Clustering Subreddit Texts

Udrea Iulia-Maria

January 25, 2025

1 Introduction

This paper aims to compare and evaluate three clustering methods: K-Medoids, Gaussian Mixture Models (GMM), and K-Means, using the Religion dataset. The primary goal is to analyze the performance of these clustering methods based on the textual representations used, including TF-IDF and Word2Vec, and to assess the quality of the resulting clusters against the dataset's ground-truth labels.

2 Dataset

The dataset consists of 35.000 texts from various topics on Reddit. Most of the data is extracted from forums focused on religion (such as Islam, Hinduism, and Christianity), but it also includes other categories such as Sport or Politics. For the clustering task, I selected distinct topics, keeping 500 examples from four categories: Hinduism, Travel, Fitness, and Food.

2.1 Preprocessing

For preprocessing, I removed links, replaced emojis with their textual descriptions, eliminated stopwords, digits, and punctuation marks, and tokenized the text into individual words. An example is presented below:

Initial Text:	Just booked my tickets to Tokyo for December 5! More info at: http://example.com
Preprocessed Text:	['booked', 'tickets', 'tokyo', 'december', 'info']

Table 1: Example of text preprocessing

The texts were split into training data (80%) and testing data (20%). I chose two distinct representations: TF-IDF and Word2Vec.

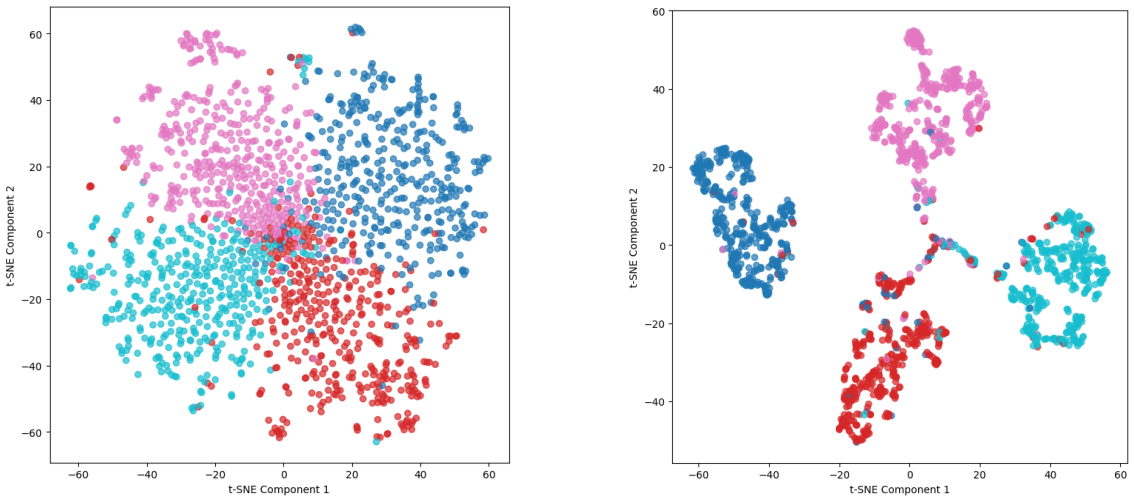
2.2 Representations

2.2.1 TF-IDF

TF-IDF is an improvement over simpler representations like Bag-of-Words, as it considers the importance of words based on their frequency across the entire dataset. However, the number of features in the TF-IDF representation equals the number of unique words in the training data, which can be very high (in this case, there are about 17.000 words).

This necessitates dimensionality reduction. Initially, I applied PCA, retaining 150 components. However, the clustering methods yielded poor results, with Silhouette Scores close or less than 0. As shown in Figure 1a, when these representations were visualized using t-SNE, texts with the same label were well grouped, but there was no space between clusters. Thus, while classes were well separated, clear clusters did not form.

To address this, I opted for dimensionality reduction using UMAP, which provides a non-linear projection that preserves both local and global structures in the data. In Figure 1b, we can observe that applying UMAP to the TF-IDF vectors resulted in much better separation, even aligning closely with the ground-truth labels.



(a) PCA reduction

(b) UMAP reduction

Figure 1: t-SNE Visualisation of Reduced Vectors

2.2.2 Word2Vec

The other representation method is Word2Vec, which represents each word as a vector in such a way that words with similar meanings are located closer in the vector space. To achieve this, I trained a Word2Vec model using the training data, generating vectors with a dimensionality of 150. Here, dimensionality reduction was not necessary, and data visualization using t-SNE demonstrated good separation, as shown in Figure 2.

