Artificial Intelligence
Clustering

Fall 2024
Parsa Darban 810100141
This file contains the report and results of the simulations conducted.

Abstract

In this project, we aim to analyze text from different news groups by using clustering algorithms. Our goal is to categorize the available data into various groups, ensuring that after applying the clustering algorithm, the data points are as accurately placed in their correct clusters as possible.

Clustering is a technique that involves grouping similar objects based on their inherent similarities. In other words, its goal is to divide data points into distinct clusters such that the points within a cluster are more similar to each other than to those in other clusters. By discovering these natural groupings, clustering algorithms can provide valuable insights into the underlying structure of the data. Clustering is used in various fields, including customer segmentation, image and document classification, anomaly detection, and recommendation systems.

The dataset used in this project consists of a collection of daily news articles in English from several news groups.

In machine learning, Unsupervised and Supervised algorithms are two main approaches. In Unsupervised learning, the model tries to identify hidden structures, patterns, and clusters in the data without using actual labels. Algorithms such as K-Means and DBSCAN fall under this category, which are used for clustering data. On the other hand, in Supervised learning, the model is trained using labeled data and aims to make accurate predictions for new data, such as in Regression and Classification algorithms. Comparing these two approaches shows that in unsupervised learning, the model is primarily designed to discover hidden structures and patterns in the data, while in supervised learning, the model focuses on prediction and classification tasks based on labeled data.

Preprocessing

Text data preprocessing is a crucial step in natural language processing that helps improve the quality of data and prepares it for further analysis. Text data typically contains noise such as stop words, punctuation, URLs, email addresses, and other irrelevant components, which can reduce the accuracy of machine learning models. Removing this noise allows the models to focus on the core, meaningful information and yields better results.

Additionally, preprocessing helps normalize the data by converting similar or diverse words into a standard form. Techniques like lemmatization or stemming reduce words with different forms (e.g., "running" and "ran") to a base form (e.g., "run"). This reduces data complexity and enhances the model's accuracy in analyzing the text's meaning. Overall, data preprocessing aids machine learning models in better simulating the underlying structure of the data and provides more accurate analysis results.

After loading the data and identifying its patterns and information, we move on to preprocessing it to clean the data, making it ready for feature extraction. First, we remove stop words like "the," "a," and "they," as these words are removed to reduce noise and focus on meaningful terms. Next, we remove any email addresses to prevent irrelevant artifacts in the text. Emails are usually not useful for semantic analysis. Then, we eliminate any URLs present in the text. We also remove numbers, punctuation, and special characters from the text. This ensures the text data is cleaner and more focused. Additionally, we remove newline (\n), carriage return (\r), and tab (\t) characters, as they are formatting artifacts with no semantic value.

Next, we proceed with tokenizing the text. The text is converted into individual words in lowercase using word_tokenize. Tokenization is essential for analyzing text at the word level. Afterward, we perform lemmatization on these words. This step reduces redundancy while preserving the semantic meaning.

Tokenization, lemmatization, and stemming are three essential techniques in natural language processing (NLP) used to process and analyze text. They serve different purposes and operate at different levels of text manipulation.

Tokenization is the process of breaking down text into smaller units, called tokens, typically words or phrases. This step is essential because it allows a machine to process text in manageable parts, making it easier to analyze and extract meaning. Tokenization helps in converting unstructured text into a structured form, ready for further analysis such as classification, sentiment analysis, or entity recognition.

Stemming involves reducing words to their root form by stripping suffixes. For example, the words "running," "runner," and "ran" would all be reduced to the root word "run." The goal of stemming is to remove derivational affixes so that words can be grouped together for analysis. However, stemming may result in words that are not real words or may lose some nuance, making it less precise than lemmatization.

Lemmatization, on the other hand, is a more advanced technique where words are reduced to their base or dictionary form, called the lemma. Unlike stemming, lemmatization takes into account the word’s meaning and context. For instance, "better" would be lemmatized to "good," and "running" would be lemmatized to "run." Lemmatization requires knowledge of the word's part of speech (noun, verb, etc.) and is more accurate but computationally expensive compared to stemming.

