Data Mining and Machine Learning Assignment 2

Aman (MDS202305), Ankan Kar (MCS202303)

March 31, 2024

1 Introduction

This task involves using K-means clustering to group documents from three datasets: Enron emails, NIPS blog entries, and KOS blog entries. The goal is to find the best number of clusters, K, by using the Jaccard index to measure document similarity based on word overlap. This helps identify the optimal clustering setup for each dataset.

2 Methods and Proceesing

We have developed and incorporated the following methods into our KMeans Clustering Model.

2.1 Calculating Jaccard Index

The Jaccard index, a widely-used similarity measure, quantifies the similarity between two sets by comparing their intersection to their union. Formally, it is defined as:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

where: - A and B represent the sets being compared. - $|A \cap B|$ denotes the size of the intersection of sets A and B. - $|A \cup B|$ denotes the size of the union of sets A and B.

The Jaccard index ranges from 0 to 1, where a value of 0 indicates no overlap between sets (complete dissimilarity), and a value of 1 indicates complete overlap (perfect similarity).

To compute the Jaccard distance (dissimilarity measure), we subtract the Jaccard index from 1:

$$D_J(A, B) = 1 - J(A, B)$$

2.2 KMeans++ Algorithm for Centroid Selection

Algorithm Summary: This algorithm initializes centroids for K-means clustering using a modified K-means++ approach. Here's a brief overview of its steps:

1. **Initialization:** Determine dataset size and create an empty centroid array.
2. **First Centroid:** Randomly select the first centroid from the dataset.
3. **Calculate Jaccard Distances:** Compute pairwise Jaccard distances between data points and centroids.

4. **Select Next Centroid:** Iteratively select centroids with probability based on distance from previously selected centroids.

5. **Update Distances:** Update distances array with minimum distances to new centroids.

6. **Return:** Return the array of centroids.

This approach efficiently selects initial centroids by ensuring they are well spread out across the dataset, which often leads to faster convergence and better clustering results compared to random initialization.

2.3 Mini-Batch KMeans

Given the dataset's size and sparsity (over 95%), we use the mini-batch method in KMeans clustering. This involves processing small random subsets, or mini-batches, of the dataset instead of the entire dataset at once. Utilizing mini-batches enhances computational efficiency, making the algorithm more scalable for large datasets. It also promotes faster convergence, particularly when the dataset size prevents processing all data in memory simultaneously.

Here's a breakdown of the mini-batch KMeans algorithm:

1. Initialization: Centroids are initialized using a predefined function.

2. Iteration: The algorithm iterates until convergence or reaching the maximum iterations..

3. Mini-Batch Selection: Each iteration processes a random subset, reducing computational load and memory requirements.

4. Cluster Assignment: Mini-batches are assigned to clusters based on proximity to current centroids.

5. Centroid Update: Centroids are updated based on data points within each cluster.

6. Within-Cluster Sum of Squares (WCSS): WCSS is calculated to evaluate clustering quality.

7. Convergence Check: The algorithm terminates if centroid changes fall below a specified tolerance threshold.

8. Iteration Update: If convergence isn't reached, the iteration counter is incremented, and the process continues with the next mini-batch.

Overall, mini-batch KMeans efficiently handles large and sparse datasets by iteratively processing small random subsets, improving scalability and computational efficiency compared to traditional batch processing methods.

3 Results and Observations

3.1 KOS Text Dataset

The dataset contains 3.88 MB of data from 3430 documents. We experimented with Batch Size, Maximum Iterations, and Tolerance to optimize Mini Batch KMeans convergence. Below is the WCSS vs. k plot for the KOS dataset.

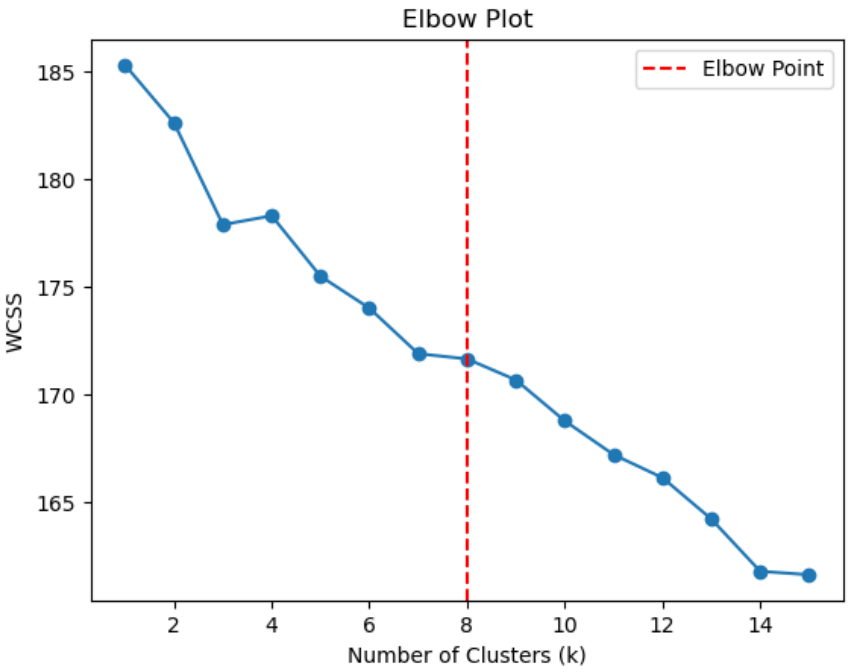

Figure 1: Plot for KOS Dataset

The red dotted line in the plot signifies the optimal K (= 8) for clustering in the KOS Dataset which took 30 seconds to run. It marks where the decrease in WCSS slows, indicating diminishing returns with additional clusters.