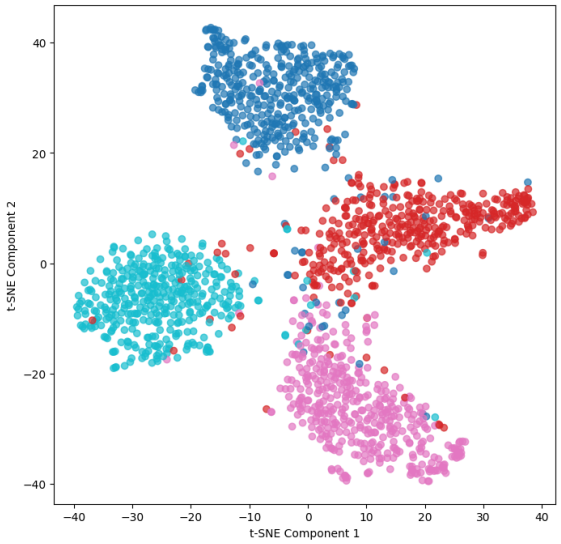


Figure 2: t-SNE Visualization of Word2Vec Representations

2.3 K-Medoids

The first clustering method used is K-Medoids, which partitions the dataset into clusters by minimizing the sum of distances between data points and their respective medoids. Unlike K-Means, K-Medoids selects actual data points as cluster centers, making it more robust to outliers. The method requires two main parameters: **n_clusters**, the number of clusters, and **metric**, which defines the distance metric used to calculate dissimilarities.

In the grid search, I tested values ranging from 2 to 9 for n_clusters and experimented with three distance metrics: Euclidean, Manhattan, and Cosine. The optimal parameters were determined based on the Silhouette Score, which measures the quality of clustering by evaluating how similar each point is to its own cluster compared to other clusters. Higher scores indicate better-defined clusters. Figure 3 displays the Silhouette Scores obtained using both text representations.

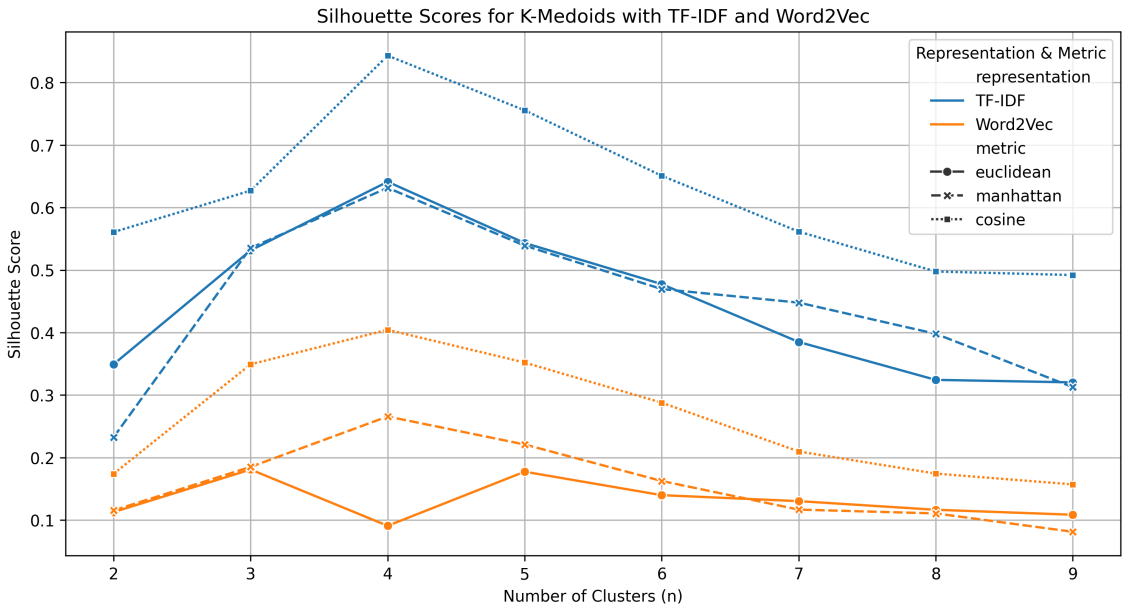
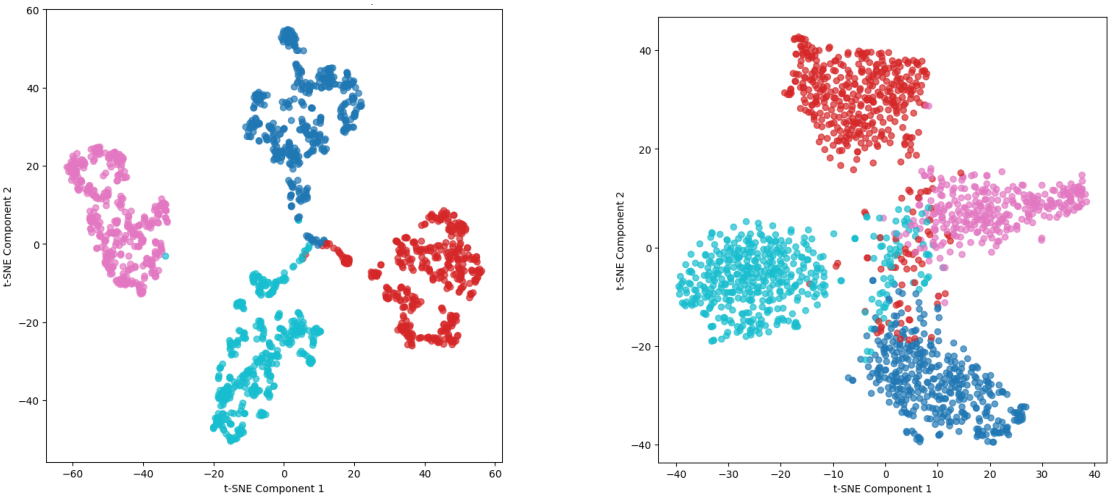