In summary, tokenization is the first step of dividing text into manageable pieces, followed by stemming or lemmatization to normalize those pieces. While stemming is faster but less accurate, lemmatization is more precise and context-aware, making it suitable for tasks that require higher accuracy, such as semantic analysis and machine learning.

Feature extraction

Feature extraction in natural language processing (NLP) is a crucial step because it helps to extract meaningful features from raw text data, which can then be used for machine learning models. Raw text data alone does not have inherent meaning for machine learning models and needs to be transformed into features that the models can analyze. Feature extraction allows important information to be extracted from the text and formatted in a numerical and structured way, making it ready for modeling algorithms.

Simply reading the text data is not enough because textual data is often complex and contains additional, unnecessary components that can reduce the accuracy of the model. For example, stop words, or noise such as URLs, punctuation, and emails, may lead to errors in text analysis. Feature extraction, by removing or transforming these irrelevant components, helps the model focus on the core and meaningful information, resulting in better prediction accuracy.

Moreover, feature extraction helps machine learning models analyze the text in a way that is understandable to them. Some features, such as word count, sentence length, or how specific words are used, can provide valuable insights into the semantic or emotional aspects of the text. Without feature extraction, models cannot effectively leverage the complexity present in text data and may produce incorrect results. Therefore, this process helps improve the performance of models and increases the accuracy of analysis.

Considering this, we need to select a model for feature extraction.

Sentence Transformer models are advanced tools in natural language processing (NLP) designed to generate meaningful and dense vector representations of sentences or text segments. Unlike traditional transformers, which focus on word-level representations, these models aim to understand the overall meaning of sentences. They are widely used for tasks such as semantic search, text clustering, and calculating similarity between sentences, where understanding semantic relationships between sentences is crucial. These models are trained using transformer-based pre-trained models like BERT or RoBERTa, combined with additional layers and loss functions to produce high-quality sentence embeddings.

One of the key advantages of Sentence Transformers is their ability to encode complete sentences into fixed-size dense vectors. These vectors can then be compared using simple mathematical metrics like cosine similarity or utilized in downstream applications like clustering or information retrieval. These models are designed to be scalable for large datasets, as they support batch processing and can efficiently process thousands of sentences while preserving their rich semantic meaning.

The all-MiniLM-L6-v2 model is a compact and optimized Sentence Transformer designed for speed and resource efficiency while delivering robust performance. This model is based on Microsoft's MiniLM architecture, which significantly reduces the number of parameters without sacrificing quality to a notable extent. It consists of six transformer layers (L6) and uses fewer hidden dimensions than larger models, making it faster for training and suitable for systems with limited computational resources, such as low-power CPUs or GPUs.

Despite its small size, all-MiniLM-L6-v2 offers excellent performance in understanding sentence meaning across various tasks, including sentence similarity, text clustering, and information retrieval. It is pre-trained and fine-tuned on diverse datasets to ensure strong general performance across

different domains. This versatility makes it a popular choice for developers and researchers seeking a balance between performance and computational efficiency.

The vectors produced by all-MiniLM-L6-v2 are dense with 384 dimensions, compact enough to enable scalable operations on large text datasets. These vectors are particularly suitable for applications like document clustering, where the model encodes textual data into numerical vectors that can be clustered based on semantic similarity. Additionally, the model's efficiency ensures fast inference times, enabling its use in real-time applications such as question-answering systems or recommendation engines. The model's ability to balance a lightweight architecture with strong semantic performance makes all-MiniLM-L6-v2 an excellent choice for NLP projects.

The use of feature vectors in text processing is crucial as they provide a structured, numerical representation of textual data that machine learning algorithms can understand and process. Raw text data is inherently unstructured and challenging for algorithms to analyze directly. Feature vectors capture the semantic and contextual information of the text in a compact, fixed-size numerical form, making it easier for clustering algorithms to identify patterns, relationships, and similarities between different pieces of text. These vectors serve as a bridge between the unstructured nature of language and the structured input required by machine learning models.

So, we extract the text feature vectors using this model and pass them to various clustering algorithms for grouping.

Dimension Reduction

PCA

Principal Component Analysis (PCA) is a technique used for dimensionality reduction, which simplifies a dataset by transforming it into a set of new variables known as principal components. These components are linear combinations of the original features, with each one capturing as much variance in the data as possible. PCA helps in reducing the complexity of high-dimensional datasets while maintaining the most significant patterns and relationships.

