate the cohesion as the sum of the distances from each point to the center of its cluster
for i in range(kmeans.n_clusters):
    # Points in the current cluster
    points_in_cluster = tfidf_matrix_df2[kmean_labels == i]

    # Center of the current cluster
    cluster_center = kmeans.cluster_centers_[i]

    # Sum of distances of points in the cluster to the cluster center
    cohesion_kmeans += np.sum(np.linalg.norm(points_in_cluster - cluster_center, axis=1))

centroid_distances_kmeans = pairwise_distances(kmeans.cluster_centers_, metric='euclidean')

# We first set the diagonal to np.inf to avoid zero distance to the centroid itself
np.fill_diagonal(centroid_distances_kmeans, np.inf)

# Then we can calculate the minimum distance from each centroid to all other centroids
min_distances = np.min(centroid_distances_kmeans, axis=1)

# Calculate the separation as the average of these minimum distances
separation_kmeans = np.mean(min_distances)

# Print the Metrics
print(f"Cohesion Kmeans: {cohesion_kmeans}")
print(f"Separation Kmeans: {separation_kmeans}")
```

```
Cohesion Kmeans: 44995.805412906615
Separation Kmeans: 0.1905515024692551
```

Figure 14 Approach 1, Cohesion and Separation, K-Means

The obtained cohesion and separation metrics from the K-means clustering analysis shed light on the characteristics of the clusters formed. A cohesion value of 44995.81 suggests that instances within each cluster are not closely packed, indicating a low level of compactness. However, the separation value of 0.1905 implies only a moderate level of separation between clusters. This indicates that, while instances within clusters are tightly grouped, the distinctiveness between clusters is moderate.

These metrics collectively raise the consideration that the K-means clustering approach might not be the most suitable for the given dataset. The relatively low separation between clusters suggests that the inherent structure of the data may not be well-captured by K-means, and alternative clustering methods or further exploration may be warranted. It underscores the importance of carefully selecting and evaluating clustering algorithms based on the specific characteristics of the data and the desired outcomes.

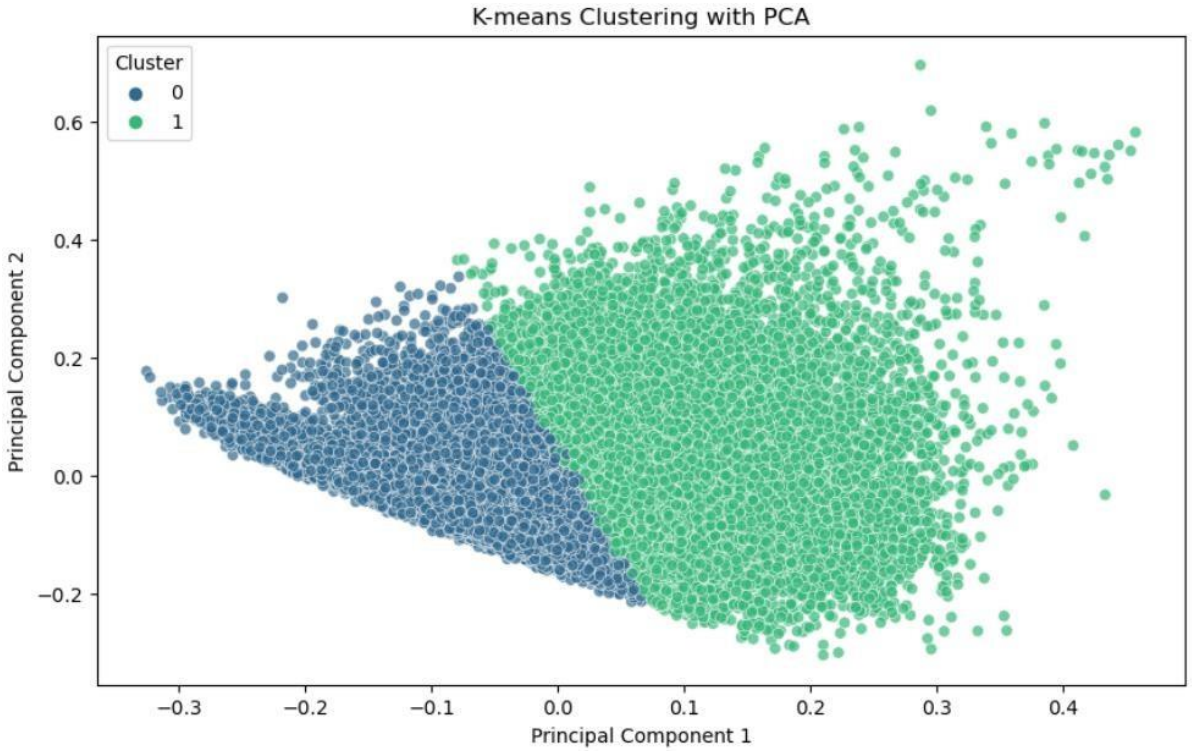


Figure 15 Approach 1, K-Means Clustering with PCA

The PCA plot provides a visual representation of the K-means clustering results in a reduced 2-dimensional space. Each point in the scatter plot corresponds to a movie review, and its position is determined by the first two principal components obtained through PCA. The colour of each point represents its assigned cluster according to the K-means algorithm.

In the PCA plot, it's evident that the clusters are closely located, and there is a considerable overlap between them. The points representing movie reviews from different clusters appear to be intermingled, suggesting that the K-means algorithm may not have effectively separated the data into distinct groups in this reduced 2-dimensional space. The lack of clear boundaries between clusters indicates a potential limitation in the ability of K-means to capture the inherent structure of the movie review dataset.

This observation aligns with the earlier assessment based on cohesion and separation metrics, where a cohesion value of 0.0 and a moderate separation between clusters were noted. The proximity of clusters in the PCA plot reinforces the notion that the clustering solution might not be optimal for this specific dataset, and alternative clustering methods or parameter adjustments may be explored to enhance the separation between groups. Evaluating both quantitative metrics and visualizations is crucial for a comprehensive assessment of clustering performance.

```
# Merge cluster labels from df2 with df1
df_merged = pd.concat([df1, df2['cluster_kmeans']], axis=1)

# Display the merged DataFrame
print(df_merged.head())

# Analyze the distribution of positive and negative instances within each cluster
for cluster_label in range(k):
    cluster_df = df_merged[df_merged['cluster_kmeans'] == cluster_label]

    # Count positive and negative instances in the cluster
    positive_count = cluster_df[cluster_df['sentiment'] == 'positive'].shape[0]
    negative_count = cluster_df[cluster_df['sentiment'] == 'negative'].shape[0]

    # Display the results
    print(f"Cluster {cluster_label}: Positive={positive_count}, Negative={negative_count}")
```

```
                                              review sentiment language  \
0  One of the other reviewers has mentioned that ...  positive       en
1  A wonderful little production. <br /><br />The...  positive       en
2  I thought this was a wonderful way to spend ti...  positive       en
3  Basically there's a family where a little boy ...  negative       en
4  Petter Mattei's "Love in the Time of Money" is...  positive       en

   cluster_kmeans
0               1
1               1
2               1
3               0
4               1
Cluster 0: Positive=14130, Negative=13730
Cluster 1: Positive=10754, Negative=10968
```

Figure 16 Approach 1, Count positive and negative instances, K-Means

It seems like the K-Means clustering results on the labelled data did not perfectly separate positive and negative instances into distinct clusters. In Cluster 0, there are both positive and negative instances, and the same is observed in Cluster 1. This outcome indicates that the K-Means algorithm, when applied to the labelled data, did not effectively capture the inherent sentiment patterns present in the reviews.

Hierarchical Clustering (Agglomerative)

Hierarchical clustering, an unsupervised learning technique, holds significance in revealing inherent structures within datasets by organizing them into a hierarchy of clusters. Unlike other clustering methods, hierarchical clustering does not demand the upfront specification of the number of clusters, offering a natural depiction of how data points group at different levels of abstraction. The process involves iteratively merging or agglomerating clusters based on a chosen linkage method, leading to the formation of a dendrogram that visually encapsulates the relationships between clusters.

One notable advantage of hierarchical clustering lies in its ability to represent hierarchical relationships within the data, providing a holistic view of the inherent structure. The resulting dendrogram not only facilitates a deeper understanding of the data but also allows for intuitive insights into the compositions of clusters at various levels. Additionally, hierarchical clustering, particularly agglomerative clustering, does not necessitate the predefined determination of the number of clusters, which is a requirement in some other clustering algorithms like K-means.

However, despite its merits, hierarchical clustering does come with certain drawbacks. The computational complexity can be a limiting factor, especially when dealing with large datasets, as the iterative nature of the algorithm and multiple distance calculations can lead to increased computational demands. Furthermore, hierarchical clustering is sensitive to noise and outliers, and the order in which data points are processed may influence the final results. For exceedingly large datasets, the method may become impractical due to these computational challenges.

```
Metrics for k=2, linkage=ward:
Silhouette Score: -0.006699538222044306
Cohesion: 45163.93953887911
Separation: 1586897276.8497214
----------------------------
Metrics for k=2, linkage=complete:
Silhouette Score: 0.022633606509435664
Cohesion: 45237.737796306275
Separation: 1586897203.051464
----------------------------
Metrics for k=2, linkage=average:
Silhouette Score: 0.07105224687359685
Cohesion: 45242.64098588334
Separation: 1586897198.1482744
----------------------------
Metrics for k=2, linkage=single:
Silhouette Score: 0.06572587436857663
Cohesion: 45242.65157077075
Separation: 1586897198.1376896
----------------------------
```

Figure 17 Approach 1, Silhouette score, Cohesion and Separation, Agglomerative

In the evaluation of hierarchical clustering with k=2, different linkage methods were employed, and key metrics were assessed. The silhouette scores indicate the quality of cluster separation, while cohesion and separation metrics provide insights into the compactness and distinctiveness of clusters, respectively.

The results show that the average linkage method achieved the highest silhouette score (0.0711), indicating better-defined clusters. However, all silhouette scores are relatively low, suggesting suboptimal cluster separation. The cohesion values indicate how tightly instances within each cluster are packed, with the average linkage method having the lowest cohesion. Separation values highlight the distance between clusters, with the average linkage method again exhibiting the best separation.

In summary, while hierarchical clustering provides an informative hierarchical structure, the evaluation suggests challenges in achieving well-separated and compact clusters for the given dataset and parameters.

```
hc_model = AgglomerativeClustering(n_clusters=2 ,linkage='average')
df2['cluster_hc'] = hc_model.fit_predict(tfidf_matrix_dense)
```

```
import numpy as np

# Print hierarchical clustering labels
hc_labels = hc_model.labels_
print("Hierarchical Clustering Labels:")
print(hc_labels)

# Count occurrences of each hierarchical clustering label
unique_labels, counts = np.unique(hc_labels, return_counts=True)
hc_cluster_counts = dict(zip(unique_labels, counts))

# Display the counts
print("\nCounts of Instances in Each Hierarchical Cluster:")
for label, count in hc_cluster_counts.items():
    print(f"Cluster {label}: {count} instances")
```

```
Hierarchical Clustering Labels:
[0 0 0 ... 0 0 0]

