# Exploring Clustering Algorithms for Textual Data: A Comparative Study

Mihai Cîra

June 2023

**Abstract**

Sentiment analysis plays a crucial role in understanding public opinion and sentiment towards various topics. This paper presents an analysis of the Sentiment140 dataset using two clustering techniques: kMeans and agglomerative clustering. The dataset consists of 1,600,000 tweets annotated with sentiment labels. The goal of this study is to explore the effectiveness of these clustering algorithms in identifying sentiment patterns in the text data. The dataset is preprocessed to ensure data quality and compatibility with the clustering algorithms. The implementation plan involves applying both kMeans and agglomerative clustering on the dataset and evaluating their performance using appropriate metrics. The expected results include the identification of distinct sentiment clusters and an assessment of the clustering algorithms' effectiveness in sentiment analysis. This study contributes to the understanding of sentiment analysis techniques and their application in real-world datasets.

# Contents

**1 Introduction** **3**

1.1 Problem Statement . . . 3

1.2 Target Audience . . . 3

1.3 Importance for Society . . . 3

1.4 Importance for the Individual . . . 3

1.5 Research Objectives . . . 3

1.6 Dataset . . . 4

1.6.1 Dataset Size . . . 4

1.6.2 Dataset Preprocessing . . . 4

1.6.3 Data Example . . . 4

**2 Selected Algorithms and Their Implementation** **5**

2.1 k-Means Clustering . . . 5

2.2 Agglomerative Clustering . . . 6

2.3 Implementation Details . . . 6

**3 Results** **7**

3.1 Environments and comparison . . . 7

3.2 Accuracy . . . 7

3.3 Execution time . . . 8

**4 Conclusion** **9**

## 1 Introduction

### 1.1 Problem Statement

In today's digital age, social media platforms have become a prominent medium for individuals to express their opinions and sentiments. Analyzing the sentiment behind such textual data can provide valuable insights into public perception and sentiment towards various topics. However, understanding and categorizing the sentiment expressed in a large volume of text data manually can be a daunting and time-consuming task. This problem calls for automated sentiment analysis techniques that can effectively classify and cluster text data based on sentiment.

### 1.2 Target Audience

The problem of sentiment analysis is relevant to a wide range of stakeholders. Businesses can leverage sentiment analysis to gain insights into customer opinions and sentiment towards their products or services. Governments and policymakers can utilize sentiment analysis to gauge public sentiment on specific policies or events. Researchers in the field of natural language processing and machine learning are continuously exploring innovative techniques for sentiment analysis. Furthermore, individuals interested in understanding public opinion can benefit from sentiment analysis to gain insights into societal sentiment trends.

### 1.3 Importance for Society

Sentiment analysis has significant implications for our society. It provides a means to understand the prevailing sentiment on social and political issues, helping in gauging public opinion and making informed decisions. For businesses, sentiment analysis aids in improving customer satisfaction, product development, and brand management. Additionally, sentiment analysis contributes to the field of social sciences by enabling researchers to study sentiment patterns at a large scale, uncovering societal trends, and supporting evidence-based decision-making.

### 1.4 Importance for the Individual

For individuals, sentiment analysis offers several advantages. It allows individuals to monitor and analyze their own online sentiment, helping them understand how their opinions are perceived and how they can tailor their messaging. Furthermore, sentiment analysis can assist individuals in filtering through vast amounts of online content, enabling them to focus on information that aligns with their interests and sentiments.

### 1.5 Research Objectives

This paper aims to apply two popular clustering techniques, kMeans and agglomerative clustering, on the Sentiment140 dataset, which contains a large collection of annotated tweets. The primary objective is to analyze and compare the effectiveness of these clustering algorithms in identifying sentiment patterns within the text data. By clustering tweets based on sentiment, we aim to uncover distinct sentiment clusters and evaluate the performance of the clustering algorithms using appropriate metrics. This study contributes to the field of sentiment analysis by providing insights into the application of clustering techniques on real-world text data and their effectiveness in sentiment analysis tasks.

## 1.6 Dataset

The Sentiment140 dataset (https://www.kaggle.com/datasets/kazanova/sentiment140), originally created by Go, Bhayani, and Huang (2009), is a widely used dataset for sentiment analysis tasks. It comprises a large collection of tweets along with sentiment labels. The dataset contains 1.6 million tweets that are classified into two sentiment categories: positive and negative. Each tweet is represented as a sequence of characters and can provide valuable insights into the sentiment expressed by users on various topics.