3.2 NIPS Text Dataset

The dataset contains 8.11 MB of data from 1500 documents. We experimented with Batch Size, Maximum Iterations, and Tolerance to optimize Mini Batch KMeans convergence. Below is the WCSS vs. k plot for the NIPS dataset

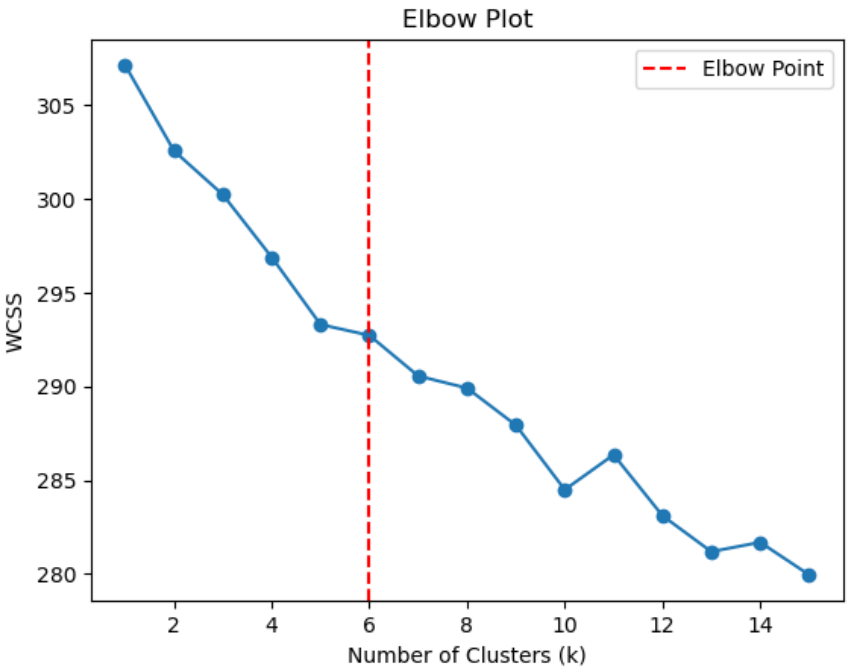

Figure 2: Plot for NIPS Dataset

The red dotted line in the plot signifies the optimal K (= 6) for clustering in the NIPS Dataset which took 190 seconds to run. It marks where the decrease in WCSS slows, indicating diminishing returns with additional clusters.

3.3 ENRON Text Dataset

The dataset contains 47.4 MB of data from 39861 documents. We experimented with Batch Size, Maximum Iterations, and Tolerance to optimize Mini Batch KMeans convergence. Below is the WCSS vs. k plot for the NIPS dataset

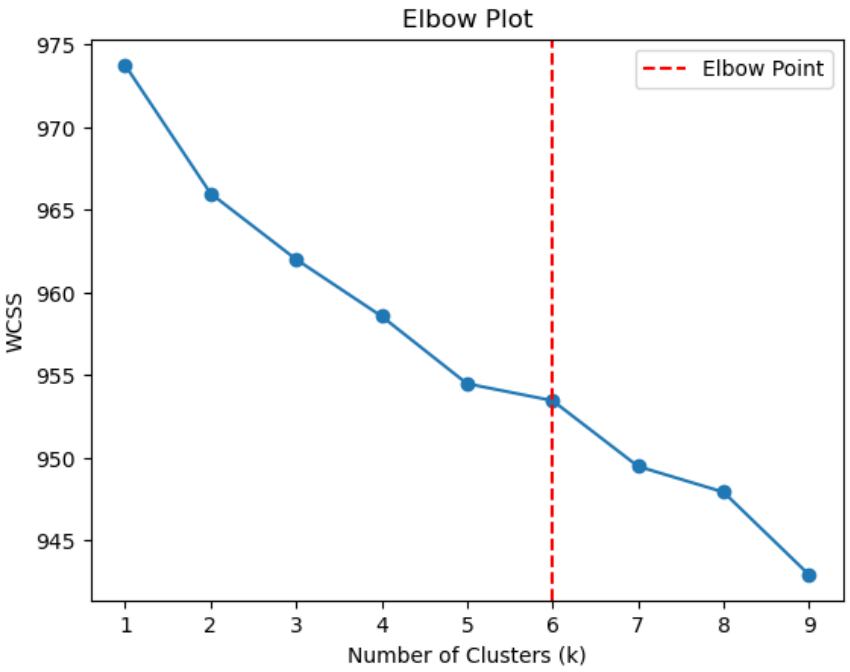

Figure 3: Plot for ENRON Dataset

The red dotted line in the plot signifies the optimal K (= 6) for clustering in the ENRON Dataset which took 389 seconds to run. It marks where the decrease in WCSS slows, indicating diminishing returns with additional clusters.