Analysis of Large Language Model Embeddings for Text Clustering

Nayeeb Rashid
nrashid@umd.edu

Oishy Saha
osaha@umd.edu

Barproda Halder
bhalder@umd.edu

1 Problem statement

With the rapid expansion of big data technologies, there has been a surge in the amount of textual information generated across various domains such as business, education, science, communication, and social media. This text data appears in different forms, such as reports, articles, online reviews, and user feedback, but a significant portion of it is unstructured and requires advanced processing techniques to extract useful patterns and information.

Text clustering serves as a foundational step in natural language processing (NLP) that involves dividing the corpus of textual data into coherent groups or clusters, such that documents within the same cluster show higher similarity to one another than to those in different clusters. The groupings are based on the semantic similarity that does not require any predefined labels. The clustering process intends to uncover the hidden topics, relationships, or trends without the need for labeled data. It spans a wide spectrum of applications ranging from information retrieval (Decherchi et al., 2009), healthcare (Torabizadeh et al., 2025), customer feedback analysis (Jardim and Mora, 2022), and education (Katz et al., 2023).

The text clustering process is grounded on the principle that text documents can be mathematically represented by transforming them into vectors, e.g., term frequency-inverse document frequency (TF-IDF) and bag-of-words (BOW), and the vectors are converted into a high-dimensional embedding space, where each dimension corresponds to a feature extracted from the documents, such as word frequency, semantic context, or syntactic patterns. By leveraging these vector representations, clustering algorithms employ similarity or distance measures (e.g., cosine similarity, Euclidean distance) to group documents that exhibit close relationships in the feature space. This approach enables the discovery of natural groupings within large text corpora, thereby facilitating efficient organization, topic discovery, and deeper insight extraction from unstructured textual data.

These representations are very high-dimensional, creating extremely sparse matrices for a small text corpus, resulting in reduced discriminative power of the distance matrix, overfitting of the model, and high computational and memory costs (Aggarwal and Zhai, 2012). Additionally, traditional clustering approaches heavily rely on feature engineering and often lack semantic understanding. In this regard, LLM-based text clustering offers a promising solution by converting the texts, documents, and paragraphs into low-dimensional dense vector embeddings that encode significant semantic meaning (Reimers and Gurevych, 2019). Furthermore, the LLM-based clustering approach overcomes the heavy preprocessing of traditional clustering by automatic feature extraction through learned token embeddings and attention mechanisms of transformers.

In this work, we investigate the utility of large language model (LLM) embedding spaces through the downstream task of text clustering. Using six publicly available datasets, we systematically evaluate how clustering performance varies across embeddings produced by multiple state-of-the-art open-source LLMs and across datasets with different semantic characteristics. In order to evaluate the interaction between algorithmic assumptions and the structure of LLM embedding spaces, we also compare a variety of traditional clustering techniques (such as centroid- and hierarchical-based approaches) with more sophisticated clustering algorithms. We examine embeddings taken from several layers of six open-source LLMs to investigate representational features throughout network depth, measuring the evolution of clustering quality with model depth. Lastly, we examine how popular embedding post-processing methods (such as principal-component removal, dimensionality reduction, normalization, and demeaning) affect clustering results, offering insight into how these changes alter the embedding space and impact cluster separability.

2 Proposed vs. accomplished

The core objective of the project, as outlined in the proposal, is maintained without modification.

- ~~Analyze a larger set of open-sourced LLMs (Encoder-only, Decoder-only, and Encoder-Decoder) for the text clustering task~~
- ~~Analyze embeddings from different hidden states to understand the relation between clustering and model depth~~
- ~~Compare a wide range of clustering algorithms over a wide range of datasets~~
- ~~Investigate the impact of post-processing techniques on the extracted embeddings~~

3 Related work

Text Embeddings: Traditionally, methods such as Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) have been employed to learn dense vector representations of words by capturing their contextual relationships within text. The introduction of BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) marked a new era in embedding techniques, utilizing a bidirectional transformer pre-trained on large corpora to produce contextual embeddings that capture richer semantic relationships for tasks such as text clustering. Today, LLMs like OpenAI's GPT (Brown et al., 2020) lead embedding generation, capturing rich linguistic and conceptual knowledge through large-scale training for deeper contextual understanding.

Text Clustering Algorithms: Beyond classical methods like k-means, AHC, Spectral, and Fuzzy c-means, recent approaches leverage modern embeddings to better capture the unique characteristics of textual data. These methods often employ deep learning models, especially autoencoders, to learn meaningful low-dimensional representations suited for clustering (Berahmand et al., 2022). Ensemble clustering methods that combine multiple algorithms to enhance performance have also gained attention (Strehl and Ghosh, 2002).

Recent studies have shown the growing impact of LLMs and transformer-based embeddings on text clustering. Pugachev and Burtsev (2021) demonstrate that Transformer-based sentence embeddings combined with various clustering methods are effective for short text clustering, while Keraghel et al. (2024) highlight the influence of LLM size on clustering performance. Viswanathan et al. (2024) further show that LLMs can enhance clustering at three stages: before clustering (improving input features), during clustering (providing constraints), and after clustering (post-correction). Zhang et al. (2023) propose CLUSTERLLM, a framework that leverages feedback from instruction-tuned LLMs like ChatGPT. Additionally, Petukhova et al. (2025) investigate how different textual embeddings, particularly those from LLMs, and various clustering algorithms affect text dataset clustering.

4 Datasets

In this project, we used six open-source datasets that cover a variety of text clustering challenges. All of these datasets are open-sourced and easily accessible. The SyskillWebert (Pazzani, 2005) dataset contains user-rated web pages and supports exploratory clustering for recommendation systems. 20Newsgroups (Mitchell, 1997) is a well-known benchmark offering noisy and unstructured text for robust clustering evaluation. We also include MN-DS (Petukhova and Fachada, 2023), a multimedia news dataset with a hierarchical label structure, enabling clustering experiments at multiple levels. The Reuters (Lewis, 1987) dataset is a standard benchmark in text mining, consisting of news articles from the Reuters agency. Finally, BBC (Greene and Cunningham, 2006) is a news article dataset collected from the BBC website, consisting of documents from five topical categories, and BBC-Sports is a subset of BBC news articles focused exclusively on sports-related content. The details of the datasets are presented in Table 1.