The process begins by standardizing the data, ensuring that each feature has a mean of 0 and a variance of 1. This is important when the dataset contains features with different scales. Next, a covariance matrix is computed to understand the relationships between the variables. PCA then calculates the eigenvalues and eigenvectors of this matrix, where the eigenvectors represent the new axes (principal components), and the eigenvalues indicate the variance captured by each component.

PCA transforms the original data by projecting it onto these new axes, reducing the number of dimensions while retaining the most important variance in the data. The number of components retained depends on the amount of variance one wants to preserve. By selecting only the first few principal components, PCA effectively reduces the number of features, leading to simpler models and improved performance in some cases.

While PCA offers several benefits, including noise reduction and improved visualization, it also has limitations. It assumes that the most meaningful patterns in the data correspond to directions with the highest variance, which may not always align with the underlying problem. Additionally, the transformed components can be difficult to interpret, and the method is sensitive to outliers, which can heavily influence the results. Despite these challenges, PCA remains a powerful tool in exploratory data analysis and machine learning preprocessing.

The PCA plot for clustering is generally similar to other data analysis plots, but its main goal is to assist in visualizing clustered data. In PCA, after reducing the dimensions of the data to two or three principal components, we can observe the data points in the new (lower-dimensional) space. This helps analysts easily identify the different clusters in the data, as the clusters may become more distinct and separable in this space.

However, it should be noted that PCA itself is not a clustering algorithm and is only used for dimensionality reduction and data visualization. After applying PCA, clustering algorithms like K-Means or DBSCAN are typically used to partition the data into different clusters. PCA plots that display the data after clustering usually differentiate the clusters with different colors to make their separation easier to observe.

t_SNE

t-SNE (t-Distributed Stochastic Neighbor Embedding) is a powerful dimensionality reduction technique that focuses on preserving the local structure of high-dimensional data, making it ideal for visualizing clusters in datasets. Unlike PCA, which preserves global structure, t-SNE works by calculating pairwise similarities between data points and using a probability distribution to group similar points. It then places the data points in a lower-dimensional space, typically 2D or 3D, and iteratively adjusts their positions to minimize the difference in pairwise similarities using gradient descent.

The technique employs a Student's t-distribution in the lower-dimensional space, which helps avoid crowding and better captures relationships between points. This distribution's heavier tails ensure that dissimilar points are placed farther apart, while similar points stay close together. t-SNE is particularly effective at revealing clusters or patterns in complex datasets, though it can be computationally expensive for large datasets due to its iterative optimization process.

Despite its strengths, t-SNE has limitations. It does not preserve global relationships, meaning that the distances between distant clusters may not be meaningful. Additionally, the results are not easily interpretable in terms of the original features, unlike PCA. For clustering plots, t-SNE is widely used as it highlights local structures, such as grouping similar data points together, allowing clear visualization of clusters or patterns that may not be easily identified in higher-dimensional spaces. However, careful parameter tuning is required for optimal results, and its use on large datasets should be approached cautiously due to its computational demands.

t-SNE (t-Distributed Stochastic Neighbor Embedding) and PCA (Principal Component Analysis) are both dimensionality reduction techniques used to visualize data in lower dimensions, but there are significant differences in how they work and their applications. PCA is a linear technique that reduces data to lower dimensions while generally preserving the global structure of the data. This means that PCA focuses on the overall variance in the data and maintains the global relationships between data points well. On the other hand, t-SNE is a non-linear technique that specifically focuses on preserving the local structure of the data, maintaining pairwise similarities between data points in a lower-dimensional space. As a result, t-SNE is more effective for visualizing clusters and local structures in the data, whereas PCA might not capture these local structures as well.

Another key difference is that PCA is easier to interpret, with the principal components that reduce the data being directly related to the variance in the data. This means that PCA can reveal the key features of the data more transparently. However, t-SNE generally does not require direct interpretation and is used more for visualization since its results are less tied to the original features of the data and focus more on preserving local structure. Additionally, t-SNE is more computationally expensive and time-consuming than PCA due to its iterative optimization process, especially when working with high-dimensional and large datasets.