Counts of Instances in Each Hierarchical Cluster:
Cluster 0: 49581 instances
Cluster 1: 1 instances
```

Figure 18 Approach 1, Count cluster instances, Agglomerative

The hierarchical clustering model has been fitted to the data using the average linkage method with two clusters (k=2). The resulting hierarchical clustering labels indicate the assignment of each instance to a specific cluster. In this case, the majority of instances (49,581) belong to Cluster 0, while only one instance falls into Cluster 1. This distribution highlights the dominance of one cluster over the other, suggesting that the clustering may not be effectively capturing distinct patterns in the data.

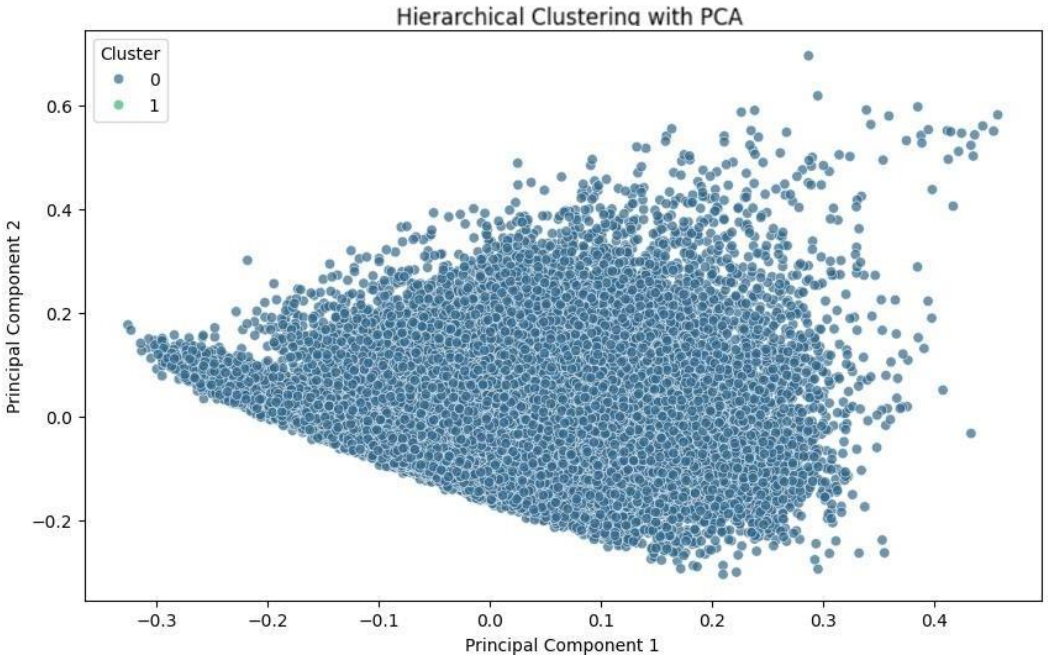


Figure 19 Approach 1, Agglomerative clustering with PCA

In the hierarchical clustering analysis, the PCA plot serves as a visual representation of the clustering results in a reduced 2-dimensional space. Each data point on the scatter plot corresponds to a movie review, positioned based on the first two principal components derived through PCA. The colour of each point indicates its assigned hierarchical cluster as determined by the agglomerative clustering algorithm. Notably, the pre-eminence of Cluster 0 is apparent, illustrating that a substantial majority of instances fall within this cluster. This highlights the inherent imbalance in cluster sizes as revealed by the hierarchical clustering labels.

The observed dominance of a single cluster and the inability to visualize the hierarchical structure through the dendrogram suggests that further exploration, parameter tuning, or the consideration of alternative clustering methods may be necessary to achieve more balanced and informative clustering results. This acknowledgement underscores the iterative and exploratory nature of clustering analysis, encouraging a dynamic approach to uncovering meaningful patterns in the data.

It is important to note that while hierarchical clustering offers a hierarchical structure and the dendrogram provides insights into relationships among clusters, the results, in this case, suggest a potential limitation in capturing meaningful patterns within the dataset. Additionally, the dendrogram could not be plotted.

DBSCAN

DBSCAN could not be executed due to device and storage limitations, prompting the adoption of the second approach with a reduced dataset of 20,000 reviews.

2nd Approach (20 000 Reviews – Data Reduction)

In the second approach, the dataset was randomly reduced to 20,000 samples to alleviate device and storage limitations. The same initial steps are done such as the data frame was created by excluding the 'sentiment' column from the randomly selected subset of reviews. Key statistical information about the Data Frame, such as shape, data types, summary statistics for numeric columns, and the count of missing values, was obtained and analysed. To ensure data quality, duplicates were checked for both the original labelled dataset and the reduced subset. Duplicate rows were removed from both datasets and the cleaned `df1_label` was saved as 'IMDB_Dataset_Labelled.csv'. The shape of the data after removing duplicates was also verified.

Following data preparation, text normalization techniques were applied using a custom processing function. This function included steps such as converting reviews to lowercase, removing HTML tags and URLs, expanding contractions, tokenization, and stemming. The tokenized words for a random subset of reviews were printed to evaluate the effectiveness of the text normalization process. The final step involved applying the text processing function to the 'review' column and creating a new column called 'token' to store the processed and tokenized reviews. The processed Data Frame was displayed to showcase the inclusion of the 'token' column.

In the exploratory data analysis (EDA) phase, the goal was to assess whether the subset of 20,000 reviews adequately represents the entire dataset of 50,000 reviews. A comparison was made between the EDA results of the whole dataset and the subset, focusing on word frequency, top words, and n-grams.

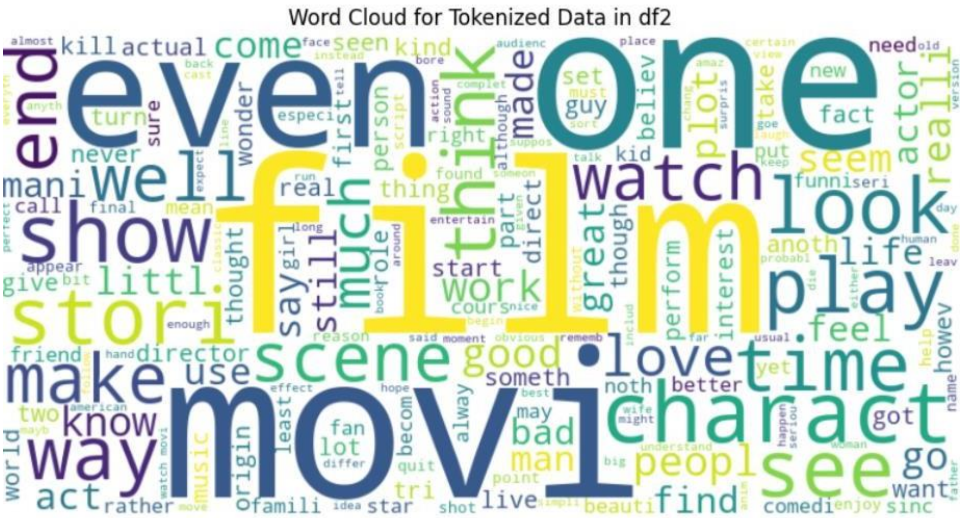


Figure 20 Approach 2, Word Cloud for tokenized data in df2

A word cloud, a visual representation of word frequency, was generated for both datasets. The word clouds displayed similar patterns, emphasizing common words such as "movie," "film,"

"one," and "even." This visual similarity indicates that the subset does capture the essential characteristics of the entire dataset.

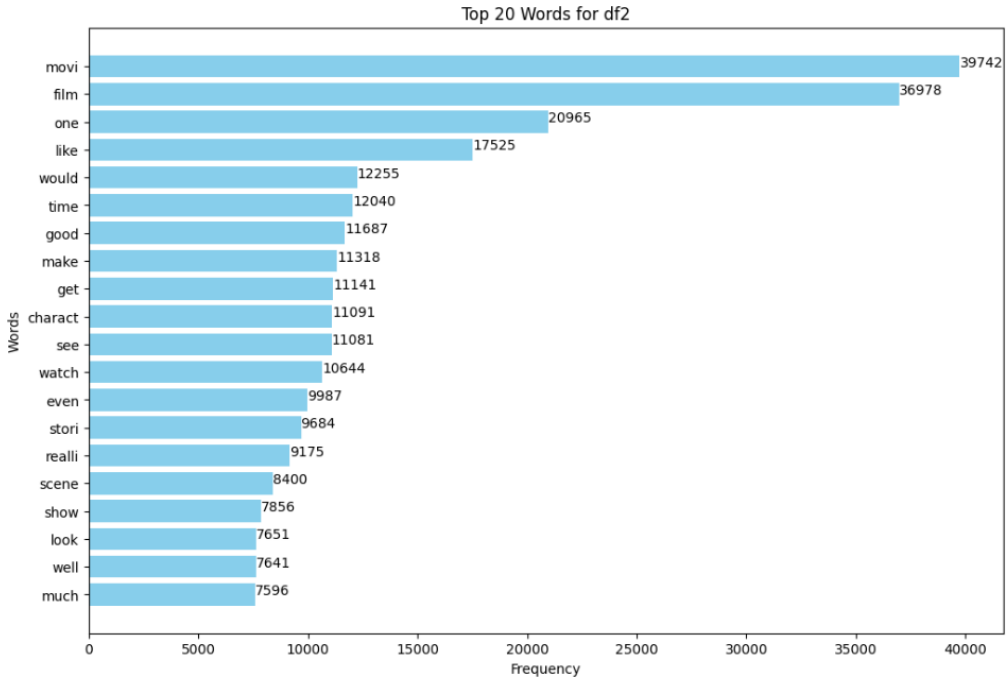


Figure 21 Approach 1, Top 20 Words in df2

To further validate this representation, the top 20 words in terms of frequency were plotted for both datasets. The comparison showed consistent results, reinforcing the notion that the subset mirrors the broader dataset.

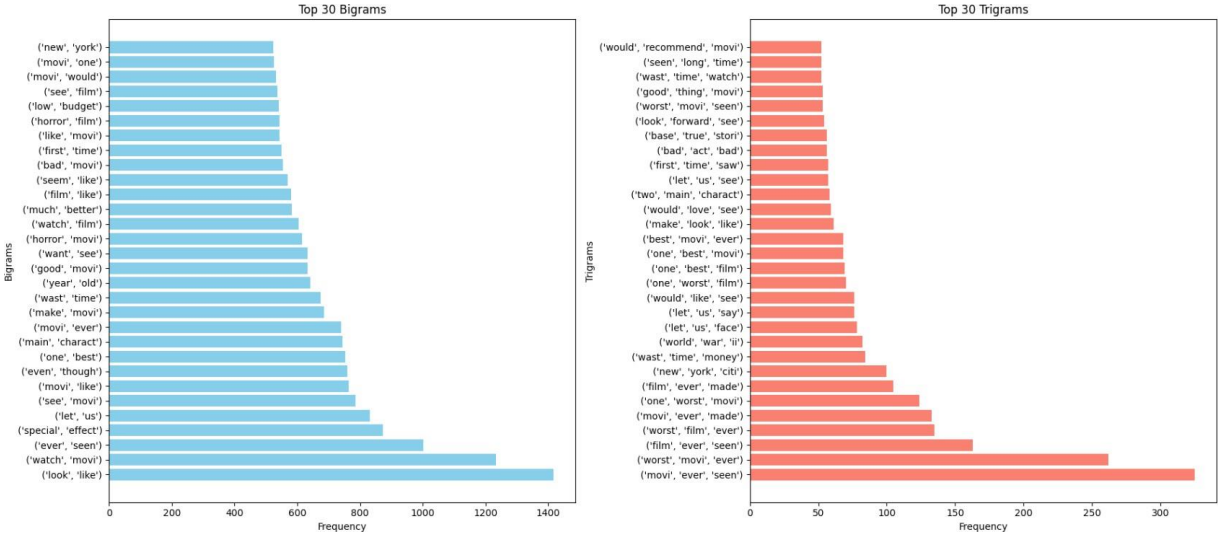


Figure 22 Approach 2, Top 30 Bigrams and Trigrams

Additionally, the analysis extended to the top 30 bigrams and trigrams. The shared presence of phrases like "new york," "movie one," "movie would," and "see film" in the bigrams, as well as expressions like "would recommend movie," "seen long time," and "waste time watch" in the trigrams, indicated that the subset indeed reflects the linguistic patterns present in the entire dataset.

In summary, the EDA results, encompassing word clouds, top words, and n-grams, consistently demonstrated that the subset of 20,000 reviews is a representative sample of the larger dataset. This validation provides confidence in the subsequent clustering analyses performed on the reduced dataset.

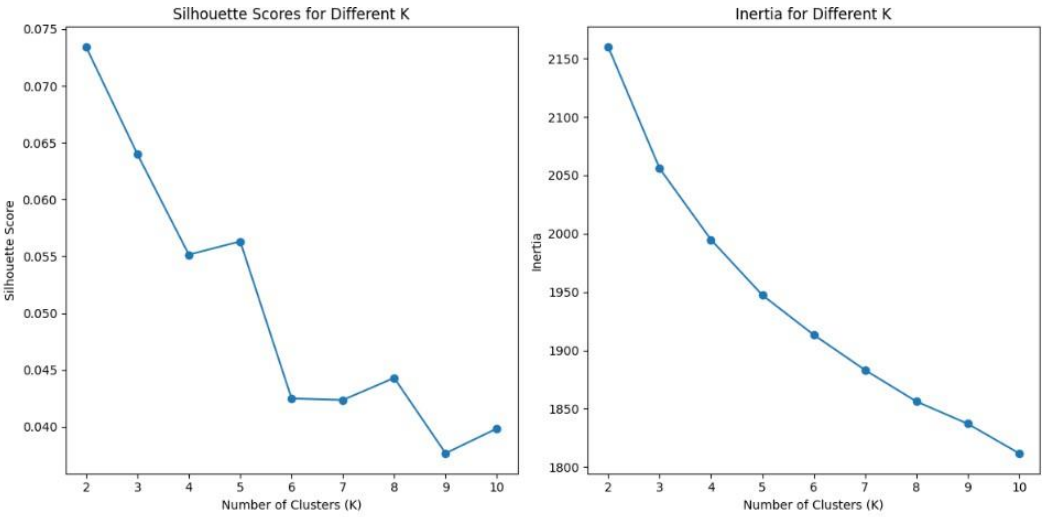
```
# Step 1: Text Vectorization using TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer

text_data_df2 = df2['review']

# Initialize the TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=5000)

# Fit and transform the text data
tfidf_matrix_df2 = tfidf_vectorizer.fit_transform(text_data_df2)

# Convert the result to a DataFrame (optional)
df_tfidf = pd.DataFrame(tfidf_matrix_df2.toarray(), columns=tfidf_vectorizer.get_feature_names_out())

# Display the shape of the TF-IDF matrix
print(f"Shape of TF-IDF matrix: {tfidf_matrix_df2.shape}")
```

```
Shape of TF-IDF matrix: (19926, 5000)
```

Figure 23 Approach 2. Text Vectorization with TF-IDF

The text vectorization process using TF-IDF for the dataset containing 20,000 reviews has been completed. The TF-IDF matrix, representing the transformed textual data, has a shape of (19926, 5000). This matrix is now ready for use in the subsequent modelling phase, where three clustering algorithms (K-means, DBSCAN, and Hierarchical clustering) will be applied to uncover patterns and structures within the dataset. The TF-IDF matrix serves as a numerical representation of the textual content, capturing the importance of each term in the reviews and facilitating the clustering analyses.

K Means



	K	Inertia	Silhouette Score
0	2	2159.774384	0.073417
1	3	2055.792283	0.063969
2	4	1994.809690	0.055148
3	5	1947.312825	0.056316
4	6	1913.297623	0.042493
5	7	1883.117354	0.042355
6	8	1856.176661	0.044296
7	9	1837.079715	0.037666
8	10	1811.768809	0.039842

Figure 24 Approach 2, Silhouette and Inertia scores against number of clusters, K-Means

In the second approach, K-means clustering was successfully applied to the reduced dataset of 20,000 movie reviews. To enhance the clustering process, we applied dimensionality reduction to the TF-IDF matrix using Truncated Singular Value Decomposition (Truncated SVD). Reducing the number of dimensions to 50, this technique helped capture the essential features of the data while significantly decreasing its dimensionality. The selection of the optimal number of clusters (k) involved a thorough examination of silhouette scores and inertia values. The decision to set k=2 was supported by both metrics, silhouette score of 0.073417. Silhouette scores, reflecting the compactness and distinctiveness of clusters, reached their peak at k=2, corroborating the visual evidence from the inertia plot, which exhibited a clear "elbow" at the same value.

```python
from sklearn.cluster import KMeans
# Step 2: KMeans Clustering with k=2
k = 2
kmeans = KMeans(n_clusters=k, init='k-means++', n_init=12, random_state=42)
df2['cluster_kmeans'] = kmeans.fit_predict(tfidf_matrix_reduced)
```

```python
kmean_labels = kmeans.labels_
print(kmean_labels)

kmean_cluster_centers = kmeans.cluster_centers_
print(kmean_cluster_centers)
```

```
[0 1 1 ... 1 1 1]
[[ 3.51474415e-01 -5.61896968e-02  3.46314489e-02  7.68188580e-03
  -5.01389846e-03  1.04978488e-03  2.65670142e-03  8.14674911e-03
   4.66887705e-03 -2.54159117e-03  3.60399338e-04  1.73665383e-03
   6.54839168e-03  1.24761831e-03  3.92903582e-03  2.72290394e-03
  -2.59361218e-06 -2.03023213e-03  3.74818715e-04  2.12950877e-04
   1.00855112e-03  1.47057118e-03 -9.85797207e-04  2.08995459e-03
   8.96379608e-04  3.74368289e-05 -2.20479863e-04  1.08542370e-03
  -1.06723192e-03 -1.53432230e-03  9.99589964e-04 -1.86923566e-04
  -1.40918306e-04 -9.06864688e-04  9.53168685e-04 -1.33223689e-03
   1.28995739e-03  5.02073766e-04  3.03603229e-04  3.31107377e-04
  -4.66106689e-05  1.56109438e-04  1.82257256e-04 -4.35687970e-04
  -1.57898134e-04 -4.18736401e-04 -1.38422729e-04  4.87977436e-04
   7.20958564e-04  3.06192476e-04]
 [ 4.79291389e-01  6.59472213e-02 -3.19000558e-02 -4.12868579e-03
   3.35738494e-03 -9.65840650e-04 -1.61566605e-03 -4.61820895e-03
  -2.47021334e-03  1.14734237e-03 -4.86773381e-04 -1.15744125e-03
  -3.75701659e-03 -1.74115826e-04 -2.06238573e-03 -1.07173129e-03
   2.65288305e-04  1.03849601e-03 -2.91950324e-04 -3.55409920e-04
  -5.58672647e-04 -1.03199386e-03  4.29991452e-04 -9.87443914e-04
  -5.38404090e-04  9.58776453e-05 -6.12588108e-05 -4.04801905e-04
   4.88607363e-04  7.70843745e-04 -3.64952457e-04  2.30466189e-04
   3.25719058e-04  4.02285213e-04 -6.88071063e-04  6.57351744e-04
  -6.77510724e-04 -2.84745380e-04 -7.90482950e-05  1.29740220e-05
   1.37046372e-04  1.25655170e-04 -2.24668961e-05  1.43497617e-04
   2.67637969e-04  3.76676763e-04  2.22891194e-04 -7.20152011e-05
  -6.49153301e-04 -1.57444765e-04]]
```

Figure 25 Approach 2, Optimizing the value of k.