5 Methodology

In this project, we demonstrate the effectiveness of text clustering using embeddings extracted from

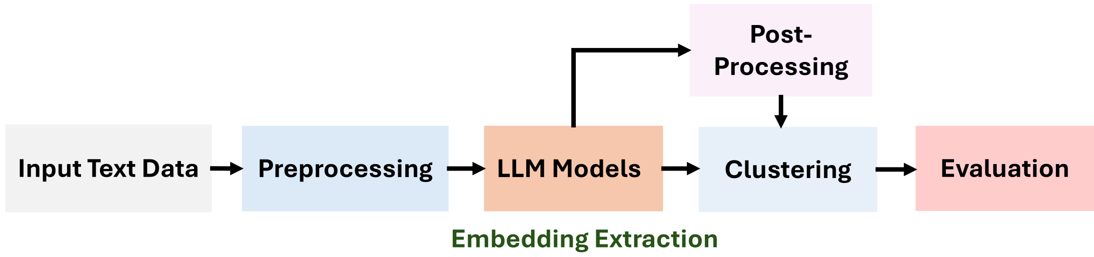


Figure 1: Overview of the proposed approach

Table 1: Details of proposed datasets for the project.

Dataset	# of Documents	# of Classes
SyskillWebert	333	4
20Newsgroups	18,846	20
MN-DS	10,917	109
Reuters	8,654	65
BBC	2,225	5
BBC-Sports	737	5

multiple open-source LLMs on several publicly available datasets. In the initial stage, we apply a unified preprocessing pipeline across all corpora to remove noise and standardize inputs, including cleaning, tokenization, stopword removal, and filtering of empty or invalid samples. The cleaned documents are then passed through multiple open-source LLMs to obtain raw, high-dimensional embedding representations for each text. Since these raw embedding spaces often contain global biases and a few dominant directions that can distort the clustering, we apply post-processing steps to improve cluster separability, such as mean-centering, removing top principal components, dimensionality reduction, and normalization. Finally, we cluster the post-processed embeddings using a suite of classical and density-based algorithms and evaluate clustering quality using standard metrics to compare the performance across datasets, models, and clustering methods. The framework of the proposed method is shown in Figure 1.

5.1 Data preprocessing

The objective of text data preprocessing is to reduce miscellaneous items and underscore key patterns of the data and improve the performance of clustering algorithms using LLM embeddings. For all the datasets used in our work, a series of preprocessing steps are performed to maintain the quality of the input data. In this regard, the initial stage involved removing HTML and XML markups, URLs, email addresses, and regular expressions from each text document and stripping all non-alphabetic characters to reduce verbosity. Following the cleaning process, all of the empty documents were removed. The cleaned text was subsequently tokenized, and commonly occurring English stop words were discarded to minimize redundancy.

5.2 LLM-Embeddings

LLM embeddings help to cluster the text more efficiently compared to the embeddings extracted using traditional methods, e.g., tf-idf, Word2Vec, GloVe, since they turn each text into a vector where semantic similarity becomes geometric proximity. An LLM model takes a text x and outputs an embedding vector $e \in \mathbb{R}^d$, where $e = f(\theta)$, and $f(\theta)$ is the encoder module, the decoder module, or an LLM adapter as an embedder. In this project, we used six open-source LLMs listed in Table 2 to extract embeddings from six different datasets. MPNet (Song et al., 2020) and DistilBERT (Sanh et al., 2019) are encoder-only models primarily designed for classification tasks. GPT-2 (Radford et al., 2019) and Qwen2.5 (Hui et al., 2024) are decoder-only models that excel at free-form text generation. In contrast, BART (Lewis et al., 2020) and T5 (Raffel et al., 2020) are encoder–decoder models, making them well-suited for tasks where the input and output differ significantly, such as machine translation. Besides, we used traditional TF-IDF embedding-based clustering as a baseline to compare the model performance across different datasets. In this project, three types of models, namely encoder-only, decoder-only, and encoder-decoder type LLM models, are investigated to extract the embeddings.

Table 2: LLM Models Used for Embedding Extraction

Embeddings	Type	Configuration
TF-IDF	–	min-df = 5, max features = 512 (no parameters)
All-MPNet-Base-V2	Encoder-only	backbone = BERT-based size, parames = 110 M
DistilBERT-Base-Uncased	Encoder-only	params = 66 M
GPT-2	Decoder-only	params = 117 M
Qwen2.5-0.5B-Instruct	Decoder-only	params = 490 M
BART-Large	Encoder–Decoder	params = 406 M
T5-Base	Encoder–Decoder	params = 220 M

5.3 Post-Processing of LLM Embeddings

The motivation of the post-processing is to reduce global biases, dominant directions, and anisotropy and obtain more ideal geometry of LLM embeddings for clustering (Mu et al., 2017; Su et al., 2021). Following the extraction of the raw embedding matrix $E \in \mathbb{R}^{N \times D}$ produced by LLMs, a series of post-processing techniques are applied before clustering, which are listed as follows:

1. **Mean-Centering:** The feature-wise mean embedding is subtracted from the raw embedding to remove the global bias and offset direction that often improves the geometry of the embeddings for clustering.

$$X \leftarrow E - \mu, \qquad \mu = \frac{1}{N}\sum_{i=1}^{N} E_i$$

2. **Top Principal Components Removal:** It performs support vector decomposition (SVD) on the centered embeddings and removes the projection onto the top k components. The technique helps to remove dominant directions that can collapse cosine similarity and cause large “hub” effects, which can hurt clustering. To execute the step, SVD is computed as follows:

$$X = USV^T$$

Then, the top K principal components (PCs) $P = V_{1:k}$ are taken and subtracted from the common directions.

$$X \leftarrow X - XP^\top P$$

In this work, we take k as 1.

3. **PCA Dimensionality Reduction:** The dimensionality reduction by principal component analysis (PCA) helps to reduce noise and makes clustering more stable and faster.

$$X \leftarrow \mathrm{PCA}_d(X), \qquad d = \texttt{pca\_dim}$$

In this work, we keep pca_dim as 256.

4. **L2 Normalization:** The L2 normalization process normalizes each vector to unit length.

$$x_i \leftarrow \frac{x_i}{\|x_i\|_2}$$

The method forces all vectors onto the unit sphere, so cosine-similarity is well-behaved and the Euclidean distance becomes more comparable across points.