Evaluation

Homogeneity and Silhouette are two different metrics for evaluating clustering quality, each working in distinct ways.

Homogeneity is a metric used to evaluate the quality of clustering by examining whether the data points within each cluster are more similar to each other. This metric requires information from the true labels of the data and evaluates clustering quality by comparing the similarity of data within a cluster and between different clusters. If all samples in a cluster belong to the same category, Homogeneity will be equal to 1. If the data in a cluster belong to different categories, the value of Homogeneity will decrease.

Silhouette is another metric used to evaluate the quality of clustering, focusing on how well each data point fits into its own cluster and how distinct it is from other clusters. This metric consists of two main parts: the distance between a data point and its own cluster (cohesion) and the distance between the data point and the nearest different cluster (separation). Silhouette assigns a value between -1 and 1 to each data point. A value of 1 indicates good clustering, where the data point fits well into its cluster and is distinct from other clusters. Values close to 0 or negative indicate problems with clustering and misallocated data. This metric can be computed without the need for true labels.

In comparison, Homogeneity requires the true labels of the data and focuses more on the similarity and uniformity of samples within clusters. On the other hand, Silhouette provides a more general evaluation of clustering quality, assessing both the cohesion within clusters and the separation between clusters. Silhouette is generally a more balanced and suitable metric for unlabeled data, while Homogeneity is more applicable for labeled data.

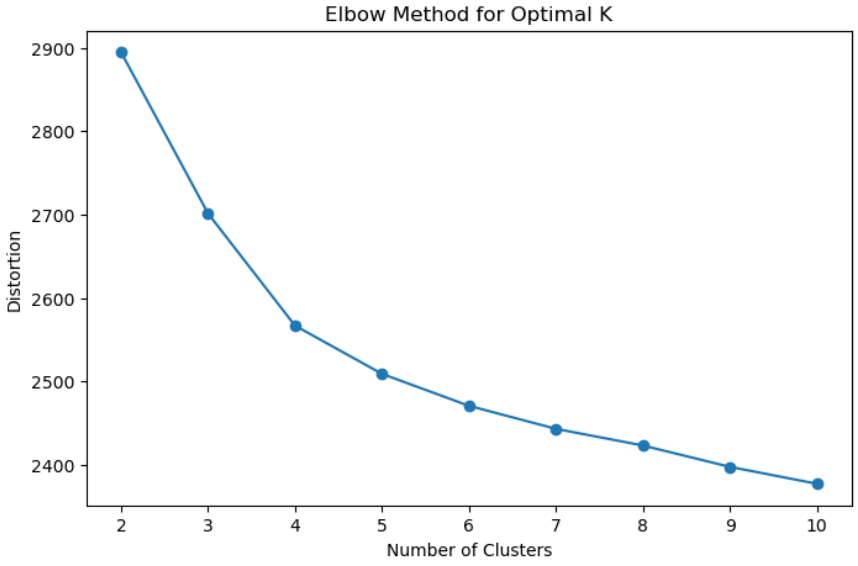
Clustering Algorithms

K_Means

K-Means is a popular unsupervised machine learning algorithm used for clustering data into distinct groups. It aims to partition a dataset into k clusters, where each cluster contains points that are more similar to each other than to points in other clusters. The algorithm is iterative and works by minimizing the variance (or inertia) within each cluster.

K-Means clustering is an iterative algorithm that partitions data into k clusters by repeatedly assigning points to the nearest cluster centroid and updating the centroids as the average of the assigned points. This process continues until the centroids stabilize or a maximum number of iterations is reached, indicating convergence. However, K-Means might fail to converge if the algorithm oscillates between cluster assignments due to poor initialization or non-convex cluster structures. To mitigate this, methods like K-Means++ are used for better centroid initialization, and a convergence criterion (like maximum iterations or minimal centroid movement) is typically applied.