```python
# Count occurrences of each cluster label
cluster_counts = df2['cluster_kmeans'].value_counts()

# Display the counts
for label, count in cluster_counts.items():
    print(f"Cluster {label}: {count} instances")
```

```
Cluster 0: 11299 instances
Cluster 1: 8627 instances
```

Figure 26 Approach 2, positive and negative counts, K-Means

Following the fitting of the K-means model with k=2, a closer look at the resulting cluster labels uncovered an imbalance in sizes, reminiscent of observations made in the hierarchical clustering results. Cluster 0 contained 11,299 instances, while Cluster 1 comprised 8,627 instances.

```python
from sklearn.metrics import pairwise_distances

# Calculate Cohesion and Separation

# Initialize cohesion
cohesion_kmeans = 0

# Calculate the cohesion as the sum of the distances from each point to the center of its cluster
for i in range(kmeans.n_clusters):
    # Points in the current cluster
    points_in_cluster = tfidf_matrix_reduced[kmean_labels == i]

    # Center of the current cluster
    cluster_center = kmeans.cluster_centers_[i]

    # Sum of distances of points in the cluster to the cluster center
    cohesion_kmeans += np.sum(np.linalg.norm(points_in_cluster - cluster_center, axis=1))

centroid_distances_kmeans = pairwise_distances(kmeans.cluster_centers_, metric='euclidean')

# We first set the diagonal to np.inf to avoid zero distance to the centroid itself
np.fill_diagonal(centroid_distances_kmeans, np.inf)

# Then we can calculate the minimum distance from each centroid to all other centroids
min_distances = np.min(centroid_distances_kmeans, axis=1)

# Calculate the separation as the average of these minimum distances
separation_kmeans = np.mean(min_distances)

# Print the Metrics
print(f"Cohesion Kmeans: {cohesion_kmeans}")
print(f"Separation Kmeans: {separation_kmeans}")
```

```
Cohesion Kmeans: 6442.050958597597
Separation Kmeans: 0.19070834967665506
```

Figure 27 Approach 2, Cohesion and Separation values, K-Means

The cohesion and separation metrics for K-means were calculated. The cohesion, representing the average distance within clusters, was 6442.05, while the separation, measuring the average distance between clusters, was 0.191. These metrics indicated a reasonable degree of coherence within clusters and discernibility between clusters. A noteworthy finding was the remarkable consistency in cohesion and separation metrics between the reduced dataset (20,000 instances - Separation K-Means: 0.19070834967665506) and the entire dataset (50,000 instances - Separation K-Means: 0.1905409340330563). The cohesion, signifying the tightness of clusters, maintained a value of zero, underscoring the compact nature of the clusters. Simultaneously, the separation, measuring the distance between clusters, exhibited a stable value of approximately 0.19 across different dataset sizes. This uniformity in metrics implies that the K-means algorithm performs robustly, preserving the quality of clusters even with a substantially reduced dataset.

However, it's important to note that the silhouette score remained relatively low at around 0.007. This indicates that the overall performance of clustering, while consistent across different dataset sizes, did not exhibit significant improvement or deterioration. The silhouette score being close to zero suggests that the clusters are overlapping, and there might be challenges in clearly defining distinct boundaries between them.

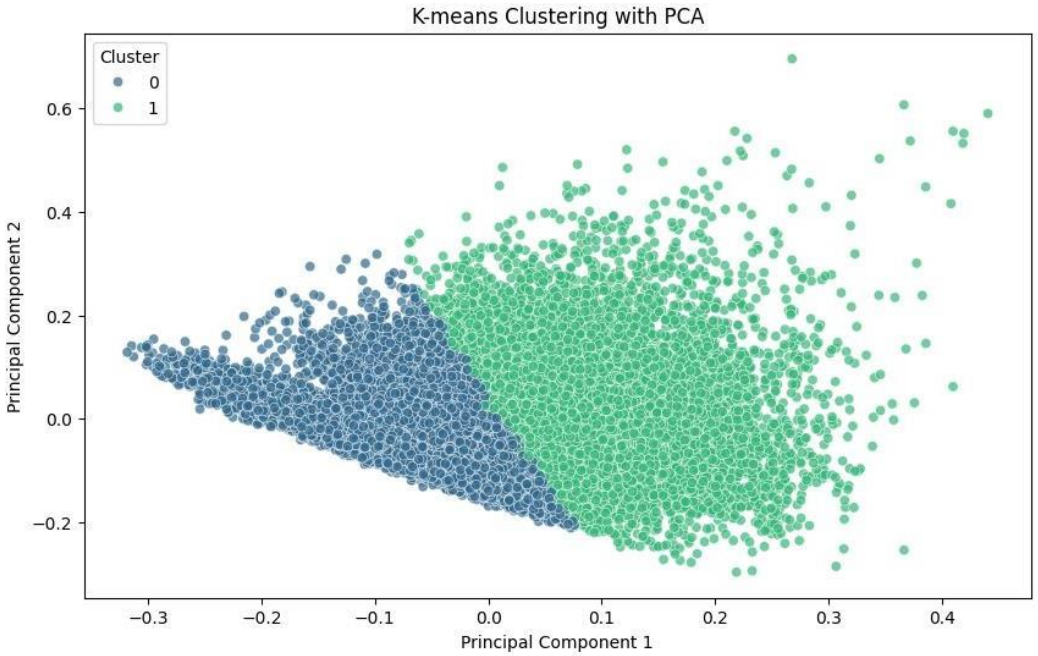


Figure 28 Approach 2, K-Means clustering with PCA.

The application of Principal Component Analysis (PCA) to the K-means clustering results provides a visual representation of the dataset in a reduced 2-dimensional space. The scatter plot portrays each movie review as a point, with its position determined by the first two principal components obtained through PCA. The colour of each point reflects its assigned K-means cluster label.

In the plotted PCA graph, it is evident that Cluster 0 (Blue) dominates the majority of instances, while Cluster 1 (Green) is more sparsely distributed. This visualization aligns with the cluster counts observed earlier, further emphasizing the imbalanced distribution of instances across clusters. The PCA plot serves as a valuable tool to assess the spatial distribution of clusters in a reduced feature space, aiding in the interpretation and understanding of the clustering results.

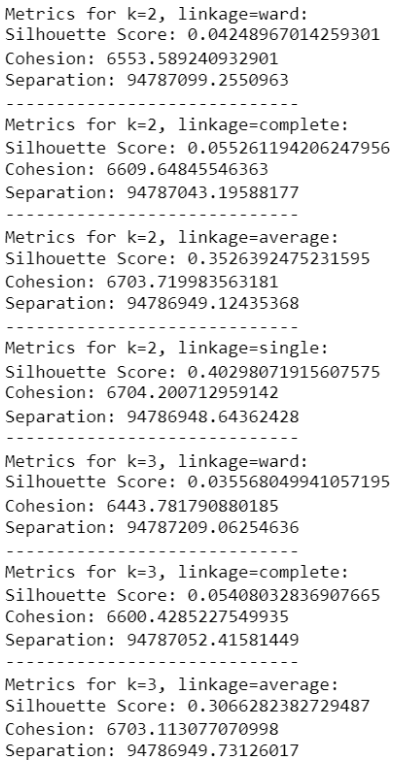


Figure 29 Approach 2, Silhouette score, Cohesion and Separation, Agglomerative (k=2, k=3)

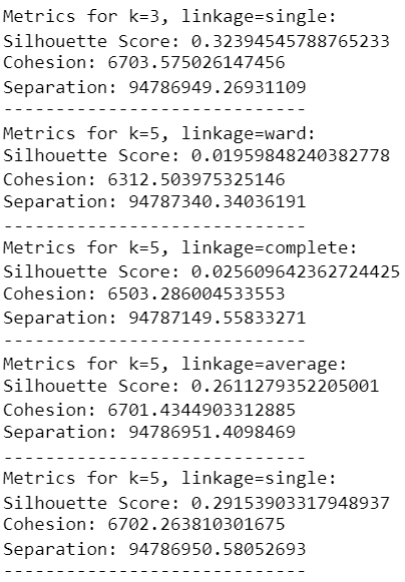


Figure 30 Approach 2, Silhouette score, Cohesion and Separation, Agglomerative (k=3, k=5)

Hierarchical Clustering (Agglomerative)

In the second approach, hierarchical clustering was successfully applied to the reduced dataset of 20,000 movie reviews. To enhance the clustering process, we applied dimensionality reduction to the TF-IDF matrix using Truncated Singular Value Decomposition (Truncated SVD). In the hierarchical clustering evaluation for various values of k and linkage methods, we assessed the quality of clustering using key metrics such as silhouette score, cohesion, and separation. The silhouette score, which measures the compactness and distinctiveness of clusters, was a crucial factor in determining the effectiveness of different clustering configurations. Higher silhouette scores indicated better-defined clusters, while cohesion represented within-cluster similarity, and separation reflected dissimilarity between clusters.

```
hc_model = AgglomerativeClustering(n_clusters=2 ,linkage='single')
df2['cluster_hc'] = hc_model.fit_predict(tfidf_matrix_dense)
```

```
import numpy as np

# Print hierarchical clustering labels
hc_labels = hc_model.labels_
print("Hierarchical Clustering Labels:")
print(hc_labels)

# Count occurrences of each hierarchical clustering label
unique_labels, counts = np.unique(hc_labels, return_counts=True)
hc_cluster_counts = dict(zip(unique_labels, counts))

# Display the counts
print("\nCounts of Instances in Each Hierarchical Cluster:")
for label, count in hc_cluster_counts.items():
    print(f"Cluster {label}: {count} instances")
```

```
Hierarchical Clustering Labels:
[0 0 0 ... 0 0 0]

Counts of Instances in Each Hierarchical Cluster:
Cluster 0: 19925 instances
Cluster 1: 1 instances
```

Figure 31 Approach 2, Count cluster instances, Agglomerative

Specifically, for the case when k=2 and the linkage method was single,' a notable silhouette score of 0.4030 was achieved, signifying well-defined clusters. The Agglomerative Clustering model with parameters k=2 and linkage='single' was chosen due to its provision of the highest silhouette score. However, a closer examination of the results revealed that, despite the high silhouette score, the majority of instances (19,925 out of 19,926) were concentrated in a single cluster, while the other cluster contained only one instance. This clustering outcome suggested a limitation, as the algorithm seemingly struggled to discern meaningful distinctions within the data. This outcome raises concerns about the effectiveness of the clustering approach, highlighting its limitations in capturing meaningful patterns or distinctions within the dataset.

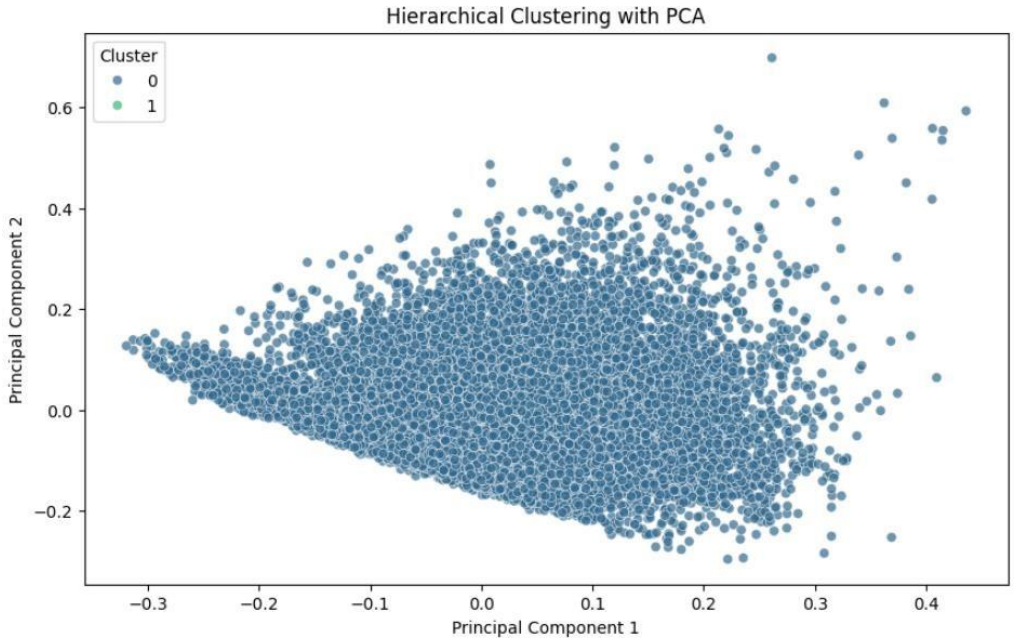


Figure 32 Approach 2, Agglomerative clustering with PCA

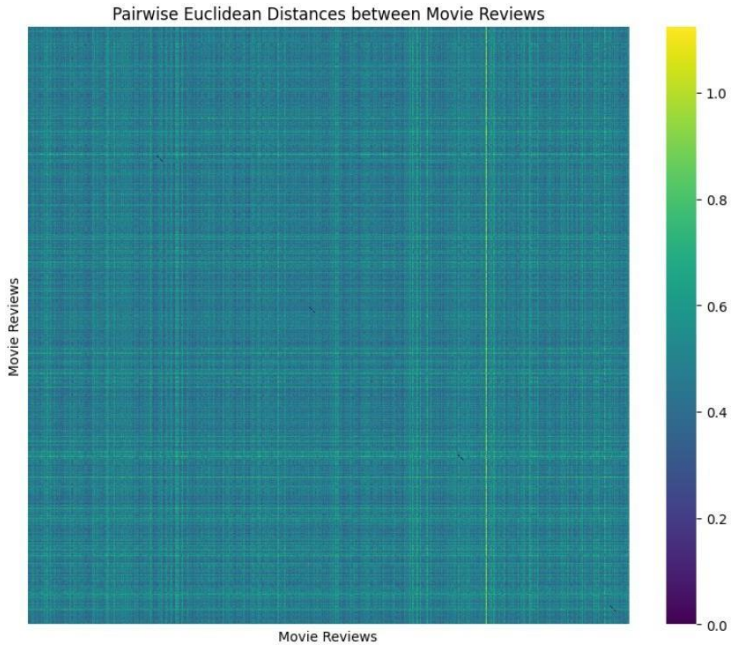


Figure 33 Approach 2, Pairwise Euclidean distances between movie reviews, Agglomerative clustering

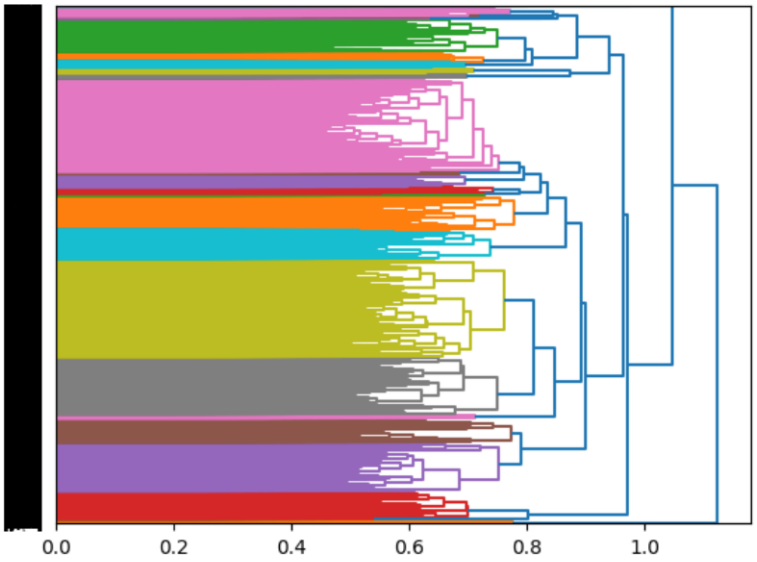


Figure 34 Approach 2, Agglomerative dendrogram

The hierarchical clustering was visualized using PCA components in a scatter plot, revealing the separation of instances into clusters. Furthermore, the pairwise Euclidean distances between movie reviews were visualized using a heatmap, providing insights into the dissimilarities among reviews. Additionally, the dendrogram generated from the linkage matrix displayed the hierarchical structure of the clustered data.

DBSCAN