Figure 3: K-Medoids Silhouette Scores during Grid Search

From the graph, we observe that the best scores for both representations were achieved using **4 clusters** and the **cosine distance** metric. We observe that the maximum score obtained for TF-IDF (0.8431) is more than twice as high as the score for Word2Vec (0.4044), which means that the clusters formed using TF-IDF are more cohesive and better separated from each other.

Furthermore, for any number of clusters, the score obtained with cosine distance is consistently higher than the scores with Euclidean or Manhattan distances. This could be because cosine distance focuses on the angular similarity between vectors rather than their magnitudes, making it suitable for high-dimensional spaces and textual data, where directionality often captures more meaningful relationships than absolute distances. A visualization of these clusters can be seen in Figures 4, where each color represents a cluster.



(a) TF-IDF with UMAP

(b) Word2Vec

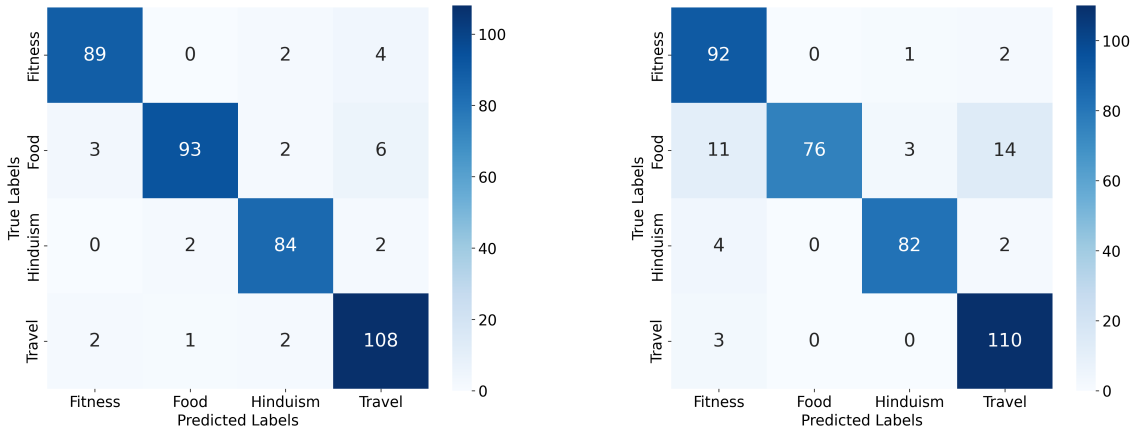
Figure 4: Visualization of K-Medoids Clusters for each Representation

We do not yet know if these clusters correspond to the actual categories of the data. Therefore, to verify this, we assign each cluster the label (ground truth) that is most frequent within it. This process is applied to both the training and testing data after generating the clusters, in order to compute the accuracy provided by the clustering. We compare this accuracy with the result obtained by running a simple classification model, K-Nearest Neighbors (KNN), with default parameters. The results are summarized in Table 2.