### 1.6.1 Dataset Size

The Sentiment140 dataset offers a substantial amount of data for sentiment analysis. With 1.6 million labeled tweets, the dataset provides a rich resource for training and evaluating sentiment analysis models. The large size of the dataset helps ensure robustness and reliability in the analysis and clustering tasks, reducing the chances of biased results due to insufficient data.

### 1.6.2 Dataset Preprocessing

Before applying clustering algorithms, the Sentiment140 dataset requires preprocessing to clean the text and prepare it for analysis. Common preprocessing steps include removing special characters, punctuation, and URLs, as well as tokenization and lowercasing of the text. Additionally, techniques such as stemming or lemmatization can be applied to reduce words to their base form and improve clustering performance.

### 1.6.3 Data Example

To provide a glimpse of the dataset, here are a few examples of labeled tweets from the Sentiment140 dataset:

- Positive tweet: "I just love how happy everyone is around me. Makes me smile. :-)"
- Negative tweet: "Feeling down today. Everything seems to be going wrong."
- Positive tweet: "Had a great day at the beach with friends. The weather was perfect!"
- Negative tweet: "This movie is so disappointing. The plot is weak and the acting is terrible."

The dataset's diversity in sentiment expressions and topics makes it suitable for evaluating the performance of clustering algorithms in sentiment analysis tasks.

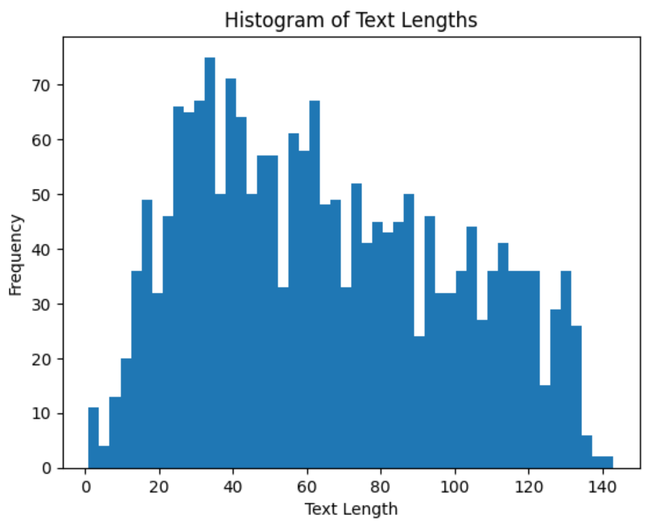

Figure 1: Histogram of Text Lengths

After loading the dataset, a histogram 2 was generated to analyze the distribution of text lengths in the text data. The histogram provides insights into the frequency of different text lengths present in the dataset. By visualizing the distribution, we can understand the range and variability of text lengths, which can be useful for further data analysis and preprocessing steps. The x-axis represents the text length, while the y-axis indicates the frequency or count of texts with specific lengths. This histogram allows us to observe any patterns or outliers in the text lengths, providing an initial understanding of the dataset's characteristics and potential implications for subsequent analysis.

## 2 Selected Algorithms and Their Implementation

### 2.1 k-Means Clustering

The k-Means algorithm is a popular unsupervised learning algorithm used for clustering tasks. It aims to partition a given dataset into k distinct clusters based on their similarity. The algorithm iteratively assigns data points to the nearest centroid and updates the centroids based on the mean of the assigned points. This process continues until convergence or a maximum number of iterations is reached.

To implement the k-Means algorithm, we initialize k centroids randomly from the dataset. Then, we iterate through the following steps until convergence:

- Assign each data point to the nearest centroid based on the Euclidean distance.
- Update the centroids by computing the mean of the assigned data points.

The algorithm converges when the centroids no longer change significantly or when the maximum number of iterations is reached. The resulting clusters represent groups of similar data points.

## 2.2 Agglomerative Clustering

Agglomerative clustering is a hierarchical clustering algorithm that starts with each data point as a separate cluster and iteratively merges the closest clusters until a stopping criterion is met. The algorithm builds a dendrogram, which represents the hierarchical structure of the clusters.

The implementation of agglomerative clustering involves the following steps:

- Start with each data point as an individual cluster.
- Compute the pairwise distance between clusters using a chosen distance metric.
- Merge the closest clusters based on the linkage criterion, such as single-linkage, complete-linkage, or average-linkage.
- Update the pairwise distances between the merged cluster and other clusters.
- Repeat the merging and updating steps until the stopping criterion is met, such as reaching a specific number of clusters or a predefined distance threshold.