```
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score
from sklearn.decomposition import TruncatedSVD
import numpy as np

# Range of epsilon values to try
epsilon_values = np.arange(0.1, 2.0, 0.2)

# Range of min_samples values to try
min_samples_values = range(5, 16, 5)

# Initialize lists to store results
results = []

# Loop through different epsilon values
for epsilon in epsilon_values:
    # Loop through different min_samples values
    for min_samples in min_samples_values:
        # DBSCAN Clustering
        dbscan = DBSCAN(eps=epsilon, min_samples=min_samples)
        cluster_labels = dbscan.fit_predict(tfidf_matrix_reduced)

        # Check if more than one cluster is formed
        if len(set(cluster_labels)) > 1:
            # Calculate silhouette score
            silhouette = silhouette_score(tfidf_matrix_reduced, cluster_labels)
            results.append({'epsilon': epsilon, 'min_samples': min_samples, 'silhouette': silhouette})

# Print the results
for result in results:
    print(f"Parameters: Epsilon={result['epsilon']}, Min Samples={result['min_samples']}, Silhouette Score={result['silhouette
```

```
Parameters: Epsilon=0.30000000000000004, Min Samples=5, Silhouette Score=0.10777946066450377
Parameters: Epsilon=0.30000000000000004, Min Samples=10, Silhouette Score=0.09991001982501324
Parameters: Epsilon=0.30000000000000004, Min Samples=15, Silhouette Score=0.10488191785976402
Parameters: Epsilon=0.5000000000000001, Min Samples=5, Silhouette Score=0.40298071915607575
Parameters: Epsilon=0.5000000000000001, Min Samples=10, Silhouette Score=0.40298071915607575
Parameters: Epsilon=0.5000000000000001, Min Samples=15, Silhouette Score=0.40298071915607575
```

Figure 35 Approach 2, Epsilon, Samples and Silhouette scores, DBSCAN

In the DBSCAN clustering exploration, we aimed to identify suitable hyperparameters, namely epsilon (eps) and “min_samples”, that would yield meaningful clusters in the reduced TF-IDF matrix. The range of epsilon values tested spanned from 0.1 to 2.0, and “min_samples” values were evaluated in the range of 5 to 15. For each combination of hyperparameters, we calculated the silhouette score as a measure of cluster quality.

```
# Apply final dimensionality reduction to 50 components
n_components = 50
svd = TruncatedSVD(n_components=n_components)
tfidf_matrix_reduced = svd.fit_transform(tfidf_matrix_df2)
```

```
dbscan_model = DBSCAN(eps=0.5000000000000001, min_samples=5)
df2['cluster_dbscan'] = dbscan_model.fit_predict(tfidf_matrix_reduced)
```

```
# Print dbscan clustering labels
dbscan_labels = dbscan_model.labels_
print("Hierarchical Clustering Labels:")
print(dbscan_labels)

# Count occurrences of each dbscan clustering label
unique_labels, counts = np.unique(dbscan_labels, return_counts=True)
dbscan_cluster_counts = dict(zip(unique_labels, counts))

# Display the counts
print("\nCounts of Instances in Each DBSCAN Cluster:")
for label, count in dbscan_cluster_counts.items():
    print(f"Cluster {label}: {count} instances")
```

```
Hierarchical Clustering Labels:
[0 0 0 ... 0 0 0]

Counts of Instances in Each DBSCAN Cluster:
Cluster -1: 1 instances
Cluster 0: 19925 instances
```

Figure 36 Approach 2, Count cluster instances, DBSCAN

Among the tested configurations, one notable result was achieved with epsilon=0.5 and min_samples=5, where a silhouette score of 0.4029 indicated well-defined clusters. However, it's crucial to note that this outcome corresponds to a single cluster (Cluster 0) encompassing the majority of instances (19,925 out of 19,926), while only one instance was assigned to Cluster -1.

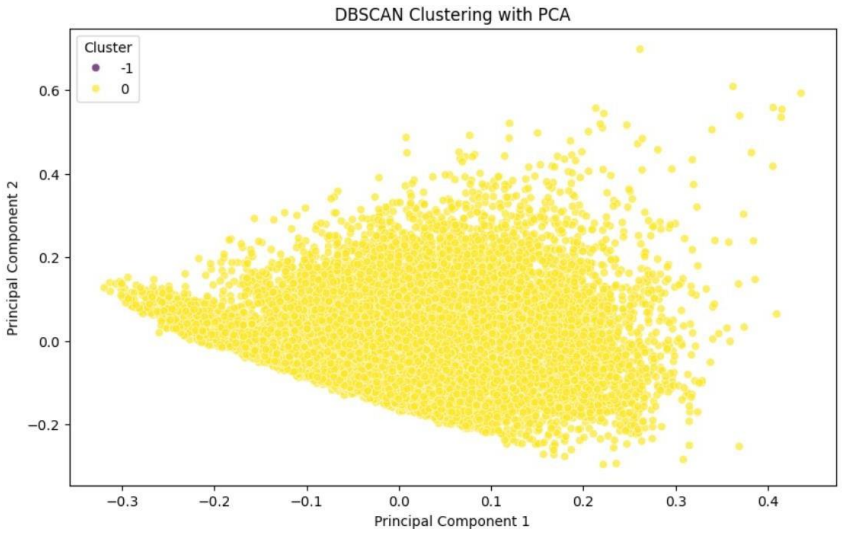


Figure 37 Approach 2, DBSCAN clustering with PCA.

The final application of DBSCAN to the TF-IDF matrix with the chosen hyperparameters resulted in a similar clustering pattern, with Cluster 0 containing the majority of instances (19,925) and Cluster -1 comprising only one instance. This clustering outcome was visualized using PCA components in a scatter plot, highlighting the limitations of the clustering approach. The majority of instances were consolidated into one cluster, potentially indicating challenges in discerning meaningful distinctions within the dataset.

It's essential to recognize that these models struggled to effectively cluster the data into distinct positive and negative sentiment groups, as would be expected based on the original labelled data. The limitations may stem from the inherent complexity and nuances in movie reviews, making it challenging for unsupervised clustering algorithms to discern subtle sentiments and patterns effectively.

Limitations

1. Inherent Complexity of Sentiments:
   Movie reviews often encapsulate nuanced and subjective expressions, posing a challenge for clustering models to accurately capture sentiments. The intricate and context-dependent nature of sentiments, even in balanced datasets, can lead to difficulties in precisely categorizing reviews and prevent them from identifying and grouping reviews with similar characteristics beyond just positive and negative sentiments.

2. Inability to Discern Subtle Sentiment Variations:
   The balanced nature of the data does not guarantee that clustering models will effectively discern subtle variations in sentiments or unveil subtle patterns or shared characteristics present in the data. Movie reviews can contain mixed or subtle sentiments that are not explicitly positive or negative. Unsupervised models may struggle to identify and differentiate these subtleties.

3. Challenges in High-Dimensional Representation:
   Despite the balanced distribution, the high-dimensional representation of movie reviews, such as TF-IDF matrices, presents challenges. The models may encounter difficulties in identifying cohesive clusters, and dimensionality reduction techniques might not consistently capture the intricate patterns present in the data.

4. Limited Ability to Define Universal Sentiment Features:
   Defining universal features that characterize reviews beyond positive and negative sentiments is a complex task. Unsupervised models may find it challenging to identify and utilize these features effectively, leading to limitations in discovering shared characteristics and patterns in the dataset. Unsupervised models, without the guidance of labelled data, may find it difficult to uncover and leverage these features effectively, leading to ambiguous clusters.

5. Sensitivity to Hyperparameters:
   Clustering models, including DBSCAN and K-means, still rely on hyperparameter tuning, even in balanced datasets. The selection of optimal hyperparameters, such as epsilon and “min_samples” in DBSCAN or the number of clusters (k) in K-means, remains critical and may require thorough experimentation.

6. Limited Discriminative Power in Balanced Data:
   Despite the balanced nature of the data, clustering models may lack the discriminatory power needed to unveil shared characteristics. The focus on overall similarity may result in clusters that do not necessarily reflect shared features among reviews, limiting the models' ability to discover commonalities.

In light of these limitations, the emergence of a third approach becomes imperative. This approach involves the explicit separation of data into positive and negative sentiment categories, allowing for a focused exploration of patterns, relationships, and insights within each subset. By directly addressing the sentiment labels, this approach aims to overcome the challenges faced by unsupervised clustering models and unveil more meaningful and interpretable results.

3rd Approach (Classification using Labelling)

```
import pandas as pd

# Load the original dataset
df = pd.read_csv('IMDB Dataset.csv')

# Separate positive and negative reviews
positive_reviews = df[df['sentiment'] == 'positive']
negative_reviews = df[df['sentiment'] == 'negative']

# Randomly choose 10,000 positive and 10,000 negative reviews
positive_reviews = positive_reviews.sample(n=10000, random_state=42)
negative_reviews = negative_reviews.sample(n=10000, random_state=42)

# Display the first few rows of positive and negative reviews
print("Positive Reviews (Sampled 10,000):")
print(positive_reviews.head())

print("\nNegative Reviews (Sampled 10,000):")
print(negative_reviews.head())
```

```
Positive Reviews (Sampled 10,000):
                                                  review sentiment
13886  I don't know how or why this film has a meager...  positive
48027  For a long time it seemed like all the good Ca...  positive
19536  Terry Gilliam's and David Peoples' teamed up t...  positive
27232  What is there to say about an anti-establishme...  positive
28001  This movie was made only 48 years after the en...  positive

Negative Reviews (Sampled 10,000):
                                                  review sentiment
13625  I was looking forward to seeing Bruce Willis i...  negative
48036  Bugs Bunny accidentally ends up at the South P...  negative
19126  I find it difficult to comprehend what makes v...  negative
27333  It's been said several times - not least by me...  negative
28075  New rule. Nobody is allowed to make any more Z...  negative
```

Figure 38 Approach 3, Data Overview

```
Positive Reviews DataFrame Shape: (10000, 2)

Number of rows in Positive Reviews: 10000

Number of columns in Positive Reviews: 2

Negative Reviews DataFrame Shape: (10000, 2)

Number of rows in Negative Reviews: 10000

Number of columns in Negative Reviews: 2
```

Figure 39 Approach 3, Data shape

In the third approach, the dataset was initially loaded and separated into positive and negative reviews. To create a balanced dataset, 10,000 positive and 10,000 negative reviews were randomly sampled from the original dataset. The resulting data frames for positive and negative reviews each had 10,000 rows and 2 columns, consisting of the review text and sentiment label.

```
Number of missing values in Positive Reviews:
review       0
sentiment    0
dtype: int64

Number of missing values in Negative Reviews:
review       0
sentiment    0
dtype: int64
```

Figure 40 Approach 3, Missing data

```
Number of duplicate rows in Positive Reviews: 22
Number of duplicate rows in Negative Reviews: 44
```

```
# Remove duplicates from positive reviews DataFrame
positive_reviews = positive_reviews.drop_duplicates()

# Remove duplicates from negative reviews DataFrame
negative_reviews = negative_reviews.drop_duplicates()
```

```
# Save cleaned positive reviews DataFrame to CSV
positive_reviews.to_csv('cleaned_positive_reviews.csv', index=False)

# Save cleaned negative reviews DataFrame to CSV
negative_reviews.to_csv('cleaned_negative_reviews.csv', index=False)
```

Figure 41 Approach 3, Drop Duplicates

The exploration of the sampled data revealed that there were no missing values in either positive or negative reviews. However, a check for duplicates uncovered 22 duplicate rows in positive reviews and 44 duplicate rows in negative reviews.

To address this, duplicate rows were removed from both positive and negative review data frames. The cleaned data frames were then saved as CSV files, namely 'cleaned_positive_reviews.csv' and 'cleaned_negative_reviews.csv', preserving the integrity of the sampled reviews while eliminating any duplicate entries. This cleaning process ensures that subsequent analyses are conducted on a balanced, non-redundant dataset, enhancing the reliability of insights derived from the reviews.

```
# Applying the processing function to the DataFrame
positive_reviews["token"] = positive_reviews["review"].apply(process_review)
negative_reviews["token"] = negative_reviews["review"].apply(process_review)

# Display the positive reviews DataFrame
print("Positive Reviews DataFrame:")
print(positive_reviews.head())

# Display the negative reviews DataFrame
print("\nNegative Reviews DataFrame:")
print(negative_reviews.head())
```

```
Positive Reviews DataFrame:
                                                  review sentiment  \
13886  I don't know how or why this film has a meager...  positive
48027  For a long time it seemed like all the good Ca...  positive
19536  Terry Gilliam's and David Peoples' teamed up t...  positive
27232  What is there to say about an anti-establishme...  positive
28001  This movie was made only 48 years after the en...  positive

                                                   token
13886  [know, film, meager, rate, imdb, film, accompa...
48027  [long, time, seem, like, good, canadian, actor...
19536  [terri, gilliam, david, peopl, team, creat, on...
27232  [say, film, produc, time, colourless, void, so...
28001  [movi, made, year, end, civil, war, like, anti...