Representation	Silhouette Score	Accuracy Train (%)	Accuracy Test (%)	Accuracy KNN (%)
TF-IDF with UMAP	0.843	0.930	0.935	0.945
Word2Vec	0.404	0.896	0.900	0.950

Table 2: Evaluation Metrics for K-Medoids

Despite the significant difference in Silhouette Scores, the overall accuracy is good for both representations, on both training and testing data. The results are slightly lower but comparable to the performance achieved with KNN and significantly higher than random chance (25%). In Figures 5a and 5b, we can see the confusion matrices for the testing data, where clusters are assigned labels based on their most frequent ground-truth category.



(a) TF-IDF with UMAP

(b) Word2Vec

Figure 5: Confusion Matrices for K-Medoids Clustering

We observe that in this case, the representations achieve similar results, despite TF-IDF not capturing the same amount of information as Word2Vec. Dimensionality reduction with UMAP was essential, as other methods like PCA failed to deliver the desired outcome, with scores decreasing as the number of clusters increased.

3 Gaussian Mixture Model

The second clustering method used is Gaussian Mixture Model (GMM), which assumes that the data is generated from a mixture of Gaussian distributions, where each cluster corresponds to one Gaussian component. This method models both the mean and covariance of the data, allowing for clusters with varying shapes and orientations.

The method uses two parameters: the number of clusters, **n_components**, and the structure of the covariance matrices, **covariance_type**. In the grid search, we tested values ranging from 2 to 9 for the number of clusters and used the following options for covariance matrices: **full**, **diag**, **spherical**. Figure 6 presents the Silhouette Scores for each combination of hyperparameters.

In this case as well, the best hyperparameters were obtained with 4 components, using the diagonal covariance matrix. There is, again, a significant difference—nearly threefold—between the Silhouette Scores of the two textual representations, and it is noteworthy that these scores are lower compared to those achieved with the K-Medoids method. This may be because Gaussian Mixture Models assume a probabilistic structure for the data, which may not align as well with the high-dimensional, sparse nature of textual data as the distance-based approach of K-Medoids.

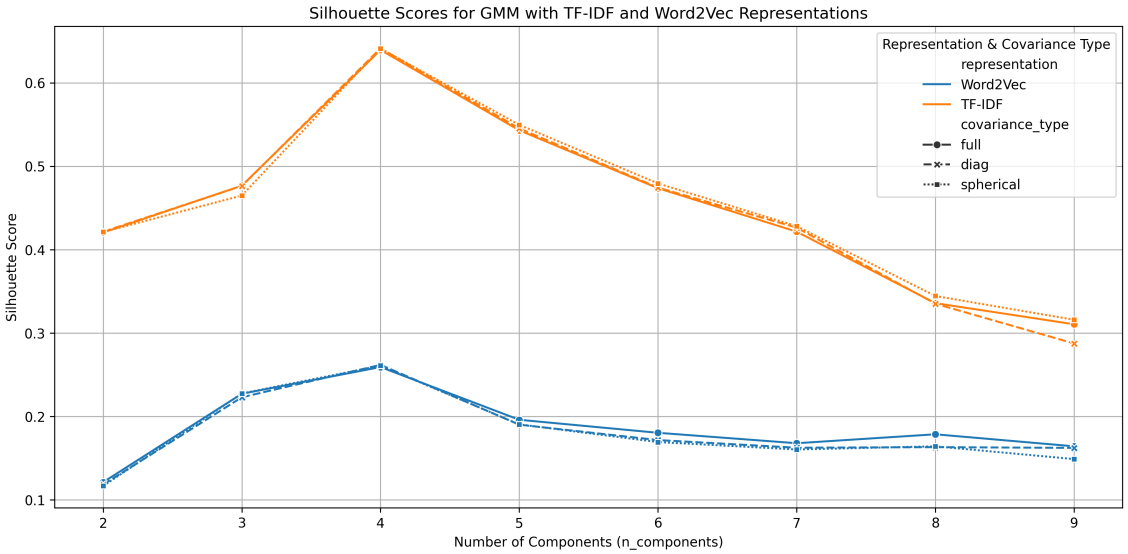
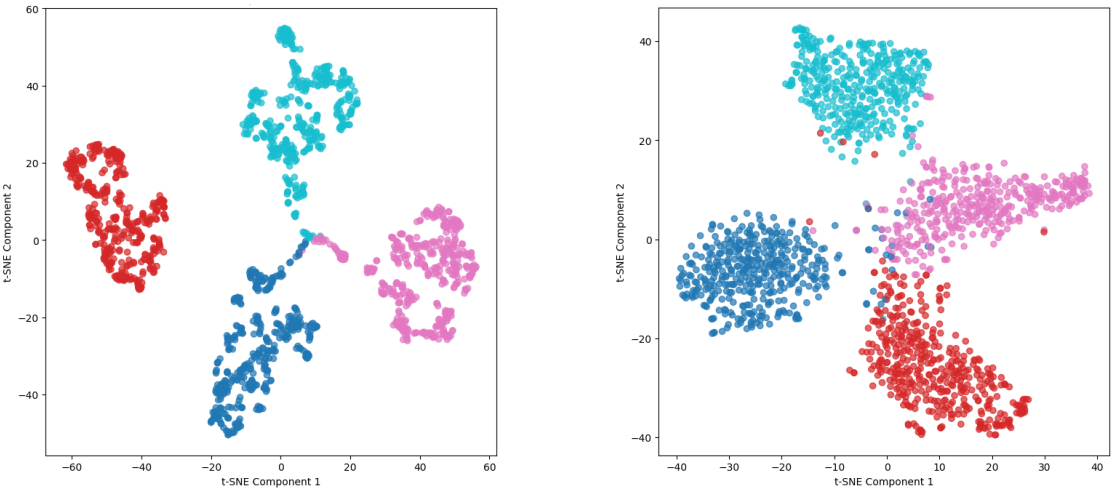