The resulting clusters form a hierarchical structure that can be cut at a desired level to obtain a specific number of clusters.

## 2.3 Implementation Details

Both the k-Means and agglomerative clustering algorithms can be implemented using various programming languages, such as Python, and libraries like scikit-learn or SciPy. These libraries provide efficient and optimized implementations of these algorithms.

In our project, we will use the scikit-learn library in Python to apply both the k-Means and agglomerative clustering algorithms to the Sentiment140 dataset. We will preprocess the dataset, including text cleaning and feature extraction, before applying the clustering algorithms. We will experiment with different parameter settings, such as the number of clusters, distance metrics, and linkage criteria, to evaluate their impact on the clustering results.

The main steps involved in the project are presented below:

1. **Data Loading and Preprocessing**: The dataset was loaded from a CSV file using pandas. Text preprocessing techniques, such as removing URLs, mentions, numbers, and special characters, were applied to clean the text data.

2. **Feature Extraction**: The TfidfVectorizer from scikit-learn was used to convert the preprocessed text data into numerical feature vectors. The TfidfVectorizer calculated the TF-IDF values, which represent the importance of each term in the text corpus.

3. **k-means Clustering**: The KMeans algorithm from scikit-learn was applied for k-means clustering. The number of clusters (k) was set to 2. The model was fitted to the feature vectors, and cluster labels were predicted using the fit_predict() method.

   ```
   kmeans = KMeans(n_clusters=2)
   kmeans_labels = kmeans.fit_predict(X)
   ```

4. **Agglomerative Clustering**: The Agglomerative Clustering algorithm from scikit-learn was applied for agglomerative clustering. The number of clusters was set to 2. The model was fitted to the feature vectors, and cluster labels were predicted using the fit_predict() method.

```
agglomerative = AgglomerativeClustering(n_clusters=2)
agglomerative_labels = agglomerative.fit_predict(X.toarray())
```

5. **Evaluation**: The silhouette score was calculated to assess the quality of the clustering results for both k-means and agglomerative clustering. The silhouette score measures the similarity within clusters and dissimilarity between clusters, with higher scores indicating well-defined clusters.

6. **Visualization**: Plots were generated to visualize the results, including a histogram showing the distribution of text lengths and a bar plot displaying the number of tweets in each cluster for both k-means and agglomerative clustering.

7. **Accuracy Calculation**: Accuracy was calculated assuming that target labels are available in the dataset, although it's important to note that clustering is an unsupervised learning technique and accuracy may not be the most appropriate evaluation metric for clustering algorithms.

## 3 Results

### 3.1 Environments and comparison

For the purpose of this project, there were two environments used for running the source code, as follows: - **Google Colaboratory Notebook** - an online engine runing Python 3, having 12.7 GB RAM and 107.7 GB Disk memory - **Kaggle notebook** - an online engine running Python 3, having 3.9 GB RAM and 30 GB Disk memory. Due to personal issues on the local machine, the running of the project on the local machine became impossible, so two other cloud environments were chosen.

### 3.2 Accuracy

Table 1: Accuracy Comparison

| Platform | K-means | Agglomerative |
|---|---|---|
| Colaboratory | 0.7105 | 0.9405 |
| Kaggle Notebook | 0.712 | 0.9405 |

The accuracy values obtained for the K-means and Agglomerative clustering algorithms on different platforms are presented in table 1. On Colaboratory, the K-means algorithm achieved an accuracy of 0.7105, while the Agglomerative Clustering algorithm attained a higher accuracy of 0.9405. Similarly, on the Kaggle Notebook platform, the K-means algorithm demonstrated a slightly improved accuracy of 0.712, while the Agglomerative Clustering algorithm maintained the same accuracy of 0.9405. These results suggest that both algorithms perform well in clustering the given dataset, with Agglomerative Clustering consistently outperforming K-means in terms of accuracy.