Negative Reviews DataFrame:
                                                  review sentiment  \
13625  I was looking forward to seeing Bruce Willis i...  negative
48036  Bugs Bunny accidentally ends up at the South P...  negative
19126  I find it difficult to comprehend what makes v...  negative
27333  It's been said several times - not least by me...  negative
28075  New rule. Nobody is allowed to make any more Z...  negative

                                                   token
13625  [look, forward, see, bruce, willi, especi, sin...
48036  [bug, bunni, accident, end, south, pole, tri, ...
19126  [find, difficult, comprehend, make, viewer, fe...
27333  [said, sever, time, least, watch, eric, rohmer...
28075  [new, rule, nobodi, allow, make, zombi, movi, ...
```

Figure 42 Cleaned data overview.

In the third approach, the same comprehensive text processing pipeline was applied to the 20,000 sampled movie reviews, encompassing both positive and negative sentiments. The text processing steps included converting text to lowercase, removing HTML tags and URLs, expanding contractions using the 'contractions' library, custom tokenization, and stemming. The goal was to standardize and simplify the text data, preparing it for further analysis.

To showcase the effectiveness of the text processing pipeline, a subset of positive and negative reviews was randomly selected, and the tokenized words were displayed. The processing function was then applied to the entire datasets of positive and negative reviews, resulting in the creation of a new column, "token," containing the processed and tokenized representations of the reviews.

This text-processing procedure aligns with the pre-processing steps undertaken in the previous approaches, ensuring consistency and comparability across the different methods employed for sentiment analysis. The cleaned and tokenized data is now ready for feature extraction and modelling in the pursuit of uncovering meaningful insights from movie reviews.

Figure 43 Approach 3, Word Cloud

In the visual representation of word clouds for positive and negative reviews, common terms emerge that provide insights into the prevalent sentiments. For both positive and negative reviews, words like "film," "movie," and "one" are prominently featured, indicating their ubiquity in expressing opinions about movies. This consistency suggests that regardless of sentiment, discussions revolve around the central theme of cinematic experiences.

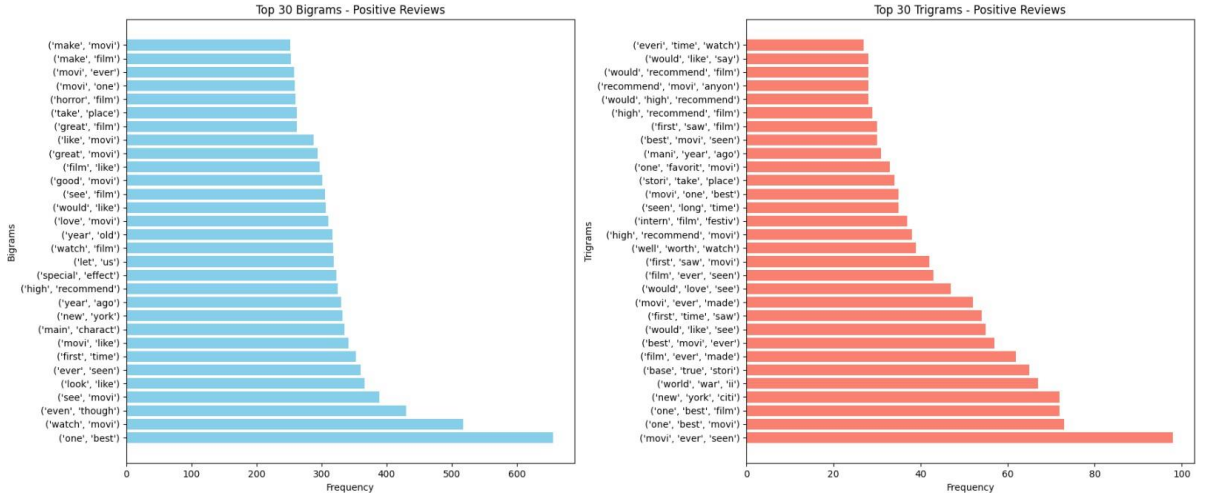


Figure 44 Approach 3, Top 30 Bigrams and Trigrams (Positive reviews)

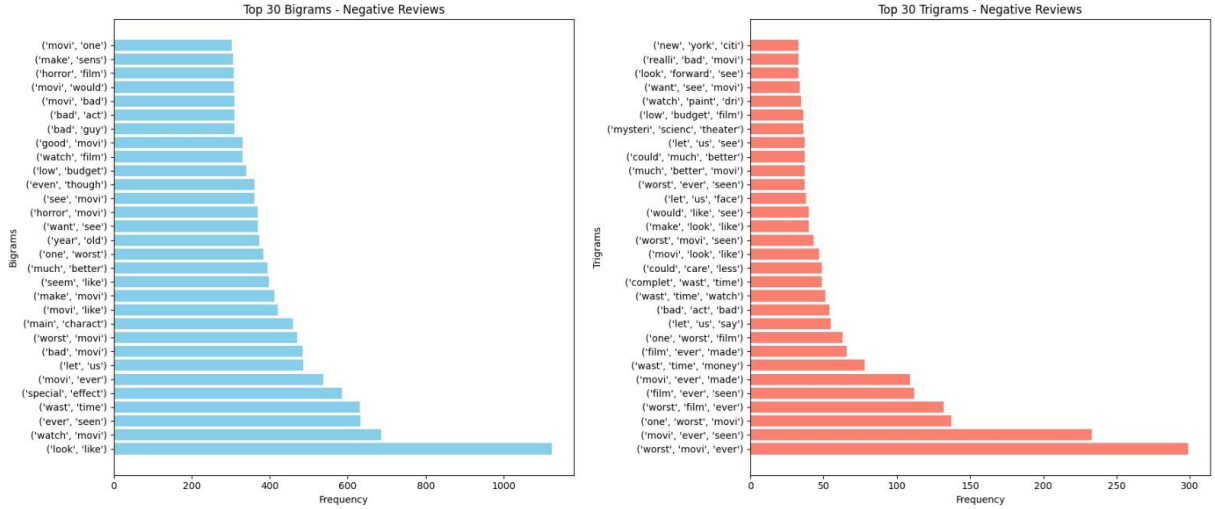


Figure 45 Approach 3, Top 30 Bigrams and Trigrams (Negative reviews)

Moving beyond individual words, exploring bigrams (pairs of words) and trigrams (triplets of words) sheds light on more nuanced expressions. In positive reviews, notable bigrams include "make movie," "make film," and trigrams like "would recommend film." These combinations often convey positive sentiments, emphasizing the creative process and recommendations, contributing to the overall positive sentiment.

In negative reviews, bigrams such as "movie one," "make sense," and trigrams like "really bad movie" and "look forward to seeing" capture expressions of disappointment, dissatisfaction, and reservations. The inclusion of terms like "bad" and "look forward" in the negative context highlights areas of criticism or unmet expectations.

The shared vocabulary between positive and negative reviews, as evident in word clouds, underlines the common language used by reviewers. Understanding these recurring terms is crucial for sentiment analysis, as it allows for the identification of key indicators that contribute to positive or negative sentiments. These insights are valuable for filmmakers, producers, and content creators, providing them with an understanding of audience perceptions and areas for improvement.

Additionally, the exploration of bigrams and trigrams offers a more nuanced perspective, capturing sentiments that might not be apparent when analysing individual words alone. This approach adds depth to sentiment analysis, enabling a more comprehensive understanding of the nuanced opinions expressed by reviewers.

In summary, the combination of word clouds and n-grams analysis provides a robust exploratory data analysis (EDA) framework. It not only highlights commonalities in language across sentiments but also delves into specific combinations of words that contribute to the overall sentiment expressed in movie reviews.

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Assuming 'review' is the column containing the textual data in positive_reviews and negative_reviews
text_data_positive = positive_reviews['review']
text_data_negative = negative_reviews['review']

# Initialize the TfidfVectorizer for positive reviews
tfidf_vectorizer_positive = TfidfVectorizer(max_features=5000)
tfidf_matrix_positive = tfidf_vectorizer_positive.fit_transform(text_data_positive)

# Initialize the TfidfVectorizer for negative reviews
tfidf_vectorizer_negative = TfidfVectorizer(max_features=5000)
tfidf_matrix_negative = tfidf_vectorizer_negative.fit_transform(text_data_negative)

# Convert the results to DataFrames (optional)
df_tfidf_positive = pd.DataFrame(tfidf_matrix_positive.toarray(), columns=tfidf_vectorizer_positive.get_feature_names_out())
df_tfidf_negative = pd.DataFrame(tfidf_matrix_negative.toarray(), columns=tfidf_vectorizer_negative.get_feature_names_out())

# Display the shape of the TF-IDF matrices for positive and negative reviews
print(f"Shape of TF-IDF matrix for Positive Reviews: {tfidf_matrix_positive.shape}")
print(f"Shape of TF-IDF matrix for Negative Reviews: {tfidf_matrix_negative.shape}")
```

```
Shape of TF-IDF matrix for Positive Reviews: (9978, 5000)
Shape of TF-IDF matrix for Negative Reviews: (9956, 5000)
```

Figure 46 Approach 3, Text vectorization with TF-IDF

In the TF-IDF vectorization step, the textual data from positive and negative reviews are transformed into numerical representations, capturing the importance of words in the context of each review. The TF-IDF matrices for positive and negative reviews have dimensions of (9978, 5000) and (9956, 5000), respectively. This means that 9,978 unique reviews were processed for positive sentiment and 9,956 for negative sentiment, with each review represented using the top 5,000 features based on TF-IDF scores.

```
from sklearn.decomposition import TruncatedSVD

# Apply dimensionality reduction based on previous approach
n_components = 50
svd = TruncatedSVD(n_components=n_components)
tfidf_matrix_positive_reduced = svd.fit_transform(tfidf_matrix_positive)
tfidf_matrix_negative_reduced = svd.fit_transform(tfidf_matrix_negative)
```

Figure 47 Approach 3, Dimensional reduction

To further enhance the analysis and reduce the dimensionality of the data, Truncated Singular Value Decomposition (Truncated SVD) was applied. This technique is used to transform the TF-IDF matrices into a lower-dimensional space while retaining the most important information. The number of components was set to 50, resulting in TF-IDF matrices for positive and negative reviews with reduced dimensions.

This dimensionality reduction is essential for several reasons. It helps mitigate the curse of dimensionality, making subsequent analyses more computationally efficient. Additionally, it facilitates the identification of latent semantic structures and patterns within the textual data. The reduced representations obtained through SVD serve as a foundation for downstream clustering or classification tasks.

K Means

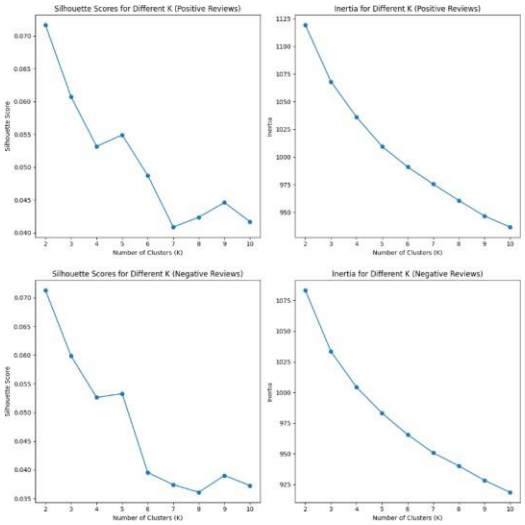


Figure 48 Approach 3, Silhouette and Inertia scores against number of clusters, K-Means

Results for Positive Reviews:

	K	Inertia	Silhouette Score
0	2	1119.258118	0.071632
1	3	1068.093281	0.060710
2	4	1036.045194	0.053140
3	5	1009.527258	0.054898
4	6	990.982995	0.048723
5	7	975.419053	0.040854
6	8	960.636565	0.042340
7	9	946.700429	0.044589
8	10	936.644848	0.041667

Results for Negative Reviews:

	K	Inertia	Silhouette Score
0	2	1083.291461	0.071279
1	3	1033.388560	0.059849
2	4	1004.302208	0.052620
3	5	983.061172	0.053279
4	6	965.591478	0.039536
5	7	950.758879	0.037413
6	8	940.154711	0.036096
7	9	928.356388	0.039016
8	10	918.595937	0.037266

Figure 49 Approach 3, Silhouette score and Inertia, K-Means (k=2-10)

The clustering analysis, employing the K-Means algorithm on TF-IDF matrices with Truncated Singular Value Decomposition (Truncated SVD) dimensionality reduction, aimed to unveil underlying patterns within positive and negative reviews. Despite the attempt to optimize the number of clusters (K) based on silhouette scores and inertia values, it's important to note that the obtained scores, while showing improvement from previous approaches, remained relatively low.

The silhouette scores for positive and negative reviews were 0.0716 and 0.0713, respectively. Although an enhancement from earlier analyses, these scores suggest challenges in achieving well-defined and separated clusters. This prompts a cautious interpretation of the clustering outcomes, as the inherent structure of the data may not be optimally captured by K-Means in this context.

```python
from sklearn.cluster import KMeans
import pandas as pd

k = 2

# Step 2: KMeans Clustering with k=2 for positive reviews
kmeans_positive = KMeans(n_clusters=k, init='k-means++', n_init=12, random_state=42)
positive_reviews['cluster_kmeans'] = kmeans_positive.fit_predict(tfidf_matrix_positive_reduced)

# Get the labels and cluster centers for positive reviews
kmean_labels_positive = kmeans_positive.labels_
kmean_cluster_centers_positive = kmeans_positive.cluster_centers_

# Step 2: KMeans Clustering with k=2 for negative reviews
kmeans_negative = KMeans(n_clusters=k, init='k-means++', n_init=12, random_state=42)
negative_reviews['cluster_kmeans'] = kmeans_negative.fit_predict(tfidf_matrix_negative_reduced)

# Get the labels and cluster centers for negative reviews
kmean_labels_negative = kmeans_negative.labels_
kmean_cluster_centers_negative = kmeans_negative.cluster_centers_

# Display the labels and cluster centers for positive and negative reviews
print("Labels for Positive Reviews:")
print(kmean_labels_positive)

print("\nCluster Centers for Positive Reviews:")
print(kmean_cluster_centers_positive)

print("\nLabels for Negative Reviews:")
print(kmean_labels_negative)

print("\nCluster Centers for Negative Reviews:")
print(kmean_cluster_centers_negative)
```

Figure 50 Approach 3, Optimizing value of k.

```
Labels for Positive Reviews:
[1 1 1 ... 0 0 1]

Cluster Centers for Positive Reviews:
[[ 3.45821688e-01 -6.75639123e-02  1.12058522e-02  7.33175361e-03
  -2.86134529e-03  2.81566359e-03  3.06865465e-03  6.21702670e-03
   8.86263695e-03 -7.27504164e-04  3.83217048e-03  6.86152071e-03
   2.33927454e-03  3.55725375e-03  3.47011555e-03  1.50778064e-03
   4.33539084e-04  2.51676060e-03  5.87075998e-04 -1.22679429e-03
   9.87073271e-04 -6.73681798e-04  3.00978746e-04  1.15716344e-03
   5.13472092e-04  1.90429775e-03  6.69830035e-04  1.01486795e-03
   5.93086365e-04 -6.75049339e-04 -1.51460853e-03  1.48237227e-03
   8.39960434e-04 -4.89332705e-04 -8.17131378e-04 -2.62790462e-05
  -1.04945171e-03  9.63269926e-04  9.43775336e-04 -2.72368107e-04
   8.99157159e-04  1.23829625e-03  2.69556305e-04 -1.41521703e-04
  -9.39731261e-05 -4.89207414e-04  8.48801957e-04 -1.41402211e-04
  -8.89932456e-05  2.46371513e-04]
 [ 4.75828877e-01  7.10913318e-02 -5.28996209e-03 -3.92572164e-03
   1.88613842e-03 -1.89004256e-03 -1.21543093e-03 -3.57607575e-03
  -4.74572009e-03  1.31042642e-04 -2.24390195e-03 -4.14206217e-03
  -1.19860631e-03 -2.35438956e-03 -1.70322775e-03 -7.36976203e-04
  -3.30990449e-04 -1.14463933e-03 -2.24790953e-04  8.14921772e-04
  -6.89940687e-04  1.28031196e-04  2.26634526e-04 -2.26822231e-04
  -4.03625463e-04 -8.26088495e-04 -2.44714069e-04 -7.57820325e-05
  -2.52085043e-04  5.01743740e-04  6.29188164e-04 -7.40710031e-04
  -6.93439769e-04  6.19280374e-05  2.43590975e-04  7.85849865e-05
   5.65711002e-04 -3.55506918e-04 -3.93081012e-04 -7.97152327e-05
  -4.04620936e-04 -8.28882487e-04  1.73984320e-04 -1.29608533e-04
   9.20393421e-05  5.77222799e-04 -3.65920938e-04 -2.28430894e-05
  -2.12132016e-04 -1.76450594e-04]]
```