Figure 6: GMM Silhouette Scores during Grid Search

Unlike the previous method, there are no large differences between the covariance matrix types, which suggests that the shape and orientation of the clusters are not strongly influenced by this parameter in this dataset. This uniformity might indicate that the clusters are relatively simple in structure and do not require complex covariance models to capture their variance. A visualization of these clusters can be seen in Figures 7, where each color represents a cluster.



(a) TF-IDF with UMAP

(b) Word2Vec

Figure 7: Visualization of GMM Clusters for each Representation

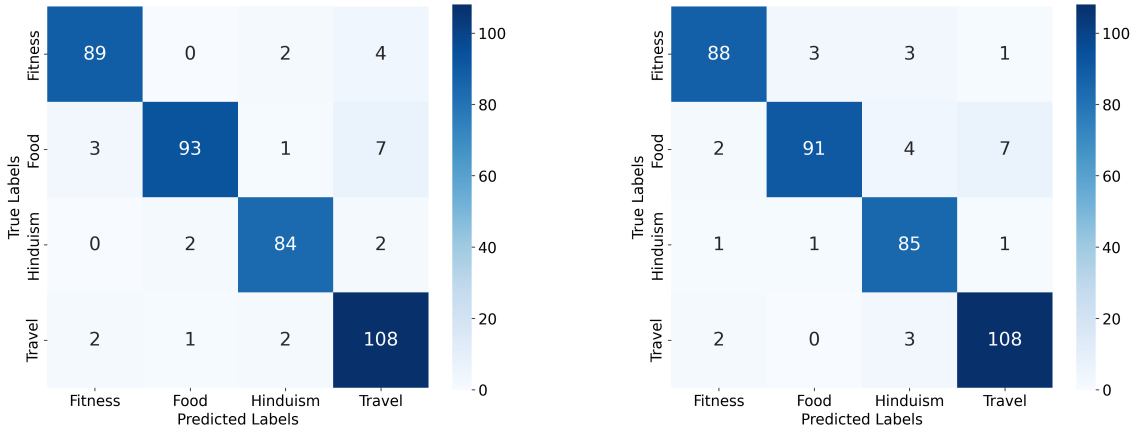
As in the previous case, we calculate the accuracy for the fixed set of hyperparameters as the predominant label within each cluster, for both the training and testing data. The high accuracy once again demonstrates that these clusters represent a good distribution of the labels. The results are summarized in Table 3:

Representation	Silhouette Score	Accuracy Train (%)	Accuracy Test (%)	Accuracy KNN (%)
TF-IDF with UMAP	0.641	0.931	0.935	0.945
Word2Vec	0.261	0.943	0.930	0.950

Table 3: Evaluation Metrics for GMM

We observe once again that, despite the significant differences in Silhouette Scores, the accuracies are similar across all data and closely match the performance of the KNN classification model. Both accuracies exceed the threshold of random chance (25%).