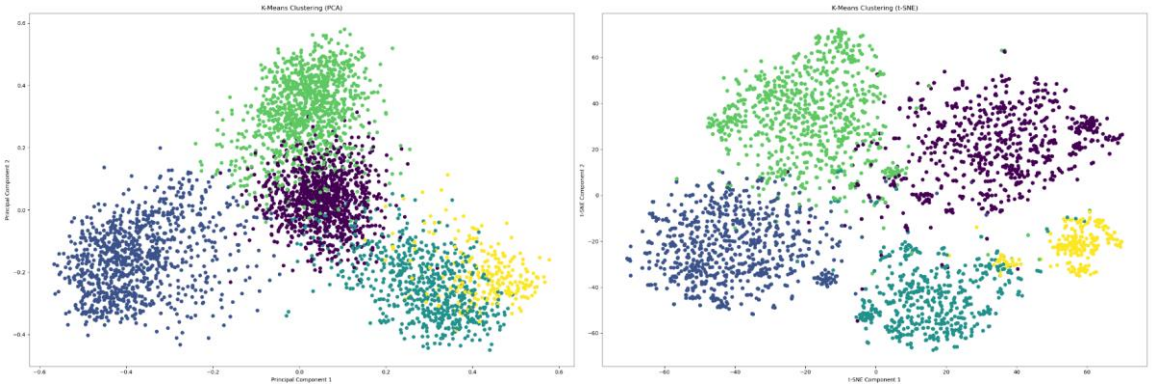
The elbow method in K-Means clustering is used to determine the optimal number of clusters, k, by plotting the within-cluster sum of squares (or inertia) against different values of k. As k increases, inertia decreases because the clusters become smaller and more compact. The idea is to identify the "elbow" point in the plot, where the rate of decrease in inertia slows down significantly. This point represents the optimal number of clusters, as adding more clusters beyond this point does not provide substantial improvements in the model's performance, indicating that further partitioning of the data is unnecessary.



With the change in slope at point k = 4, we consider the number of clusters to be 4. The result is as follows:

```
Enter the optimal number of clusters based on the elbow method: 4
Silhouette Score for K-Means: 0.1076
```

The results of PCA and t-SNE are also as follows:


K-Means Clustering (PCA)
Principal Component 1
Principal Component 2
K-Means Clustering (t-SNE)
t-SNE Component 1
t-SNE Component 2

DBSACN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a powerful clustering algorithm that identifies clusters of varying shapes and sizes by grouping together points that are closely packed. Unlike methods like K-Means, which require the number of clusters to be predefined, DBSCAN determines clusters based on the density of points in the data. It uses two key parameters: eps, which defines the radius of a neighborhood around a point, and min_samples, which specifies the minimum number of points required to form a dense region.

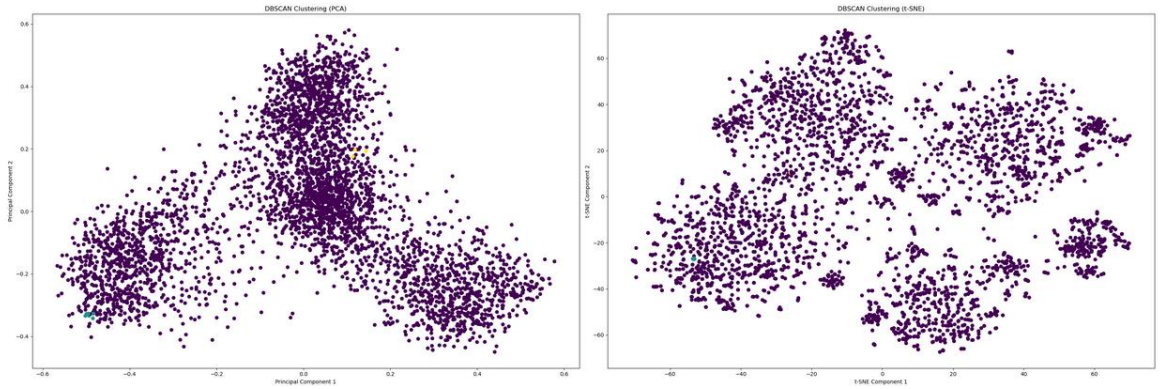
The algorithm starts by randomly selecting a data point. If the point has at least min_samples neighbors within the radius eps, it is classified as a "core point," and a cluster is initiated. All neighboring points are recursively checked to expand the cluster. Points that do not meet the density criteria but lie within the radius of a core point are labeled as "border points" and assigned to the cluster. Points that do not belong to any cluster are considered "noise." This approach makes DBSCAN effective for detecting irregularly shaped clusters and outliers.