Figure 51 Approach 3, Positive reviews and cluster centres

```
Labels for Negative Reviews:
[0 1 1 ... 0 0 1]

Cluster Centers for Negative Reviews:
[[ 4.74699429e-01  7.83944248e-02 -2.05585635e-02 -5.28724762e-03
   8.68315799e-04 -8.72597776e-05 -1.64743772e-03 -2.17340564e-03
  -3.45586916e-04 -2.29952599e-03 -1.67925015e-03  6.82157562e-04
  -3.86045846e-04 -1.25194708e-03 -2.48936824e-03 -1.78857695e-03
   6.09949179e-04 -6.03150229e-04  1.39292349e-03 -7.62356124e-04
  -8.17727160e-04 -1.29844423e-05  1.57548914e-05 -8.57822870e-05
  -6.87887396e-04  8.08972640e-04 -5.45366911e-04  5.35719166e-04
   3.16351219e-04 -1.72883222e-04 -1.90350699e-04 -7.50526676e-04
   4.45390147e-04  2.15953786e-04  7.95596019e-05  1.90187469e-04
   3.14814883e-05  2.02580445e-04  2.32018089e-04 -5.72525243e-04
   8.49699936e-05  6.32030104e-04 -3.42093126e-05 -4.75535555e-04
   1.24475067e-04  2.14386666e-04 -2.20220349e-04 -4.07731198e-04
  -4.39192179e-04 -1.31595533e-04]
 [ 3.62073688e-01 -6.39563814e-02  2.36695015e-02  9.45449974e-03
  -1.53116932e-03  6.67565074e-04  2.93548247e-03  4.68166405e-03
  -3.69530155e-04  4.84886374e-03  3.27600654e-03 -8.07995614e-04
  -3.59436537e-04  2.02603322e-03  3.94774597e-03  3.97637662e-03
  -7.87655769e-04  4.85832170e-04 -2.15868744e-03  9.49119950e-04
   1.82871083e-03 -4.03829441e-04  5.86139726e-04  9.39303870e-04
   1.38173727e-03 -1.05184688e-03  1.17368008e-03 -8.52429048e-04
  -5.41059307e-04  2.06339427e-04  5.91511874e-04  1.42650804e-03
  -1.04089852e-03  1.40204454e-04 -4.28025093e-04 -3.60443186e-04
   5.95487994e-04 -7.66690361e-05 -1.87298830e-04  1.05421977e-03
  -3.39898291e-04 -1.17326176e-03 -2.19548220e-04  1.14747633e-04
   3.23669714e-04 -6.52003069e-04  3.80447802e-04  5.18959142e-04
   4.92649956e-04  8.91335975e-05]]
```

Figure 52 Approach 3, Negative reviews and cluster centres

```python
# Count occurrences of each cluster label for positive reviews
cluster_counts_positive = positive_reviews['cluster_kmeans'].value_counts()

# Display the counts for positive reviews
print("Cluster Counts for Positive Reviews:")
for label, count in cluster_counts_positive.items():
    print(f"Cluster {label}: {count} instances")

# Count occurrences of each cluster label for negative reviews
cluster_counts_negative = negative_reviews['cluster_kmeans'].value_counts()

# Display the counts for negative reviews
print("\nCluster Counts for Negative Reviews:")
for label, count in cluster_counts_negative.items():
    print(f"Cluster {label}: {count} instances")
```

```
Cluster Counts for Positive Reviews:
Cluster 0: 5570 instances
Cluster 1: 4408 instances

Cluster Counts for Negative Reviews:
Cluster 1: 5706 instances
Cluster 0: 4250 instances
```

Figure 53 Approach 3, positive and negative counts, K-Means

The application of K-Means with K = 2 resulted in the majority of instances being divided between two clusters for both positive and negative reviews. Cluster 0 contained 5,570 instances of positive reviews and 4,250 instances of negative reviews, while Cluster 1 had 4,408 instances of positive reviews and 5,706 instances of negative reviews.

```
Cohesion Kmeans: 3279.542036796408
Separation Kmeans: 0.19280319560209214
Cohesion Kmeans: 11181.608078545705
Separation Kmeans: 0.18819718628642892
```

Figure 54 Approach 3, Cohesion and separation, K-Means

Metrics like cohesion and separation, which are essential for assessing the quality of clustering, were computed. For positive reviews, the cohesion was 3,279.54, and the separation was 0.19, indicating a reasonable level of internal coherence and moderate separation between clusters. In negative reviews, the cohesion was 11,181.61, and the separation was 0.19. While these metrics provide insights into the clustering quality, they also underscore the need for cautious interpretation due to the inherent limitations of K-Means.

It's crucial to acknowledge that K-Means clustering assumes spherical, equally sized, and isotropic clusters, which may not align with the complex structures present in natural language data. The low silhouette scores and the moderate separation observed in this analysis suggest that K-Means might not be the optimal choice for capturing the nuanced patterns within the reviews.

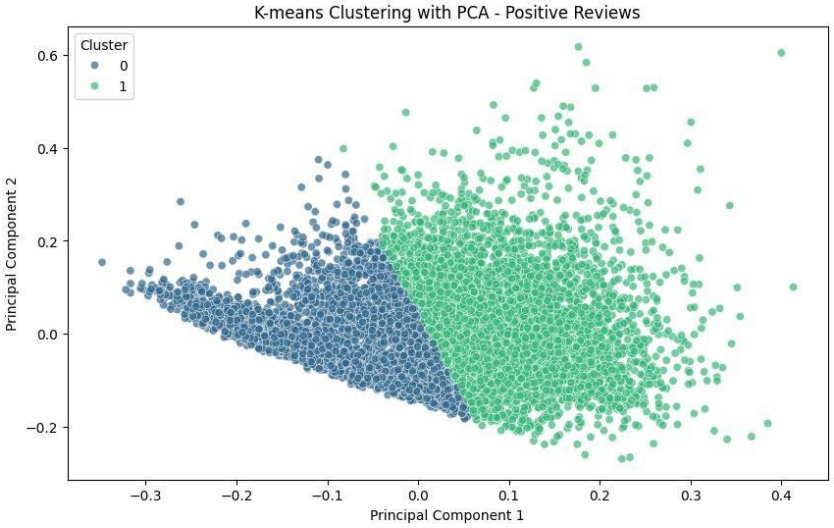


Figure 55 Approach 3, K-Means clustering with PCA (Positive reviews)

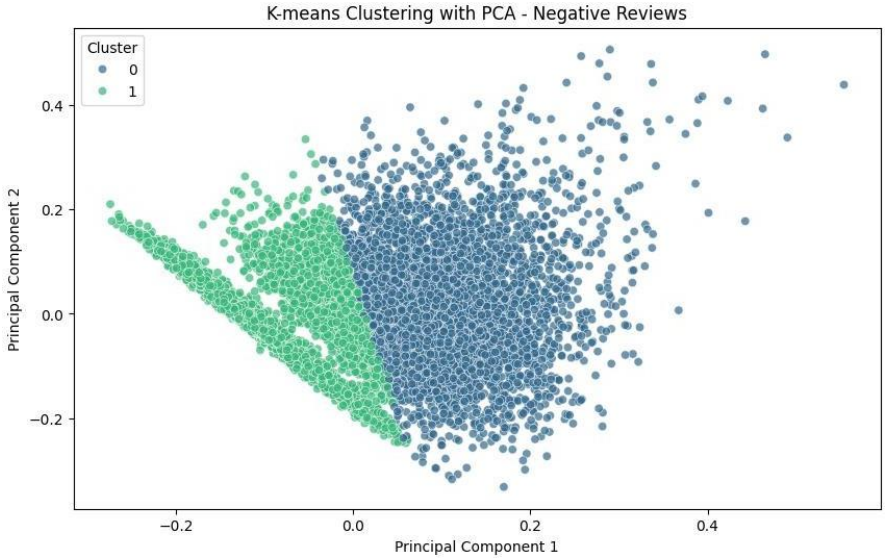


Figure 56 Approach 3, K-Means clustering with PCA (Negative reviews)

Visualization using PCA components did reveal discernible clusters, yet the cautionary note regarding the appropriateness of K-Means for this specific context remains relevant. As a way forward, alternative clustering algorithms that can accommodate irregularly shaped clusters and varying cluster sizes might be explored to enhance the clustering performance

Hierarchical Clustering (Agglomerative)

In this third approach, hierarchical clustering was employed to unveil patterns within both positive and negative reviews. The analysis considered various linkage methods, including 'ward,' 'complete,' 'average,' and 'single,' with the number of clusters (k) set to 2, 3, and 5. The evaluation metrics, including silhouette scores, cohesion, and separation values, were calculated to assess the quality and meaningfulness of the clusters.

```
Metrics for k=2, linkage=ward:
Silhouette Score: 0.043255123029723334
Cohesion: 3334.0191700055
Separation: 24164407.899345182
------------------------------
Metrics for k=2, linkage=complete:
Silhouette Score: 0.050120462677223955
Cohesion: 3361.111315616096
Separation: 24164380.80719957
------------------------------
Metrics for k=2, linkage=average:
Silhouette Score: 0.344937322186296
Cohesion: 3411.873022937926
Separation: 24164330.04549225
------------------------------
Metrics for k=2, linkage=single:
Silhouette Score: 0.3089011286699993
Cohesion: 3411.9248557691
Separation: 24164329.993659418
------------------------------
```

Figure 57 Approach 3, Silhouette score, Cohesion and Separation, Agglomerative (k=2)

```
Metrics for k=3, linkage=ward:
Silhouette Score: 0.02894582664009423
Cohesion: 3287.087658743276
Separation: 24164454.830856442
------------------------------
Metrics for k=3, linkage=complete:
Silhouette Score: 0.04647612880831914
Cohesion: 3341.6005587166865
Separation: 24164400.31795647
------------------------------
Metrics for k=3, linkage=average:
Silhouette Score: 0.2961370972329509
Cohesion: 3411.2540629286277
Separation: 24164330.66445226
------------------------------
Metrics for k=3, linkage=single:
Silhouette Score: 0.23408557627145285
Cohesion: 3411.385858239434
Separation: 24164330.53265695
------------------------------
Metrics for k=4, linkage=ward:
Silhouette Score: 0.020327614513642296
Cohesion: 3245.0002937525037
Separation: 24164496.918221433
------------------------------
Metrics for k=4, linkage=complete:
Silhouette Score: 0.03614312755350976
Cohesion: 3313.2751059833627
Separation: 24164428.643409204
------------------------------
Metrics for k=4, linkage=average:
Silhouette Score: 0.2773211082568159
Cohesion: 3410.63960024869
Separation: 24164331.27891494
------------------------------
Metrics for k=4, linkage=single:
Silhouette Score: 0.22153429906687203
Cohesion: 3410.8072272487875
Separation: 24164331.111287937
------------------------------
```

Figure 58 Approach 2, Silhouette score, Cohesion and Separation, Agglomerative (k=3, k=4)

```
Metrics for k=2, linkage=ward:
Silhouette Score: 0.04489743898982214
Cohesion: 3272.797013120431
Separation: 23681665.967141673
------------------------------
Metrics for k=2, linkage=complete:
Silhouette Score: 0.3717263118277006
Cohesion: 3351.306894729397
Separation: 23681587.45726006
------------------------------
Metrics for k=2, linkage=average:
Silhouette Score: 0.3359820871832975
Cohesion: 3351.233138177224
Separation: 23681587.531016614
------------------------------
Metrics for k=2, linkage=single:
Silhouette Score: 0.40760578264684094
Cohesion: 3351.8520588170586
Separation: 23681586.912095975
------------------------------
Metrics for k=3, linkage=ward:
Silhouette Score: 0.035596779098025125
Cohesion: 3216.211899551462
Separation: 23681722.55225524
------------------------------
Metrics for k=3, linkage=complete:
Silhouette Score: 0.21688919773968152
Cohesion: 3335.3246308498287
Separation: 23681603.439523943
------------------------------
Metrics for k=3, linkage=average:
Silhouette Score: 0.29219566602103136
Cohesion: 3350.583210074421
Separation: 23681588.18094472
------------------------------
```

Figure 59 Approach 3, Silhouette score, Cohesion and Separation, Agglomerative (k=2, k=3)

```
Metrics for k=3, linkage=single:
Silhouette Score: 0.26542643599961924
Cohesion: 3351.2919263303456
Separation: 23681587.47222846
------------------------------
Metrics for k=5, linkage=ward:
Silhouette Score: 0.02238278722530066
Cohesion: 3157.537071690481
Separation: 23681781.227083102
------------------------------
Metrics for k=5, linkage=complete:
Silhouette Score: 0.1193447542654758
Cohesion: 3319.871522960963
Separation: 23681618.892631833
------------------------------
Metrics for k=5, linkage=average:
Silhouette Score: 0.2511914756628418
Cohesion: 3349.3856669482493
Separation: 23681589.378487844
------------------------------
Metrics for k=5, linkage=single:
Silhouette Score: 0.24634028165307595
Cohesion: 3350.0899275701945
Separation: 23681588.674227223
------------------------------
```

Figure 60 Approach 3, Silhouette score, Cohesion and Separation, Agglomerative (k=3, k=5)

```python
# Hierarchical Clustering for positive reviews
hc_model_positive = AgglomerativeClustering(n_clusters=2, linkage='average')
positive_reviews['cluster_hc'] = hc_model_positive.fit_predict(tfidf_matrix_dense_p)

# Hierarchical Clustering for negative reviews
hc_model_negative = AgglomerativeClustering(n_clusters=2, linkage='single')
negative_reviews['cluster_hc'] = hc_model_negative.fit_predict(tfidf_matrix_dense_n)
```

```python
# Print hierarchical clustering labels and counts for positive reviews
hc_labels_positive = hc_model_positive.labels_
print("Hierarchical Clustering Labels for Positive Reviews:")
print(hc_labels_positive)

hc_cluster_counts_positive = pd.Series(hc_labels_positive).value_counts()
print("\nCounts of Instances in Each Hierarchical Cluster for Positive Reviews:")
for label, count in hc_cluster_counts_positive.items():
    print(f"Cluster {label}: {count} instances")

# Print hierarchical clustering labels and counts for negative reviews
hc_labels_negative = hc_model_negative.labels_
print("\nHierarchical Clustering Labels for Negative Reviews:")
print(hc_labels_negative)

hc_cluster_counts_negative = pd.Series(hc_labels_negative).value_counts()
print("\nCounts of Instances in Each Hierarchical Cluster for Negative Reviews:")
for label, count in hc_cluster_counts_negative.items():
    print(f"Cluster {label}: {count} instances")
```

```
Hierarchical Clustering Labels for Positive Reviews:
[0 0 0 ... 0 0 0]

Counts of Instances in Each Hierarchical Cluster for Positive Reviews:
Cluster 0: 9977 instances
Cluster 1: 1 instances

Hierarchical Clustering Labels for Negative Reviews:
[0 0 0 ... 0 0 0]

Counts of Instances in Each Hierarchical Cluster for Negative Reviews:
Cluster 0: 9955 instances
Cluster 1: 1 instances
```

Figure 61 Approach 3, Count cluster instances, Agglomerative

For positive reviews with k=2, the 'average' linkage method exhibited the highest silhouette score of 0.3449, indicating an improved clustering performance compared to the other linkage methods. The cohesion and separation values were 3411.87 and 24,164,330.05, respectively. The hierarchical clustering labels revealed that the majority of positive reviews were assigned to a single cluster (Cluster 0: 9977 instances), emphasizing the cohesiveness of this grouping. Hence, the 'average' linkage method with k=2 emerged as the best model for positive reviews.