Compared to the previous method, the results for this method are slightly better, even though GMM is generally considered a more sophisticated method and despite the much lower Silhouette Scores. In Figure 8, we can see the confusion matrices for the testing data, where clusters are assigned labels based on their most frequent ground-truth category.



(a) TF-IDF with UMAP

(b) Word2Vec

Figure 8: Confusion Matrices for GMM Clustering

4 K-Means

The simplest clustering method is K-Means, which differs from K-Medoids in that it selects centroids as the mean position of all points within a cluster, rather than choosing actual data points as cluster centers. This approach makes K-Means more sensitive to outliers compared to K-Medoids.

K-Means has a single essential hyperparameter: the number of clusters (k). I included this method in the analysis to provide a baseline for comparison with more sophisticated clustering approaches and to evaluate its performance on the given textual data representations. The Silhouette Scores obtained during grid search are presented in Figure 9.

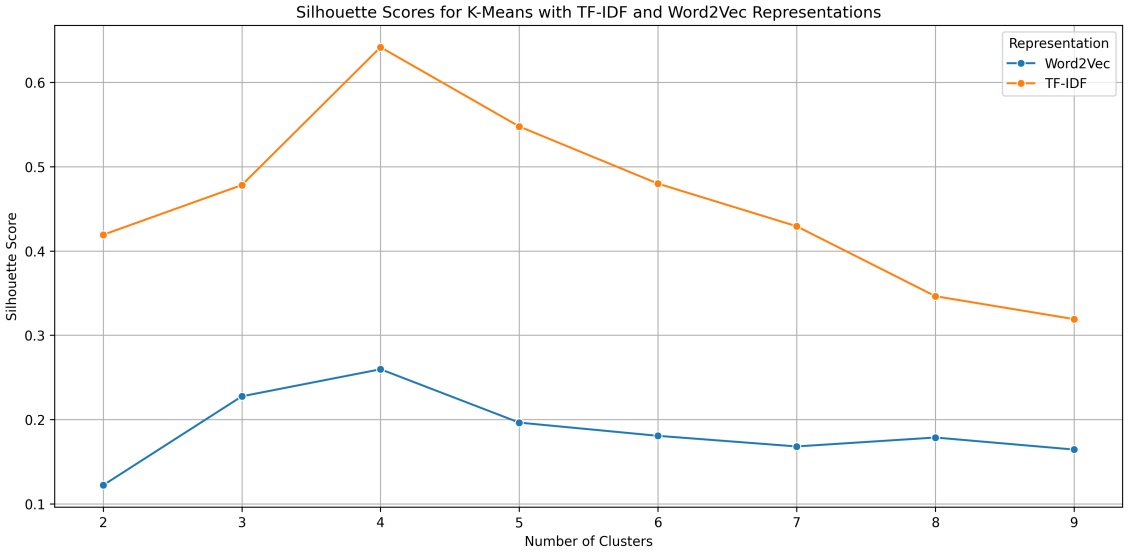


Figure 9: K-Means Silhouette Scores during Grid Search

The peaks are observed for 4 clusters in both representations. Compared to K-Medoids, the scores are lower but approximately equal to those achieved with GMM. Table 4 summarizes the results obtained. Despite the lower Silhouette Score, the accuracy is slightly higher than that obtained with K-Medoids but below GMM. This demonstrates that even simple methods can perform well with appropriate representations, sometimes even surpassing more complex models.

Representation	Silhouette Score	Accuracy Train (%)	Accuracy Test (%)	Accuracy KNN (%)
TF-IDF with UMAP	0.642	0.927	0.935	0.945
Word2Vec	0.259	0.920	0.922	0.950

Table 4: Evaluation Metrics for K-Means

5 Conclusions

The results of this analysis highlight several findings regarding text clustering and the impact of representations and methods. The choice of representation plays a critical role in clustering performance. While TF-IDF captures word importance, it required dimensionality reduction with UMAP to achieve meaningful clusters. Without it, the clustering results for TF-IDF were less satisfactory. In contrast, Word2Vec effectively represents contextual relationships between words, offering a robust foundation for clustering.

A key observation is the disparity between Silhouette Scores and clustering accuracy. While Silhouette Scores were lower for methods like GMM, the accuracy of clustering was still competitive. Future work could explore more advanced representations or evaluate other clustering algorithms to further refine the results.