5.4 Clustering Techniques

In this work, we adopt five different clustering techniques on the post-processed embeddings.

5.4.1 K-Means Clustering

K-means clustering is an iterative and centroid-based popular clustering algorithm that divides M data points in N dimensions into K different clusters so that the distance of data points within a cluster is minimized (Hartigan and Wong, 1979). Given data points $\{x_i \in \mathbb{R}^d\}_{i=1}^n$, the K-means clustering algorithm assigns each point to one cluster and a set of K centroids $\mu_{k}{}_{k=1}^{n}$, that minimizes the objective.

$$\min_{\{C_k\}_{k=1}^K, \{\mu_k\}_{k=1}^K} J = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|_2^2 .$$

K-means is simple, scalable, and effective when clusters are roughly spherical and similarly sized, but it can be sensitive to initialization and outliers; common remedies include running multiple initializations (e.g., k-means++) and selecting K using criteria such as the elbow method or silhouette score.

5.4.2 Agglomerative Clustering

Agglomerative clustering is a special type of hierarchical clustering that builds a hierarchical cluster using a bottom-up approach. It begins by treating each data point as an individual cluster. It progressively merges the closest pairs of clusters at each step until only a single cluster remains, encompassing the entire dataset (Müllner, 2011). In agglomeration clustering, it is essential to establish a method for measuring the distance between clusters. Several commonly used cluster distance metrics can be applied to a predefined distance function for individual data points. In the single linkage approach, the distance between two clusters A and B can be defined as:

$$d(A, B) = \min_{x \in A, y \in B} d(x, y) \tag{1}$$

This method corresponds to constructing a minimum spanning tree for the graph defined by the distance function, following Kruskal's algorithm (Greenberg, 1998).

5.4.3 Spectral Clustering

Spectral clustering method is a graph-based machine learning technique that identifies clusters within data by converting the dataset into a graph and analyzing its structure through spectral decomposition, which involves computing eigenvalues and eigenvectors (Xiang and Gong, 2008). It treats data points as nodes in a graph and specifies the likeness of different nodes using a similarity matrix S. For the Gaussian kernel, the similarity matrix can be computed as follows:

$$S_{ij} = exp(-\frac{|x_i - x_j|}{2\sigma^2}) \tag{2}$$

Since the clustering method uses graph connectivity, it captures the complex structure of the data.

5.4.4 HDBSCAN Clustering

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering method like DBSCAN, except it handles variable-density clusters much better over varying ϵ (McInnes et al., 2017). It executes the concepts of DBSCAN across many density levels, builds a cluster hierarchy, and extracts a flat clustering by choosing the most stable clusters. Given data $X = \{x_1, \ldots, x_n\}$ in a metric space with distance d, it picks the minimum number of samples k, and the core distance of a point is defined as its k-nearest-neighbor distance.

5.4.5 Spherical K-Means Clustering

Spectral K-Means clustering combines the graph-based power of spectral clustering with the efficiency of K-Means. It utilizes the eigenvectors of a Laplacian matrix to transform complex arbitrary-shaped data into a lower-dimensional embedding space and subsequently applies the standard K-means technique to obtain those clusters (Ng et al., 2001). The technique overcomes the shortcoming associated with the traditional K-means clustering algorithm to cluster the non-convex shapes.

In all of the clustering algorithms, the number of clusters is chosen based on the number of labels present in the datasets. The selected algorithms and their respective parameters are listed in Table 3.

Table 3: Used clustering algorithms and respective parameters

Algorithm	Parameters
K-Means	distance = Euclidean, n-init = 20
Agglomerative	metric = cosine, linkage = average
Spectral	affinity = nearest neighbors
HDBSCAN	distance = Euclidean, cluster selection method= eom
Spherical K-Means	normalization = l_2

6 Computational Resources

For most of the experiments in the project, we used A100, L4 and T4 GPUs of Google Colab Pro. For larger models such as Qwen2.5-7B-Instruct and Qwen2.5-14B-Instruct, we use RTX 6000 CUDA 12.8.

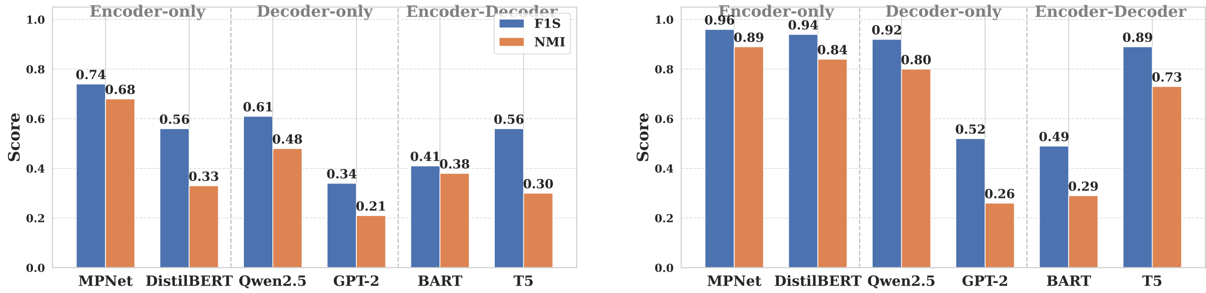
7 Results

In this section, we describe all the different metrics that have been used to evaluate the text clustering performance for the different embedding models on different datasets and then analyze the overall result from various viewpoints.

7.1 Evaluation Metrics

To gain a comprehensive evaluation of the different embedding models and clustering algorithms, and all their possible combinations, we used six different performance metrics in this project.

1. **Weighted F1-Score (F1S):** F1-score is used as an external clustering metric that measures how well the predicted clusters match with the true classes. It finds the best one-to-one relabeling of predicted cluster IDs to true class IDs that maximizes the number of



(a) Reuters Dataset

(b) BBC Dataset

Figure 2: Comparison of performance metrics (F1-score and NMI) for Reuters and BBC datasets for different types of model architecture

matches (Kuhn, 1955). Weighted F1S is calculated as follows:

$$F1_{\text{weighted}} = \sum_{c=1}^{C} \frac{n_c}{N} F1_c$$

where $F1_c$ is the standard classification F1.