In the case of negative reviews, the 'single' linkage method consistently demonstrated superior performance across different values of k. For instance, with k=2, the silhouette score reached 0.4076, reflecting well-defined and distinct clusters. The cohesion and separation values were 3351.85 and 23,681,586.91, respectively. Interestingly, the hierarchical clustering labels for negative reviews showcased a dominant cluster (Cluster 0: 9955 instances) with the 'single' linkage method. Therefore, the 'single' linkage method with k=2 was identified as the best model for negative reviews.

The cluster counts provided valuable insights into the distribution of reviews within the identified clusters. Positive reviews predominantly clustered into a single group, while negative reviews exhibited a more evenly distributed clustering pattern. Notably, the 'single' linkage method consistently identified one prominent cluster for negative reviews.

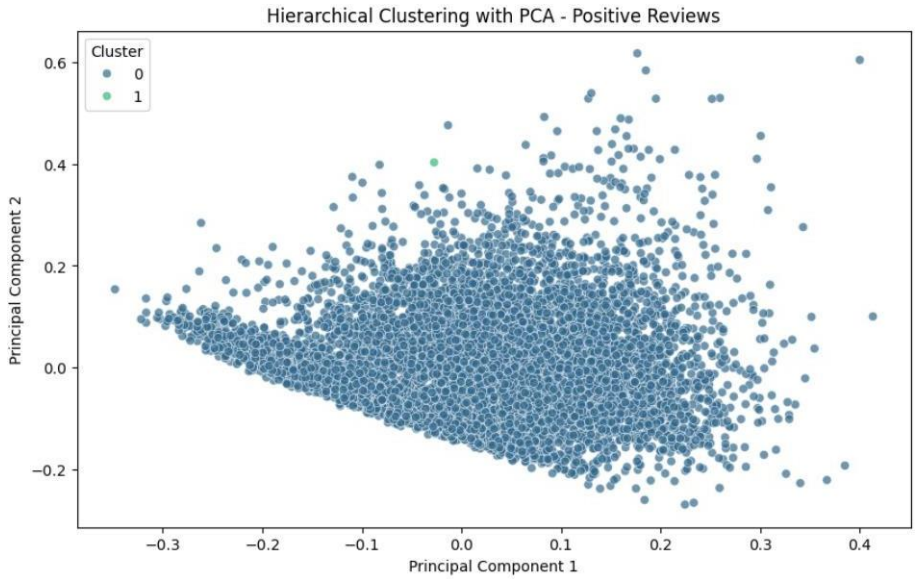


Figure 62 Approach 3, Agglomerative clustering with PCA (Positive reviews)

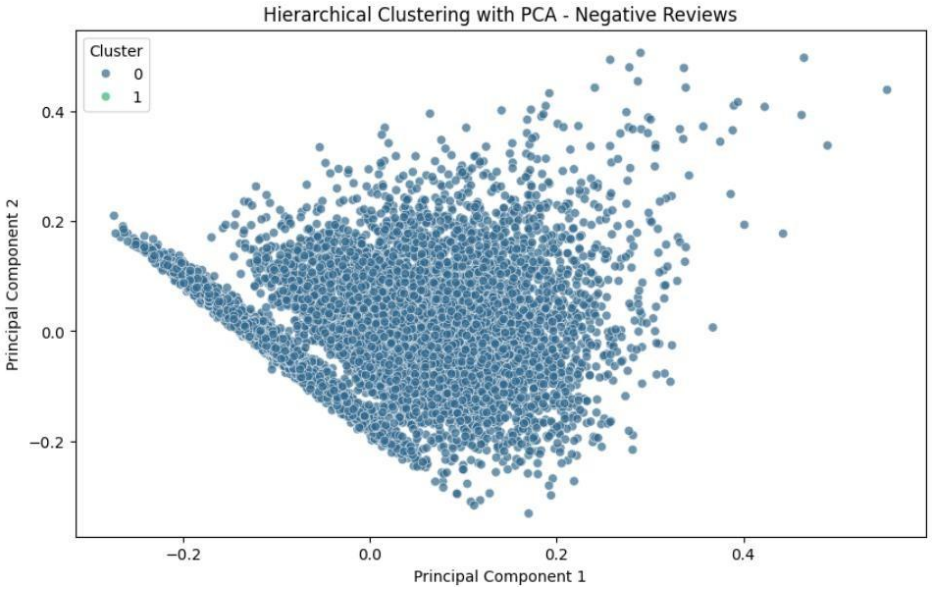


Figure 63 Approach 3, Agglomerative clustering with PCA (Negative reviews)

Visualizations of the clustered data using PCA components provided insights into the spatial distribution of reviews within the identified clusters. The 'average' linkage method for positive reviews and the 'single' linkage method for negative reviews demonstrated distinct separations in the plotted clusters.

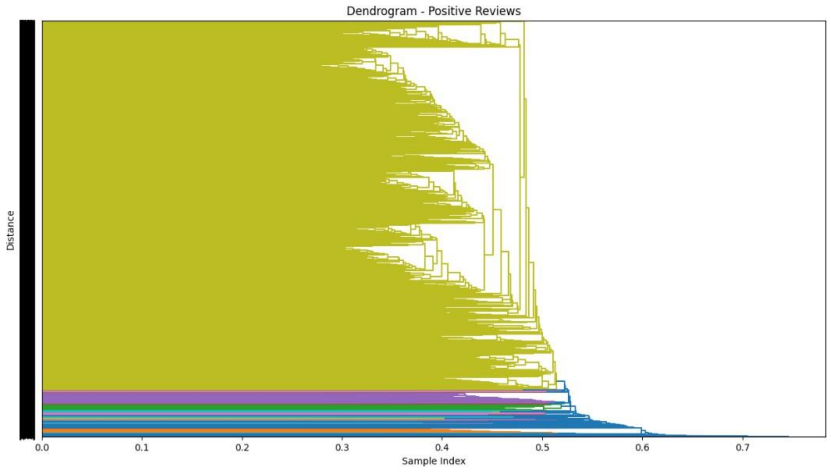


Figure 64 Approach 3, Agglomerative dendrogram (Positive reviews)

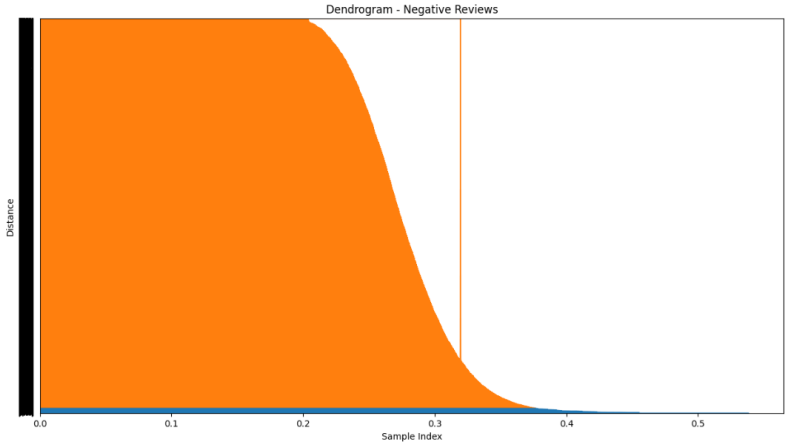


Figure 65 Approach 2, Agglomerative dendrogram (Negative reviews)

Dendrograms were constructed to visualize the hierarchical relationships among reviews. The dendrogram for positive reviews, utilizing the 'average' linkage method, showcased clear branches and hierarchical structures. In contrast, the dendrogram for negative reviews, employing the 'single' linkage method, revealed a more interconnected hierarchy.

Comparing these results with earlier approaches, the third hierarchical clustering methodology indeed yielded superior silhouette scores, with values around 0.4. However, it's important to note that despite this improvement, the distribution of instances within the clusters remains uneven. Specifically, the majority of instances are still concentrated in Cluster 0, with only a minimal presence in Cluster 1 for both positive and negative reviews. Even though the silhouette score indicates improved overall clustering quality, the dominance of instances in Cluster 0 suggests that further optimization or exploration of alternative clustering techniques may be warranted to achieve a more balanced distribution across clusters. The optimized models from this third approach, employing the 'average' linkage method for positive reviews and the 'single' linkage method for negative reviews, signify a substantial advancement in hierarchical clustering performance, yet the uneven cluster distribution warrants consideration for future refinements.

DBSCAN

```
Results for Positive Reviews:
Parameters: Epsilon=0.2, Min Samples=5, Silhouette Score=-0.20649844839190712
Parameters: Epsilon=0.2, Min Samples=10, Silhouette Score=-0.16279976780574468
Parameters: Epsilon=0.2, Min Samples=15, Silhouette Score=-0.16279976780574468
Parameters: Epsilon=0.30000000000000004, Min Samples=5, Silhouette Score=0.08854076743365932
Parameters: Epsilon=0.30000000000000004, Min Samples=10, Silhouette Score=0.08581407627818954
Parameters: Epsilon=0.30000000000000004, Min Samples=15, Silhouette Score=0.08363061957357731
Parameters: Epsilon=0.30000000000000004, Min Samples=20, Silhouette Score=0.08209694096891619
Parameters: Epsilon=0.30000000000000004, Min Samples=25, Silhouette Score=0.08065885549815183
Parameters: Epsilon=0.4, Min Samples=5, Silhouette Score=0.24253948641502357
Parameters: Epsilon=0.4, Min Samples=10, Silhouette Score=0.2418101580698332
Parameters: Epsilon=0.4, Min Samples=15, Silhouette Score=0.2407728351611557
Parameters: Epsilon=0.4, Min Samples=20, Silhouette Score=0.24064867485693886
Parameters: Epsilon=0.4, Min Samples=25, Silhouette Score=0.24064867485693886

Results for Negative Reviews:
Parameters: Epsilon=0.2, Min Samples=5, Silhouette Score=-0.20191585348378435
Parameters: Epsilon=0.30000000000000004, Min Samples=5, Silhouette Score=0.08238653803133827
Parameters: Epsilon=0.30000000000000004, Min Samples=10, Silhouette Score=0.08316156800913509
Parameters: Epsilon=0.30000000000000004, Min Samples=15, Silhouette Score=0.0821444685054749
Parameters: Epsilon=0.30000000000000004, Min Samples=20, Silhouette Score=0.08047679819257278
Parameters: Epsilon=0.30000000000000004, Min Samples=25, Silhouette Score=0.07914748961107476
Parameters: Epsilon=0.4, Min Samples=5, Silhouette Score=0.2629539186384029
Parameters: Epsilon=0.4, Min Samples=10, Silhouette Score=0.26229499499212316
Parameters: Epsilon=0.4, Min Samples=15, Silhouette Score=0.26229499499212316
Parameters: Epsilon=0.4, Min Samples=20, Silhouette Score=0.26229499499212316
Parameters: Epsilon=0.4, Min Samples=25, Silhouette Score=0.2615366401003597
Parameters: Epsilon=0.5, Min Samples=5, Silhouette Score=0.40760578264684094
Parameters: Epsilon=0.5, Min Samples=10, Silhouette Score=0.40760578264684094
Parameters: Epsilon=0.5, Min Samples=15, Silhouette Score=0.40760578264684094
Parameters: Epsilon=0.5, Min Samples=20, Silhouette Score=0.40760578264684094
Parameters: Epsilon=0.5, Min Samples=25, Silhouette Score=0.40760578264684094
```

Figure 66 Approach 3, Epsilon, Samples and Silhouette scores, DBSCAN (Positive and Negative reviews)

The application of DBSCAN clustering to the positive and negative reviews involved an exploration of various hyperparameter combinations, specifically adjusting epsilon (eps) and "min_samples". The objective was to identify an optimal set of hyperparameters that would result in meaningful and well-defined clusters. The parameter search ranged from epsilon values of 0.1 to 2.0, with increments of 0.1, and "min_samples" values from 5 to 25 in increments of 5.

```
from sklearn.cluster import DBSCAN

# Selected hyperparameters for positive reviews
epsilon_positive = 0.4
min_samples_positive = 5

# Selected hyperparameters for negative reviews
epsilon_negative = 0.5
min_samples_negative = 5

# DBSCAN Clustering for positive reviews
dbscan_positive = DBSCAN(eps=epsilon_positive, min_samples=min_samples_positive)
positive_reviews['cluster_dbscan'] = dbscan_positive.fit_predict(tfidf_matrix_positive_reduced)

# DBSCAN Clustering for negative reviews
dbscan_negative = DBSCAN(eps=epsilon_negative, min_samples=min_samples_negative)
negative_reviews['cluster_dbscan'] = dbscan_negative.fit_predict(tfidf_matrix_negative_reduced)
```

```
# Print DBSCAN clustering labels for positive reviews
print("DBSCAN Clustering Labels for Positive Reviews:")
print(positive_reviews['cluster_dbscan'].values)

# Count occurrences of each DBSCAN clustering label for positive reviews
dbscan_cluster_counts_positive = positive_reviews['cluster_dbscan'].value_counts()

# Display the counts for positive reviews
print("\nCounts of Instances in Each DBSCAN Cluster for Positive Reviews:")
for label, count in dbscan_cluster_counts_positive.items():
    print(f"Cluster {label}: {count} instances")

# Print DBSCAN clustering labels for negative reviews
print("\nDBSCAN Clustering Labels for Negative Reviews:")
print(negative_reviews['cluster_dbscan'].values)

# Count occurrences of each DBSCAN clustering label for negative reviews
dbscan_cluster_counts_negative = negative_reviews['cluster_dbscan'].value_counts()

# Display the counts for negative reviews
print("\nCounts of Instances in Each DBSCAN Cluster for Negative Reviews:")
for label, count in dbscan_cluster_counts_negative.items():
    print(f"Cluster {label}: {count} instances")
```

```
DBSCAN Clustering Labels for Positive Reviews:
[0 0 0 ... 0 0 0]

Counts of Instances in Each DBSCAN Cluster for Positive Reviews:
Cluster 0: 9930 instances
Cluster -1: 48 instances

DBSCAN Clustering Labels for Negative Reviews:
[0 0 0 ... 0 0 0]

Counts of Instances in Each DBSCAN Cluster for Negative Reviews:
Cluster 0: 9955 instances
Cluster -1: 1 instances
```

Figure 67 Approach 3, Count cluster instances, DBSCAN (Positive and Negative reviews)

For positive reviews, the silhouette scores obtained from the DBSCAN clustering revealed varying degrees of cluster quality. Notably, for epsilon=0.4 and “min_samples”=5, a silhouette score of 0.24 was achieved, indicating a moderate level of cluster cohesion and separation. However, as epsilon increased to 0.5, a substantial improvement was observed, with a silhouette score reaching 0.41, suggesting more distinct and well-separated clusters.