DBSCAN is advantageous because it can identify clusters of arbitrary shapes and is robust to noise and outliers. It does not require prior knowledge of the number of clusters, which makes it particularly useful for exploratory data analysis. However, it has limitations, such as sensitivity to the choice of eps and min_samples. Poor parameter selection can lead to either over-clustering or failing to find meaningful structures. Additionally, DBSCAN struggles with datasets of varying densities and high-dimensional data where defining a meaningful neighborhood radius is challenging.

The result is as follows:

```
Silhouette Score for DBSCAN (excluding noise): 0.8767
```

The results of PCA and t-SNE are also as follows:



Hierarchical

Hierarchical clustering is a versatile and intuitive method for clustering data by creating a hierarchy of clusters. It comes in two main types: agglomerative and divisive. Agglomerative clustering, the more common approach, starts with each data point as its own cluster and successively merges the most similar clusters until all points are grouped into a single cluster or a desired number of clusters is reached. Conversely, divisive clustering begins with all points in one cluster and recursively splits them into smaller clusters. The method relies on a distance metric (like Euclidean or Manhattan) and a linkage criterion (like single, complete, or average linkage) to determine how clusters are formed.

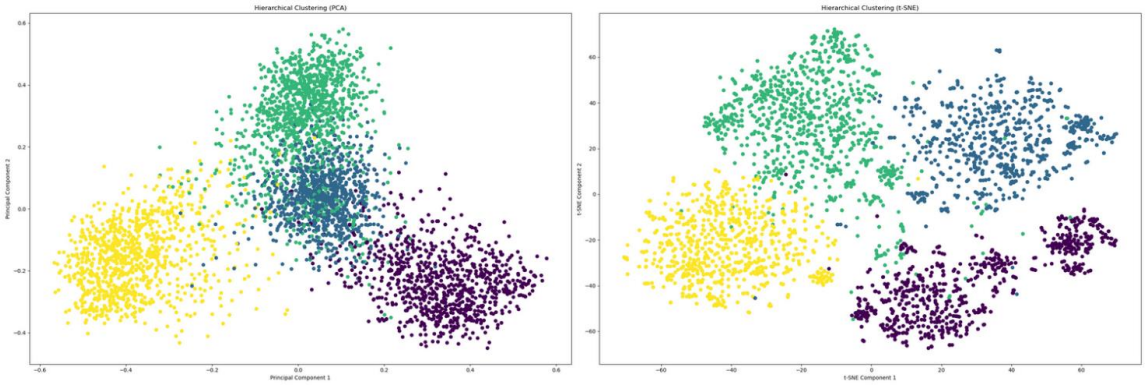
A significant advantage of hierarchical clustering is its ability to create a dendrogram, a tree-like structure that visually represents the process of cluster merging or splitting. This provides insights into the data's hierarchical structure and allows users to decide the number of clusters by "cutting" the dendrogram at an appropriate level. Unlike algorithms such as K-Means, hierarchical clustering does not require specifying the number of clusters in advance. It is especially useful in exploratory data analysis for datasets with small to medium sizes or for understanding nested relationships, such as in biology (e.g., phylogenetic trees) and text analysis.

Despite its strengths, hierarchical clustering has limitations. It is computationally expensive, with a time complexity of $O(n^2)$ or higher, making it unsuitable for large datasets without optimization techniques. Additionally, hierarchical clustering is sensitive to noise and outliers, which can significantly affect the resulting dendrogram. Once clusters are merged or split, the process cannot be reversed, potentially leading to suboptimal results. Despite these drawbacks, hierarchical clustering remains a powerful tool when the dataset's structure and the relationships among data points are of primary interest.

The result is as follows:

```
Silhouette Score for Hierarchical Clustering: 0.1039
```

The results of PCA and t-SNE are also as follows:



DBSCAN is the best algorithm for this dataset because it effectively handles noise and captures the local density structure, resulting in more meaningful and well-separated clusters. K-Means and Hierarchical methods may struggle with datasets that have irregular cluster shapes, noise, or density variations.