2. **Normalized Mutual Information (NMI):** NMI is a technique to quantify the similarity between predicted labels vs. ground truth labels by measuring shared information, normalized by their individual entropies. It allows comparison between different numbers of clusters, indicating how much uncertainty about true classes is reduced by knowing cluster assignments. NMI is calculated as follows:

$$NMI(Y,C) = \frac{2 \times I(Y;C)}{H(Y) + H(C)}$$

3. **Adjusted Rand Index (ARI):** Adjusted rank index quantifies the similarity between two clusterings. For every pair of samples, ARI checks whether the pair is in the same cluster in both labelings or in different clusters in both labelings.

4. **Fowlkes–Mallows Index (FMI):** The Fowlkes–Mallows Index is defined as the geometric mean of pairwise precision and recall.

$$\text{FMI} = \frac{\text{TP}}{\sqrt{(\text{TP}+\text{FP})(\text{TP}+\text{FN})}}$$

5. **Davies-Bouldin Index (DBI):** Davies-Bouldin Index (DBI) is an internal clustering metric that scores how well clusters are compact and well-separated. The lower the value of DBI, the better the performance is. The DBI is calculated as follows:

$$\text{DBI} = \frac{1}{K} \sum_{i=1}^{K} \max_{j \neq i} \left(\frac{S_i + S_j}{M_{ij}} \right)$$

where K is the number of clusters, S_i denotes the within-cluster scatter for cluster i, and $M_{i,j}$ is the distance between cluster centroids i and j.

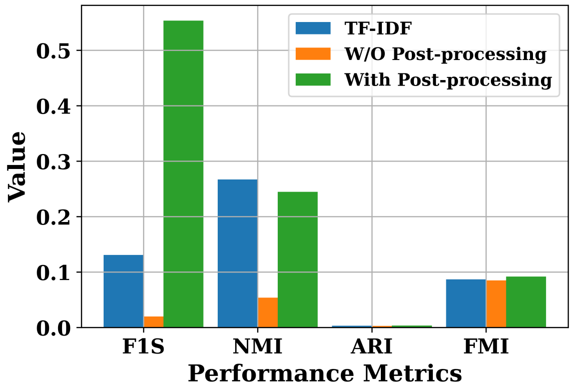
6. **Silhouette Coefficient:**Silhouette coefficient measures the similarity of an object to its cohesive cluster compared to other separated clusters. For each data point t, the silhouette coefficient s(i) can be calculated as follows:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}$$

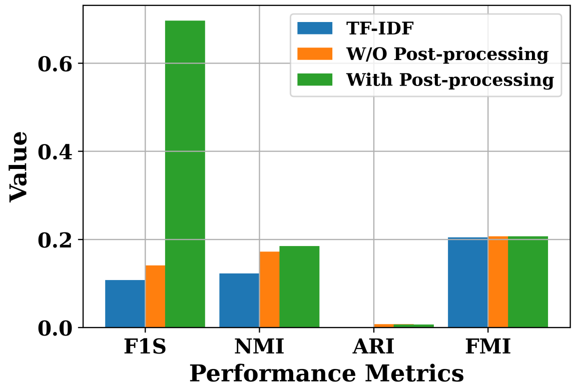
7.2 Clustering Performance Evaluation

In this work, we analyze the text clustering performance on six different datasets using the embedding from six different LLMs. For each model and dataset combination, we analyze the text clustering performance using five different clustering algorithms and report the results using six different metrics. Besides, we generate the baseline result for each of the datasets using the classical term frequency-inverse document frequency (TF-IDF) method. We report the results of the best-performing clustering algorithm for all model and dataset combinations in TABLE 4. The detailed results of all algorithms for all models and datasets are reported in the Appendix section.

From the table, we can observe that, overall, the LLM-based embeddings outperform the classic



(a) MN-DS Dataset



(b) 20 News Group Dataset

Figure 3: Comparison of performance metrics for MN-DS and 20-News Group datasets

TF-IDF-based text clustering method, apart from the case of the SyskillWebert dataset. However, the SyskillWebert dataset is the smallest dataset with the lowest amount of data, which could be the reason behind the TF-IDF method outperforming the LLMs. We can also observe a significant performance gap between the TF-IDF and the LLMs as the dataset size increases. We can also observe that the dataset size has a direct effect on the overall performance as well. As we notice for smaller datasets, such as the SyskillWebert, BBC, and BBC-Sports, most of the embedding methods show impressive clustering performance, while as we increase the dataset size and number of clusters to classify, the task starts becoming increasingly difficult as well. As for the performance difference between different types of models and different clustering algorithms, we go into more details in the following subsections.

7.3 Effect of Model Architecture

In this project, we used three types of model architecture: encoder-only, decoder-only, and encoder-decoder models. For each of these types of architecture, we used two representative models. From TABLE 4, we can observe that the performance for each type of model varies as we change the dataset; however, a general trend can be noticed that the encoder-only models are typically performing better than the other two types of model, and both of the encoder-only model shows somewhat consistent performance for the same dataset. To analyze this further, we have plotted the F1S and NMI of all the models for the Reuters and BBC datasets in Fig. 2. From the figure, we can observe that for both of the datasets, the encoder-only models (MPNet and Distillbert) exhibit the best result, and the Qwen2.5 0.5B instruction-tuned model also shows similar performance. While models like GPT-2 and BART struggle the most because of how they were trained and the downstream tasks that these models specialize in. An interesting trend we notice is that the T5 model exhibits consistently better clustering abilities compared to the BART model whcih is the other encoder-decoder model, even though it is smaller in terms of parameters compared to the BART model. We explore the effect of model parameter size in more detail in subsection 8.6.

7.4 Comparison of Different Clustering Algorithms

We evaluated a total of five clustering algorithms in our project. TABLE 4 shows the performance of the best performing clustering algorithm for each model and dataset combination and the detailed result of each clustering algorithm for each dataset using each model is provided in the Appendix section in TABLE 6, 7, 9, 8, 10, and 11. To better understand the performance difference between different types of clustering algorithms, we have plotted the F1-score of these different algorithms using different models on the Reuters and BBC datasets in Fig. 4.