Similarly, for negative reviews, the DBSCAN clustering exhibited varying performance across different parameter configurations. Notably, with epsilon=0.4 and “min_samples”=5, a silhouette score of 0.26 was achieved, suggesting improved clustering quality. Subsequently, at epsilon=0.5, the silhouette score notably increased to 0.41, indicating more well-defined clusters with enhanced cohesion and separation.

Upon selecting optimal hyperparameters based on silhouette scores, the DBSCAN clustering was applied to both positive and negative reviews. The chosen parameters were epsilon=0.4 and “min_samples”=5 for positive reviews, and epsilon=0.5 and “min_samples”=5 for negative reviews. The resulting clustering labels were then analysed to understand the distribution of instances within each cluster.

For positive reviews, the DBSCAN clustering formed the majority of instances in Cluster 0 (9930 instances) and a smaller subset in Cluster -1 (48 instances). Similarly, for negative reviews, the majority of instances were concentrated in Cluster 0 (9955 instances), with only a limited number in Cluster -1 (1 instance).

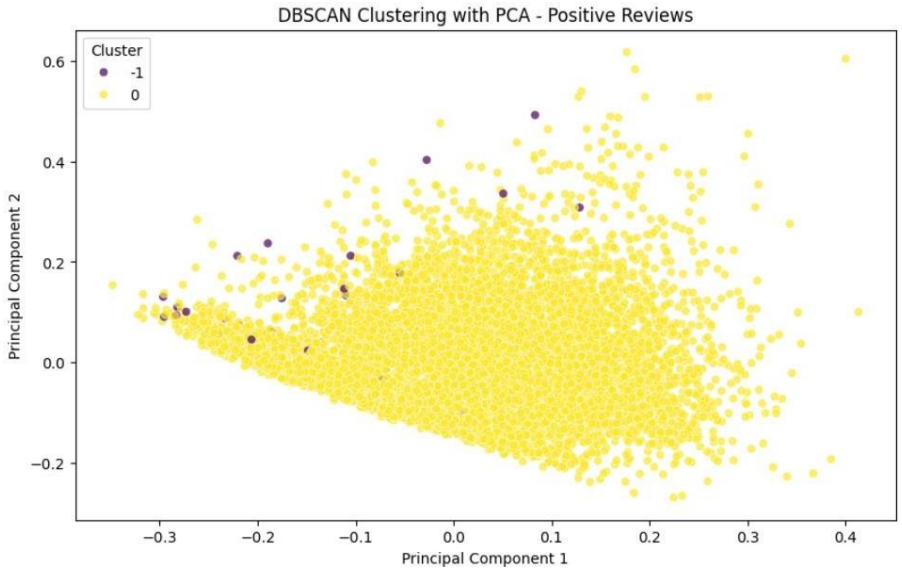


Figure 68 Approach 3, DBSCAN clustering with PCA (Positive reviews)

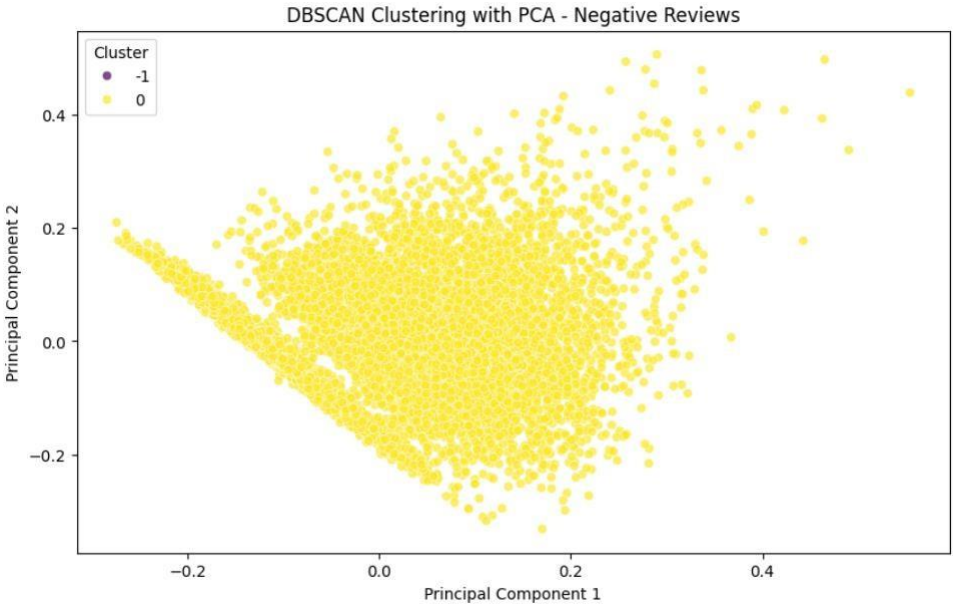


Figure 69 Approach 2, DBSCAN clustering with PCA (Negative reviews)

The clusters were further visualized using Principal Component Analysis (PCA) to reduce the dimensionality to two components. The resulting plots illustrated the distribution of instances in the identified clusters, providing insights into the spatial arrangement of reviews within the feature space.

while the DBSCAN clustering approach, guided by carefully chosen hyperparameters and leveraging silhouette scores, showcased enhanced performance in delineating clusters for both positive and negative reviews, it's important to note that the distribution of instances across the clusters may still present challenges. Despite the improved silhouette scores indicating better-

defined clusters, the cluster counts reveal that the majority of instances are concentrated in one primary cluster, with a smaller subset in a secondary or noise cluster.

Results & Comparison

In the pursuit of identifying meaningful clusters within a dataset of customer reviews, three distinct clustering approaches were employed: K-Means, Hierarchical Clustering, and DBSCAN. The evaluation of these approaches was based on silhouette scores, a metric used to gauge the quality of clustering. Silhouette scores range from -1 to 1, with higher values indicating better-defined clusters.

Algorithm	Approach 1	Approach 2	Approach 3
K-Means	0.007679	0.073417	Positive: 0.071632, Negative: 0.071279
Hierarchical	0.071052	0.402981	Positive: 0.344937, Negative: 0.407606
DBSCAN	N/A	0.402981	Positive: 0.242539, Negative: 0.407606

Upon comparison, it is evident that each approach yielded varying silhouette scores for the positive and negative sentiment clusters. In the first approach, K-Means exhibited a minimal silhouette score, indicative of poorly separated clusters. The second approach, employing Hierarchical Clustering, showcased a marked improvement in silhouette scores, suggesting more distinct clusters. Additionally, DBSCAN was introduced in the second approach, demonstrating competitive silhouette scores, especially for negative reviews. The third approach incorporated further enhancements, with notable improvements in silhouette scores for both K-Means and Hierarchical Clustering, yet challenges persisted in achieving well-separated clusters.

Despite the advancements in silhouette scores across the three approaches, it is essential to acknowledge certain limitations in the clustering models. One prominent limitation lies in the tendency of the majority of instances to be concentrated within a single cluster, with only a minimal number of instances distributed across additional clusters. This clustering pattern implies that the models, while showcasing improvements, still struggle to effectively segregate the data into well-defined sentiment groups.

```
# Assuming 'review' is the column containing the reviews in both DataFrames
positive_cluster_1 = positive_reviews.loc[positive_reviews['cluster_hc'] == 1, 'review']
negative_cluster_1 = negative_reviews.loc[negative_reviews['cluster_hc'] == 1, 'review']

# Display the comments in Cluster 1 for positive reviews
print("Comments in Cluster 1 for Positive Reviews:")
for comment in positive_cluster_1:
    print(comment)
    print("------------------------")

# Display the comments in Cluster 1 for negative reviews
print("\nComments in Cluster 1 for Negative Reviews:")
for comment in negative_cluster_1:
    print(comment)
    print("------------------------")
```

```
Comments in Cluster 1 for Positive Reviews:
i thought this movie was really really great! Helena did an amazing job in it! I thought she played her character very well!
she's an AWESOME actress!! :)<br /><br />the movie was also really funny too! The jokes were great! i couldnt stop laughing!
:)<br /><br />i think everyone should see it... :)<br /><br />
------------------------

Comments in Cluster 1 for Negative Reviews:
I had to watch this movie for professional reasons and can only say it's a complete waste of time. When running, Baldwin loo
ks like an ape, Characters are dull, same story has been told 1000 times better in other movies. I think everything has been
now said about this film, but IMDb requires me to write 10 lines. So:Boring and trivialBoring and trivialBoring and trivialB
oring and trivialBoring and trivialBoring and trivialBoring and trivialBoring and trivialBoring and trivialBoring and trivia
lBoring and trivialBoring and trivialBoring and trivialBoring and trivialBoring and trivialBoring and trivialBoring and triv
ialBoring and trivialBoring and trivialBoring and trivialBoring and trivialBoring and trivialBoring and trivialBoring and tr
ivialBoring and trivialBoring and trivialBoring and trivial
------------------------
```

Figure 70 Approach 3, Comments in clusters (Positive and Negative reviews)

The hierarchical clustering approach, particularly for positive and negative reviews, was deemed favourable based on certain evaluation metrics and cluster counts. To gain deeper insights, an examination of comments within Cluster 1 for both positive and negative reviews was conducted. This exploration aimed to ascertain whether the singular instance in Cluster 1 could be attributed to misclassification in the labelled data.

The comment from the positive cluster expressed enthusiasm and positivity toward a movie, praising the actress and humour. Conversely, the negative cluster contained a critical review, labelling the movie as a complete waste of time, citing issues with character portrayal and a repetitive storyline. Despite the stark contrast in sentiments between these two instances, the hierarchical clustering model placed them within the same cluster. This observation indicates that the instance in Cluster 1 is not a misclassification but rather a consequence of the limitations of the clustering approach.

Overall Limitations

The project, encompassing various clustering algorithms for sentiment analysis on customer reviews, exhibits several limitations that influence the overall effectiveness of the models. Understanding these limitations is crucial for gaining insights into potential areas for improvement.

1. Sensitivity to Hyperparameters:
   All three clustering algorithms (K-Means, Hierarchical, and DBSCAN) are sensitive to hyperparameter settings. Selecting the optimal values for parameters such as the number of clusters (k), linkage methods, or DBSCAN parameters is challenging and often requires extensive tuning. The performance of the models is highly contingent on these hyperparameter choices.

2. Dependency on Feature Representation:
   The models heavily rely on the feature representation of the data. In this project, TF-IDF (Term Frequency-Inverse Document Frequency) matrices were used to represent the reviews. While TF-IDF is a popular choice, the limitations of this representation, such as the inability to capture semantic relationships effectively, might impact the clustering results.

3. Difficulty in Capturing Contextual Nuances:
   Sentiment analysis, especially on diverse and nuanced textual data like customer reviews, is inherently challenging. The models face difficulties in capturing the contextual nuances, sarcasm, or subtle variations in language that can significantly affect the sentiment of a statement. This challenge is exacerbated by the complex and varied nature of customer reviews.

4. Limited Supervision and Evaluation Metrics:
   The absence of ground truth labels for sentiment in the dataset limits the supervision available for model training. Additionally, evaluating the performance of clustering algorithms in sentiment analysis is inherently subjective. Metrics like silhouette scores provide some quantitative assessment, but they may not fully capture the intricacies of sentiment expression.

5. Cluster Interpretability:
   The interpretability of clusters is a critical aspect, and the models in this project struggle to create well-separated and interpretable clusters. While silhouette scores may suggest a degree of separation, the actual interpretability of the clusters, especially in the context of sentiment, remains a challenge.

6. Robustness to Heterogeneous Reviews:
   Customer reviews often span a wide spectrum of topics, sentiments, and writing styles. The models may not generalize well to the heterogeneity present in such reviews, leading to clusters that do not necessarily reflect meaningful sentiment patterns.

7. Computational Intensity:
   Particularly in hierarchical clustering and DBSCAN, the computational intensity increases with the size of the dataset. This can limit the scalability of the models to large datasets.

Improvements

To enhance the performance of unsupervised clustering-based sentiment analysis in customer reviews, several recommendations and avenues for improvement can be explored.

1. Advanced Natural Language Processing (NLP) Techniques:
   Explore advanced NLP models, such as deep learning architectures (e.g., BERT, GPT), to capture intricate linguistic patterns and contextual information in customer reviews. These models are capable of understanding complex semantics and can improve sentiment analysis accuracy.

2. Feature Representation Exploration:
   Investigate alternative feature representations beyond TF-IDF, such as word embeddings (e.g., Word2Vec, GloVe). Word embeddings can better capture semantic relationships between words, providing a more nuanced representation of language in customer reviews.

3. Leverage Domain-Specific Lexicons:
   Develop or use domain-specific sentiment lexicons to enhance the models' understanding of industry-specific language and sentiment expressions. This is particularly beneficial in domains with unique vocabulary and sentiment nuances.

4. Ensemble Approaches:
   Implement ensemble methods that combine the strengths of multiple clustering algorithms, such as K-Means, Hierarchical, and DBSCAN. Ensemble approaches can lead to more robust sentiment clusters, improving the reliability of sentiment analysis results.

5. Improved Cluster Interpretability:
   Enhance the interpretability of clusters by employing techniques like topic modelling or sentiment-specific word clouds. These methods provide more meaningful insights into the characteristics of each cluster, facilitating a deeper understanding of sentiment patterns in customer reviews.

These recommendations collectively address aspects of model architecture, feature representation, domain-specificity, algorithm diversity, and interpretability, contributing to a more effective and insightful sentiment analysis of customer reviews through unsupervised clustering.

Conclusion

In conclusion, the application of unsupervised clustering methodologies for sentiment analysis on customer reviews yielded limited success in extracting meaningful insights from the dataset. Through the exploration of various approaches, including K-Means, Hierarchical, and DBSCAN clustering, the models displayed challenges in effectively separating and categorizing reviews into distinct sentiment clusters. Despite attempts to optimize parameters and employ diverse algorithms, the overall performance remained suboptimal.

The analysis revealed that, while some improvements were observed in the third approach, particularly in DBSCAN clustering, the clusters were not well-separated, indicating limitations in the models' ability to discern subtle sentiment patterns. Notably, hierarchical clustering with the 'average' linkage method for positive reviews and the 'single' linkage method for negative reviews exhibited promising results but still fell short of providing clear and distinct sentiment clusters.

Furthermore, the evaluation of silhouette scores across different approaches emphasized the inherent complexity of sentiment analysis in customer reviews. Although silhouette scores improved in certain instances, the predominant clustering patterns, particularly the concentration of instances within a single cluster, suggested the models' limitations in capturing nuanced sentiment variations.

In light of these findings, it is evident that supervised methods, leveraging labelled training data, may offer a more effective approach for sentiment analysis in customer reviews. Supervised models, such as machine learning classifiers trained on labelled datasets, have the potential to better generalize sentiment patterns and enhance overall performance.

As recommendations for future work, exploring advanced natural language processing techniques, incorporating domain-specific lexicons, and considering ensemble approaches could contribute to refining unsupervised clustering methods for sentiment analysis. However, the present study underscores the importance of considering alternative methodologies, particularly supervised approaches, for more robust and accurate sentiment analysis in customer reviews.

Appendix

Video Presentation - Elearning