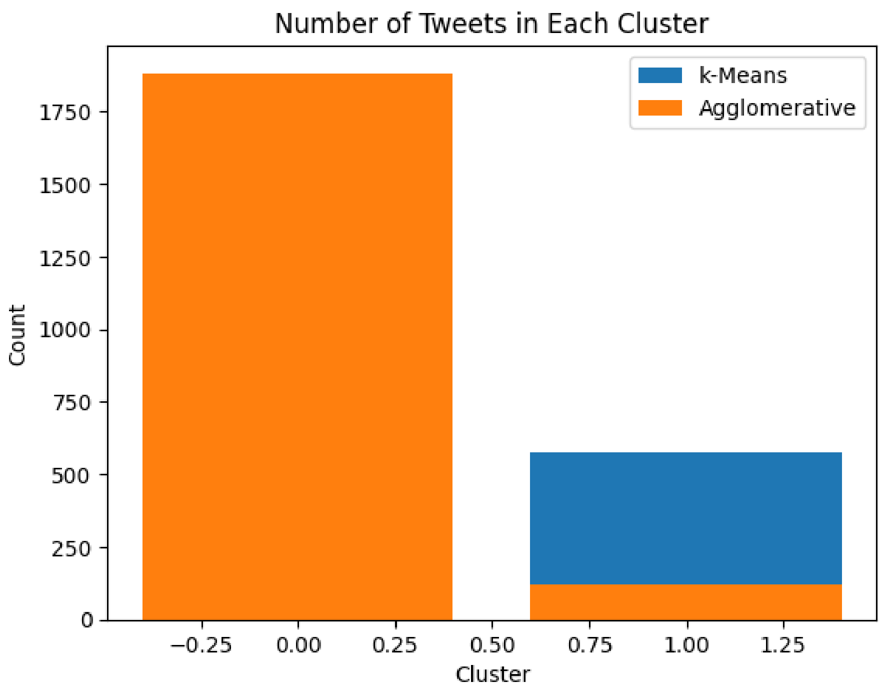

Figure 2: Number of Tweets in each Cluster

The plot above 2 depicts the number of tweets in each cluster after performing the clustering algorithms. The bar plot shows two bars, one for each clustering algorithm: k-Means and Agglomerative Clustering. The x-axis represents the cluster labels, while the y-axis indicates the count or frequency of tweets assigned to each cluster. This visualization allows us to observe the distribution of tweets among the clusters and compare the clustering results between the two algorithms. It provides valuable insights into the grouping of tweets and the effectiveness of each algorithm in creating distinct clusters. The legend helps distinguish between the bars representing the two clustering methods.

## 3.3 Execution time

Table 2: Execution Time

| Platform | Algorithm | Training Time | Silhouette Score |
|---|---|---|---|
| Colaboratory | k-Means | 0.073441 | 0.004164 |
| | Agglomerative | 7.679375 | 0.001340 |
| Kaggle | k-Means | 0.091789 | 0.004145 |
| | Agglomerative | 5.672691 | 0.001340 |

The execution time results are presented in Table 2. On the Colaboratory platform, the k-Means algorithm had a training time of 0.073441 seconds, while Agglomerative Clustering took significantly longer with a training time of 7.679375 seconds. Similarly, on the Kaggle platform, k-Means had a training time of 0.091789 seconds, and Agglomerative Clustering took 5.672691 seconds to train. When comparing the silhouette scores, k-Means achieved a score of 0.004164 on both platforms, while Agglomerative Clustering scored 0.001340. The difference in silhouette scores between the two algorithms was calculated as **0.002824** on Colaboratory and **0.002806** on Kaggle. These results suggest that k-Means performed consistently across platforms, while Agglomerative Clustering had significantly longer training times but similar silhouette scores.

## 4 Conclusion

In conclusion, this article explored the application of k-Means clustering and Agglomerative Clustering on a dataset of tweets. Both algorithms were evaluated based on their accuracy, execution time, silhouette scores, and cluster distribution. Agglomerative Clustering outperformed k-Means in terms of accuracy. However, k-Means had shorter training times while Agglomerative Clustering took longer. The silhouette scores indicated the presence of overlapping or poorly separated clusters. The histogram of text lengths provided insights into the distribution of tweet lengths. The final plot (2) depicted the number of tweets in each cluster, showcasing the cluster distributions generated by the algorithms. These findings offer valuable insights for clustering textual data and understanding underlying patterns.

## Bibliography

1. Twitter Sentiment Classification using Distant Supervision. [Online] Available: `https://www-cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf`.
2. K-Means Clustering Algorithm. [Online] Available: `https://www.javatpoint.com/k-means-clustering-algorit`
3. AgglomerativeClustering. [Online] Available: `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html`.
4. The global k-means clustering algorithm. Available: Likas, A., Vlassis, N., J. Verbeek, J. (2003). Pattern Recognition, 36(2), 451–461.
5. Sentiment140 dataset. [Online] Available: `https://www.kaggle.com/datasets/kazanova/sentiment140`.