After analyzing the results, the first high-level takeaway we obtain is that it is difficult to pinpoint one single champion clustering algorithm that outperforms all other algorithms for all the datasets and models. The performance of these clustering algorithms varies a lot as we change the dataset and the model. We notice that there exists a cor-

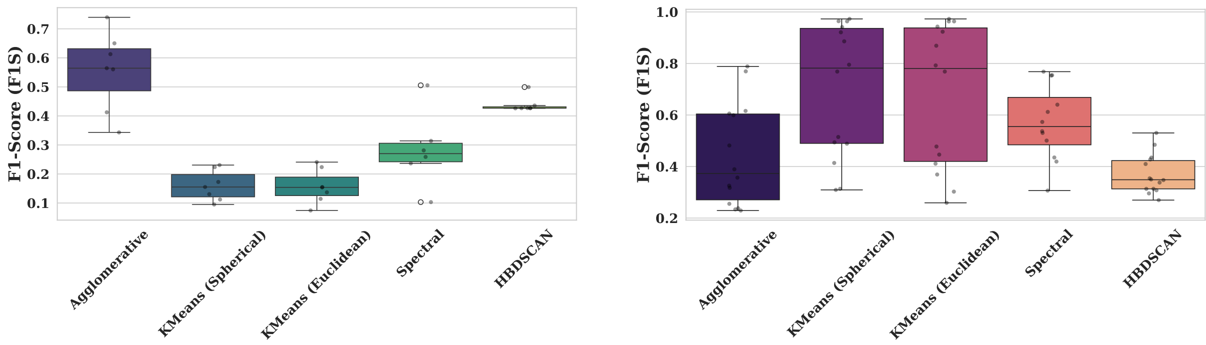
Table 4: Text clustering performance for the best-performing clustering algorithm for each combination of dataset and embedding model.

Dataset	Model Type	Model	Best Algorithms	Metrics					
				F1S	NMI	ARI	FMI	Silhouette	Davies-Bouldin
Reuters	Baseline	TF-IDF	Agglomerative	0.650	0.565	0.675	0.754	0.131	2.997
	Encoder-only	MPNet	Agglomerative	0.740	0.682	0.838	0.878	0.100	2.321
		Distillbert	Agglomerative	0.564	0.330	0.220	0.494	0.216	1.402
	Decoder-only	Qwen2.5	Agglomerative	0.613	0.485	0.523	0.646	0.135	1.481
		GPT-2	Agglomerative	0.343	0.213	0.074	0.299	0.140	1.965
	Encoder-Decoder	Bart	Agglomerative	0.412	0.376	0.227	0.398	0.209	1.448
		T5	Agglomerative	0.560	0.304	0.226	0.496	0.180	1.542
20Newsgroups	Baseline	TF-IDF	KMeans (Spherical)	0.379	0.399	0.126	0.203	0.034	6.067
	Encoder-only	MPNet	KMeans (Spherical)	0.570	0.563	0.393	0.425	0.070	4.477
		Distillbert	Spectral	0.353	0.434	0.185	0.272	0.015	2.574
	Decoder-only	Qwen2.5	KMeans (Euclidean)	0.364	0.388	0.213	0.258	0.112	2.442
		GPT-2	KMeans (Spherical)	0.149	0.131	0.051	0.108	0.151	3.506
	Encoder-Decoder	Bart	KMeans (Euclidean)	0.149	0.142	0.049	0.102	0.063	2.991
		T5	KMeans (Spherical)	0.280	0.292	0.146	0.192	0.022	3.744
SyskillWebert	Baseline	TF-IDF	KMeans (Euclidean)	0.946	0.835	0.867	0.905	0.068	4.944
	Encoder-only	MPNet	KMeans (Euclidean)	0.765	0.612	0.675	0.765	0.192	3.058
		Distillbert	Spectral	0.696	0.533	0.545	0.683	0.143	2.753
	Decoder-only	Qwen2.5	KMeans (Spherical)	0.705	0.493	0.570	0.694	0.275	2.227
		GPT-2	HBDSCAN	0.506	0.293	0.146	0.532	-0.355	2.434
	Encoder-Decoder	Bart	Spectral	0.482	0.227	0.193	0.416	0.191	1.973
		T5	KMeans (Spherical)	0.667	0.398	0.484	0.636	0.192	2.749
MN-DS	Baseline	TF-IDF	KMeans (Spherical)	0.343	0.538	0.182	0.196	0.150	3.191
	Encoder-only	MPNet	KMeans (Spherical)	0.376	0.560	0.242	0.249	0.120	2.943
		Distillbert	KMeans (Euclidean)	0.269	0.461	0.140	0.149	0.087	2.800
	Decoder-only	Qwen2.5	KMeans (Euclidean)	0.292	0.490	0.165	0.173	0.134	2.464
		GPT-2	KMeans (Spherical)	0.146	0.328	0.058	0.068	0.102	3.819
	Encoder-Decoder	Bart	KMeans (Euclidean)	0.313	0.503	0.175	0.184	0.096	2.887
BBC	Baseline	TF-IDF	KMeans (Spherical)	0.769	0.651	0.522	0.635	0.043	6.178
	Encoder-only	MPNet	KMeans (Spherical)	0.964	0.889	0.916	0.933	0.130	3.350
		Distillbert	KMeans (Euclidean)	0.943	0.837	0.865	0.893	0.197	2.408
	Decoder-only	Qwen2.5	KMeans (Euclidean)	0.923	0.799	0.820	0.856	0.291	1.987
		GPT-2	KMeans (Spherical)	0.515	0.255	0.193	0.359	0.247	2.417
	Encoder-Decoder	Bart	KMeans (Spherical)	0.489	0.286	0.220	0.397	0.198	2.320
		T5	KMeans (Spherical)	0.886	0.728	0.732	0.787	0.187	2.601
BBC-Sports	Baseline	TF-IDF	KMeans (Spherical)	0.963	0.886	0.902	0.925	0.053	5.668
	Encoder-only	MPNet	KMeans (Spherical)	0.973	0.923	0.924	0.942	0.214	2.451
		Distillbert	KMeans (Spherical)	0.795	0.730	0.683	0.758	0.256	2.140
	Decoder-only	Qwen2.5	Spectral	0.612	0.474	0.284	0.532	0.183	2.109
		GPT-2	Agglomerative	0.326	0.028	-0.017	0.440	0.206	1.314
	Encoder-Decoder	Bart	Spectral	0.419	0.196	0.135	0.364	0.127	2.153
		T5	Spectral	0.501	0.338	0.196	0.472	0.141	2.189

relation between the clustering algorithm and the dataset, as the Agglomerative algorithm performs well for all the models for the Reuters dataset, and similarly for the smaller datasets, K-Means seems to be the best clustering algorithm. The Spectral and HDBSCAN clustering algorithms show more consistent performance when used on embeddings from different models, and in some particular cases, they even outperform the other algorithms. However, we also notice that for the small datasets, the HBDSCAN algorithm does show a consistently poor performance. The Spherical K-Means algorithm consistently outperforms the standard K-Means algorithm, which uses Euclidean distance, indicating that the embedding space produced by these embeddings resides on a hypersphere where magnitude matters less than direction.

