Amirkabir University of Technology

(Tehran Polytechnic)

Clustering Analysis Using Non-Negative Matrix Factorization

Zohreh Sarmeli Saeedi

April 2025

Abstract

This report presents a clustering analysis of the 20 Newsgroups dataset, focusing on five news categories. After preprocessing the data by removing stop words, punctuation, and numbers, a Term-Document matrix was constructed using the TF-IDF algorithm. Document clustering was performed using Non-Negative Matrix Factorization (NMF), custom K-Means, and Scikit-learn's K-Means. The performance of these algorithms was evaluated using Adjusted Rand Index (ARI), Adjusted Mutual Information (AMI), and Silhouette Score. Results indicate that Scikit-learn's K-Means outperformed others in ARI and AMI, while NMF showed moderate performance in intra-cluster cohesion.

1 Introduction

The 20 Newsgroups dataset, sourced from Scikit-learn (7), was used for this experiment, comprising five news categories: comp.graphics, talk.politics.guns, alt.atheism, sci.med, and sci.space. Each category contains a distinct set of textual documents, assumed to have unique textual features enabling clustering. The raw dataset was loaded without initial preprocessing.

Recent advancements in text clustering emphasize the importance of robust preprocessing and feature extraction to improve clustering performance (1). This study builds on such methodologies, applying TF-IDF for feature extraction and comparing clustering algorithms.

2 Data Preprocessing

Data preprocessing was performed using the NLTK library (2), involving the following steps:

1. Converting text to lowercase for uniformity.
2. Removing numbers using regular expressions.
3. Eliminating punctuation using a mapping table.
4. Tokenizing text into individual words.
5. Removing English stop words to reduce noise.

It was hypothesized that stop words and punctuation negatively impact clustering quality. The output was clean, tokenized text ready for constructing the Term-Document matrix.

3 TF-IDF Methodology

The Term Frequency-Inverse Document Frequency (TF-IDF) method measures the importance of a term t in a document d relative to the entire corpus (6). It comprises two components:

- **Term Frequency (TF)**: The frequency of a term in a document, normalized by the document's length.
- **Inverse Document Frequency (IDF)**: Measures a term's rarity across the corpus, calculated as:

$$\text{IDF}(t) = \log\left(\frac{N}{\text{df}(t)}\right)$$

  where N is the total number of documents, and $\text{df}(t)$ is the number of documents containing term t.

Terms frequent in a document but rare across the corpus receive higher weights, enhancing clustering relevance.

4 Term-Document Matrix Construction

Using Scikit-learn's `TfidfVectorizer`, a Term-Document matrix was constructed. The TF-IDF algorithm weighted terms based on their importance within documents and the corpus. The resulting matrix dimensions reflect the number of documents and unique words after preprocessing. It was assumed that high-frequency terms in a document with low corpus frequency provide more clustering information.

5 Non-Negative Matrix Factorization (NMF)

NMF from Scikit-learn was applied with five clusters (4). The TF-IDF matrix was decomposed into two matrices: W (document-to-cluster weights) and H (term-to-cluster weights). Documents were assigned to the cluster with the highest weight in W. NMF showed moderate performance in intra-cluster cohesion (Silhouette Score) but was outperformed by Scikit-learn's K-Means in ARI and AMI metrics.

6 Custom K-Means Implementation

A custom K-Means algorithm was implemented with the following steps:

- Random selection of initial cluster centroids.
- Computing Euclidean distances from documents to centroids.
- Assigning documents to the nearest cluster.
- Updating centroids until convergence.

A maximum of 100 iterations and a tolerance of 10^{-4} were assumed sufficient for convergence. Due to random centroid initialization, results varied across runs. This method performed the worst across all metrics (5).

7 Scikit-learn K-Means

Scikit-learn's K-Means was applied with five clusters, incorporating optimizations for centroid initialization and distance calculations (7). It was hypothesized that these optimizations would yield better performance than the custom implementation. The results confirmed superior performance in ARI and AMI metrics.

8 Evaluation Metrics

Three metrics were used to evaluate clustering performance:

- **Adjusted Rand Index (ARI)**: Measures similarity between predicted and true labels, adjusted for chance (3). Values range from -1 to 1, with 1 indicating perfect agreement and 0 indicating random labeling.
- **Adjusted Mutual Information (AMI)**: Measures statistical dependency between two clusterings, adjusted for chance (9). Values range from 0 to 1, with 1 indicating perfect agreement.
- **Silhouette Score**: Evaluates intra-cluster cohesion and inter-cluster separation (8). Higher values indicate better-defined clusters.

9 Results

The performance metrics are summarized in the following table:

Table 1: Clustering Performance Metrics

Method	ARI	AMI	Silhouette
NMF	0.4017	0.5566	0.006952
Custom K-Means	0.3355	0.4632	0.006565
Scikit-learn K-Means	0.4833	0.6048	0.006843

9.1 Adjusted Rand Index (ARI)

Scikit-learn K-Means achieved the highest ARI (0.4833), followed by NMF (0.4017) and Custom K-Means (0.3355).

9.2 Adjusted Mutual Information (AMI)

Scikit-learn K-Means led with an AMI of 0.6048, followed by NMF (0.5566) and Custom K-Means (0.4632).

9.3 Silhouette Score

All methods yielded low Silhouette Scores, indicating poor cluster separation in the TF-IDF feature space. NMF slightly outperformed others (0.006952), followed by Scikit-learn K-Means (0.006843) and Custom K-Means (0.006565).

Figure 1: Placeholder for clustering results visualization. Please upload the relevant image.

10 Conclusion

Scikit-learn's K-Means demonstrated superior performance in ARI and AMI, benefiting from optimized centroid initialization and distance computations. NMF performed slightly better in intra-cluster cohesion (Silhouette Score) but was moderate in ARI and AMI. Custom K-Means performed the worst, likely due to unoptimized centroid selection. The primary challenge was achieving clear cluster separation in the TF-IDF feature space, suggesting potential improvements with advanced feature engineering (1).

- **Best Method**: Scikit-learn K-Means (based on ARI and AMI).
- **Main Challenge**: Poor cluster separation in TF-IDF space.
- **Moderate Performance**: NMF showed slightly better intra-cluster cohesion.
- **Relative Weakness**: Custom K-Means underperformed across all metrics.

References

[1] Aggarwal, C. C. (2020). *Machine Learning for Text*. Springer.

[2] Bird, S., Klein, E., Loper, E. (2009). *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media.

[3] Hubert, L., Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2(1), 193–218.

[4] Lee, D. D., Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755), 788–791.

[5] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 281–297.

[6] Manning, C. D., Raghavan, P., Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

[7] Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

[8] Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20, 53–65.

[9] Vinh, N. X., Epps, J., Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11, 2837–2854.

Appendix

The complete project implementation, along with related documentation, is available in the GitHub repository below:

`https://github.com/Zohreh004/Implementation-Project-3-`