7.5 Effect of Post-Processing

In our work, we compared the performance of the proposed method with the traditional TF-IDF-based clustering and clustering without post-processing techniques. The result is displayed in Figure 3 for the MN-DS and 20 Newsgroups datasets. Among different LLMs and clustering techniques, the proposed method performs better for the MPNET Base-V2 model with HDBSCAN clustering. Figure 3 demonstrates superior performance in clustering for all metrics with the post-processing techniques.



(a) Reuters Dataset (b) BBC Dataset

Figure 4: F1-score distribution of different clustering algorithms across all the embedding models for the Reuters and BBC dataset

7.6 Effect of Model Size on Clustering Performance

We investigate the clustering performance across different sizes of Qwen2.5 models. As shown in Table 5, clustering performance degrades as the model size increases. For this experiment, embeddings are extracted from the last hidden states, with no post-processing applied. A possible explanation is the curse of dimensionality: larger models produce higher-dimensional embeddings, which are often more abstract and can reduce the effectiveness of distance-based clustering methods. In high-dimensional spaces, distances between data points become less discriminative, making it more difficult to form well-separated clusters. In the Appendix, Table 12 shows the detailed clustering performance for different clustering algorithms.

7.7 Effect of Embedding Layer

We also analyzed the effect of model depth for the task of text clustering. For this analysis, we experiment with two models, the Qwen2.5 0.5B decoder-only model and the MPNet encoder-only model. We used the Reuters and BBC datasets for this experiment. For both models, we extracted the text embeddings from four different hidden layers of the model that were equally spaced inside the model and compared the performance for each case. The key research question we hoped to answer with this experiment is whether models of such large depth are necessary to generate meaningful embeddings for the task of text clustering.

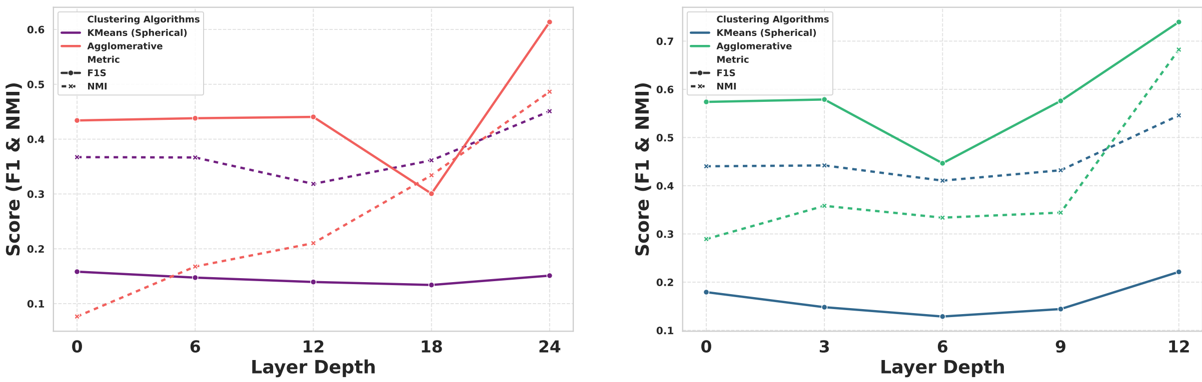
The detailed result of this experiment is presented in TABLE 13 in the Appendix section. We visualize the outcome of this experiment on the Reuters dataset in Fig. 5. From the figure, we can observe that for the best performing clustering algorithm in this case, the performance initially remains similar for the first few hidden layers, and there is also a drop in performance when using embedding from the middle layers; however, towards the end of the model, we notice a rapid increase in the performance. While this does indicate we need the model of the full depth for the most optimized performance, it could also be the case that, because of the nature of the training process, the last layer learns to hold more meaningful final embeddings that help in text clustering.

7.8 Analyzing Cluster Compactness

Finally, apart from simply looking at the performance in terms of the F1-score and analyzing the best performance, we also conduct a holistic analysis of the different datasets and models of different sizes and types in Fig. 6 using both the accuracy and compactness metric in terms of Silhouette Score (SS). The Silhouette Score tells us how tight or separated the detected clusters are geometrically, while the F1 score tells us how these detected clusters compare to the human label. By inspecting Fig. 6, we can observe that for small datasets like BBC, BBC-Sports, and SyskillWebert, even though all the models achieve very high accuracy, the clusters they detect are not that compact. Whereas for larger datasets, in some cases, the bigger models are detecting very compact clusters, which are showing very poor accuracy, indicating that in these cases, the models are detecting some well compact clusters which are not matching with the topics the humans are interested in.

Table 5: Best clustering performance on Reuters dataset for each Qwen2.5 model.

Model	Clustering Algorithm	F1S	NMI	ARI	FMI	Silhouette	Davies-Bouldin
Qwen2.5-0.5B-Instruct	Agglomerative	0.6135	0.4864	0.5235	0.6470	0.1408	1.4761
Qwen2.5-1.5B-Instruct	Agglomerative	0.5913	0.4484	0.5102	0.6332	0.1907	1.3419
Qwen2.5-3B-Instruct	Agglomerative	0.5627	0.3280	0.2496	0.5024	0.0912	1.3083
Qwen2.5-7B-Instruct	Agglomerative	0.5561	0.3298	0.3063	0.5192	0.1908	1.2831
Qwen2.5-14B-Instruct	Agglomerative	0.5646	0.3691	0.3536	0.5415	0.0631	1.3335



(a) Qwen2.5 0.5B Model

(b) MPNet Model

Figure 5: Evolution of clustering quality (F1-Score and NMI) across the layers of an embedding model

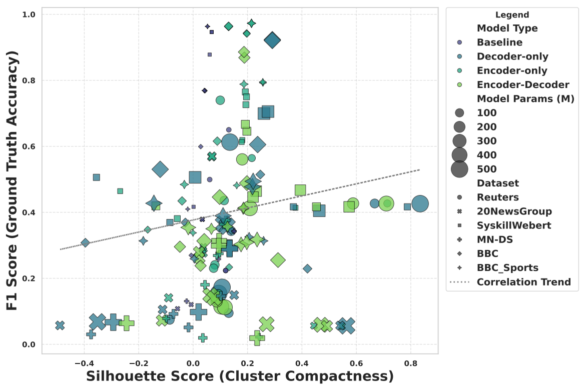


Figure 6: Analysis of accuracy vs cluster compactness across all the models of different parameter sizes and datasets

8 Contributions of group members

In this project, three members are in the group.

- Barproda Halder: She developed the initial coding framework, produced clustering performance results on one dataset, and evaluated performance across multiple Qwen2.5 models. For the report, she wrote the related work, datasets, and portions of the experimental results, along with other sections as they were developed.
- Nayeeb Rashid: He generated clustering performance for four datasets and expanded the code with additional clustering algorithms. For the report, he wrote most portions of the experimental results, generated tables and figures and contributed to other sections as they were developed.
- Oishy Saha: She generated clustering performance for one dataset and did post-processing on the embedding for two datasets. For the report, she wrote the problem statement, methodology, and portions of experimental results (evaluation metrics and effects of post processing), along with other sections as they were developed.

9 Conclusion

The main takeaway of our project is that embeddings from encoder-only models cluster more effectively, and clustering performance further improves with post-processing steps such as applying PCA before clustering. We observed that decoder-only and encoder–decoder LLMs perform comparatively worse, which is intuitive given that they are trained with different objectives. Moreover, increasing the size of decoder-only models leads to a decline in clustering performance, highlighting the curse of dimensionality. As a future research direction, we plan to focus on decoder-only models and investigate ways to enhance the clusterability of their embeddings as an emerging property of larger model scales.

10 AI Disclosure

We used ChatGPT, Gemini3.0, and QuillBot as AI assistance in our work.

The overall experience was good. We used ChatGPT for the debugging of the code, paraphrasing the initial drafts, checking grammatical errors, and finding out the relevant papers on several topics. In coding, ChatGPT and Gemini 3.0 were used to create some code blocks, specifically looking for different clustering algorithms that were faster. While writing in Overleaf, ChatGPT was used to generate complex equations and large comparison tables. AI tools really helped us to finalize the draft faster.

References

Aggarwal, C. C. and Zhai, C. (2012). A survey of text clustering algorithms. In *Mining text data*, pages 77–128. Springer.

Berahmand, K., Daneshfar, F., Dorosti, M., Aghajani, M. J., et al. (2022). An improved deep text clustering via local manifold of an autoencoder embedding.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Decherchi, S., Gastaldo, P., Redi, J., and Zunino, R. (2009). A text clustering framework for information retrieval. *Journal of information Assurance and Security*, 4:174–182.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Greenberg, H. J. (1998). Greedy algorithms for minimum spanning tree. *University of Colorado at Denver*.

Greene, D. and Cunningham, P. (2006). Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*, pages 377–384. ACM Press.

Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108.

Hui, B., Yang, J., Cui, Z., Yang, J., Liu, D., Zhang, L., Liu, T., Zhang, J., Yu, B., Lu, K., et al. (2024). Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Jardim, S. and Mora, C. (2022). Customer reviews sentiment-based analysis and clustering for market-oriented tourism services and products development or positioning. *Procedia Computer Science*, 196:199–206. International Conference on ENTERprise Information Systems / ProjMAN - International Conference on Project MANagement / HCist - International Conference on Health and Social Care Information Systems and Technologies 2021.

Katz, A., Shakir, U., and Chambers, B. (2023). The utility of large language models and generative ai for education research. *arXiv preprint arXiv:2305.18125*.

Keraghel, I., Morbieu, S., and Nadif, M. (2024). Beyond words: a comparative analysis of llm embeddings for effective clustering. In *International Symposium on Intelligent Data Analysis*, pages 205–216. Springer.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Lewis, D. (1987). Reuters-21578 Text Categorization Collection. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C52G6M.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 7871–7880.

McInnes, L., Healy, J., Astels, S., et al. (2017). hdbscan: Hierarchical density based clustering. *J. Open Source Softw.*, 2(11):205.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mitchell, T. (1997). Twenty Newsgroups. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5C323.

Mu, J., Bhat, S., and Viswanath, P. (2017). All-but-the-top: Simple and effective postprocessing for word representations. *arXiv preprint arXiv:1702.01417*.

Müllner, D. (2011). Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*.

Ng, A., Jordan, M., and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14.

Pazzani, M. (2005). Syskill and webert web page ratings.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Petukhova, A. and Fachada, N. (2023). Mn-ds: A multilabeled news dataset for news articles hierarchical classification. *Data*, 8(5):74.

Petukhova, A., Matos-Carvalho, J. P., and Fachada, N. (2025). Text clustering with large language model embeddings. *International Journal of Cognitive Computing in Engineering*, 6:100–108.

Pugachev, L. and Burtsev, M. (2021). Short text clustering with transformers. *arXiv preprint arXiv:2102.00541*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Technical Report*.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y. (2020). Mpnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems*, 33:16857–16867.

Strehl, A. and Ghosh, J. (2002). Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617.

Su, J., Cao, J., Liu, W., and Ou, Y. (2021). Whitening sentence representations for better semantics and faster retrieval. *arXiv preprint arXiv:2103.15316*.

Torabizadeh, K., Hedjam, R., Allaoui, M., and Abdulrazak, B. (2025). Embedding-enhanced patient clustering for customized medical report summarization using llms. In *2025 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)*, pages 1–6. IEEE.

Viswanathan, V., Gashteovski, K., Gashteovski, K., Lawrence, C., Wu, T., and Neubig, G. (2024). Large language models enable few-shot clustering. *Transactions of the Association for Computational Linguistics*, 12:321–333.

Xiang, T. and Gong, S. (2008). Spectral clustering with eigenvector selection. *Pattern Recognition*, 41(3):1012–1029.

Zhang, Y., Wang, Z., and Shang, J. (2023). Clusterllm: Large language models as a guide for text clustering. *arXiv preprint arXiv:2305.14871*.

A Detailed Result on All Datasets

Table 6: Clustering performance for Reuters across different models and clustering algorithms

Model Type	Model	Clustering Algorithm	F1S	NMI	ARI	FMI	Silhouette	Davies-Bouldin
Baseline	TF-IDF	KMeans (Euclidean)	0.224	0.493	0.094	0.236	0.120	3.470
		KMeans (Spherical)	0.224	0.493	0.094	0.236	0.120	3.470
		Agglomerative	0.650	0.565	0.675	0.754	0.131	2.997
		HDBSCAN	0.499	0.343	0.070	0.389	0.061	2.228
Encoder-only	MPNet	KMeans (Euclidean)	0.241	0.555	0.116	0.275	0.080	3.365
		KMeans (Spherical)	0.231	0.553	0.115	0.274	0.075	3.354
		Agglomerative	0.740	0.682	0.838	0.878	0.100	2.321
		HDBSCAN	0.436	0.019	0.014	0.497	0.114	4.609
		Spectral	0.505	0.549	0.299	0.447	0.033	2.390
	DistillBERT	KMeans (Euclidean)	0.155	0.431	0.070	0.201	0.091	2.764
		KMeans (Spherical)	0.155	0.437	0.069	0.200	0.094	2.775
		Agglomerative	0.564	0.330	0.220	0.494	0.216	1.402
		HDBSCAN	0.427	0.004	0.001	0.500	0.714	0.584
		Spectral	0.259	0.463	0.140	0.284	0.038	2.273
Decoder-only	Qwen2.5	KMeans (Euclidean)	0.154	0.452	0.076	0.214	0.094	2.659
		KMeans (Spherical)	0.173	0.456	0.086	0.230	0.106	2.599
		Agglomerative	0.613	0.485	0.523	0.646	0.135	1.481
		HDBSCAN	0.426	0.002	0.000	0.500	0.834	0.499
		Spectral	0.314	0.508	0.201	0.350	0.026	2.164
	GPT-2	KMeans (Euclidean)	0.074	0.191	0.026	0.118	-0.086	1.713
		KMeans (Spherical)	0.095	0.252	0.035	0.136	0.131	3.603
		Agglomerative	0.343	0.213	0.074	0.299	0.140	1.965
		HDBSCAN	0.427	0.004	0.001	0.500	0.667	0.765
		Spectral	0.103	0.228	0.026	0.122	-0.109	1.755
Encoder-Decoder	BART	KMeans (Euclidean)	0.114	0.362	0.056	0.176	0.103	2.612
		KMeans (Spherical)	0.112	0.367	0.055	0.173	0.117	2.660
		Agglomerative	0.412	0.376	0.227	0.398	0.209	1.448
		HDBSCAN	0.427	0.003	0.001	0.500	0.710	0.718
		Spectral	0.236	0.411	0.121	0.264	0.074	2.194
	T5	KMeans (Euclidean)	0.137	0.400	0.063	0.187	0.087	2.890
		KMeans (Spherical)	0.130	0.401	0.062	0.187	0.086	3.064
		Agglomerative	0.560	0.304	0.226	0.496	0.180	1.542
		HDBSCAN	0.427	0.001	0.000	0.501	0.588	0.405
		Spectral	0.281	0.447	0.158	0.304	0.028	2.354

Table 7: Clustering performance on 20NewsGroup dataset across different models and clustering algorithms

Model type	Model	Clustering Algorithm	F1S	NMI	ARI	FMI	Silhouette	Davies-Bouldin
Baseline	TF-IDF	KMeans (Euclidean)	0.379	0.399	0.126	0.203	0.034	6.067
		KMeans (Spherical)	0.379	0.399	0.126	0.203	0.034	6.067
		Agglomerative	0.120	0.148	0.041	0.214	-0.007	5.093
		HDBSCAN	0.108	0.123	0.001	0.205	-0.057	2.848
Encoder-only	MPNet	KMeans (Euclidean)	0.568	0.563	0.395	0.427	0.068	4.360
		KMeans (Spherical)	0.570	0.563	0.393	0.425	0.070	4.477
		Agglomerative	0.277	0.462	0.192	0.337	0.037	4.136
		HDBSCAN	0.141	0.172	0.007	0.207	-0.090	2.999
		Spectral	0.476	0.600	0.313	0.397	0.030	3.644
	DistillBERT	KMeans (Euclidean)	0.286	0.324	0.148	0.196	0.035	3.026
		KMeans (Spherical)	0.295	0.329	0.162	0.209	0.038	3.057
		Agglomerative	0.056	0.006	0.000	0.220	0.444	1.478
		HDBSCAN	0.055	0.002	0.000	0.220	0.547	1.984
		Spectral	0.353	0.434	0.185	0.272	0.015	2.574
Decoder-only	Qwen2.5	KMeans (Euclidean)	0.364	0.388	0.213	0.258	0.112	2.442
		KMeans (Spherical)	0.358	0.388	0.211	0.257	0.125	2.472
		Agglomerative	0.056	0.006	0.000	0.222	0.564	1.266
		HDBSCAN	0.068	0.045	0.001	0.211	-0.352	2.506
		Spectral	0.348	0.438	0.157	0.259	0.067	1.993
	GPT-2	KMeans (Euclidean)	0.105	0.057	0.020	0.076	-0.112	1.492
		KMeans (Spherical)	0.149	0.131	0.051	0.108	0.151	3.506
		Agglomerative	0.057	0.009	0.000	0.220	0.540	1.776
		HDBSCAN	0.057	0.012	0.000	0.222	-0.489	1.639
		Spectral	0.123	0.095	0.032	0.096	-0.111	1.513
Encoder-Decoder	BART	KMeans (Euclidean)	0.149	0.142	0.049	0.102	0.063	2.991
		KMeans (Spherical)	0.140	0.138	0.046	0.100	0.078	3.016
		Agglomerative	0.061	0.010	0.000	0.209	0.270	1.559
		HDBSCAN	0.059	0.003	0.000	0.207	0.470	1.580
		Spectral	0.178	0.223	0.070	0.158	0.024	2.276
	T5	KMeans (Euclidean)	0.262	0.282	0.141	0.188	0.023	3.690
		KMeans (Spherical)	0.280	0.292	0.146	0.192	0.022	3.744
		Agglomerative	0.055	0.005	0.000	0.223	0.492	1.792
		HDBSCAN	0.072	0.025	0.003	0.201	-0.115	3.502
		Spectral	0.276	0.363	0.130	0.233	-0.020	2.883

Table 8: Clustering performance of SyskillBert with different embeddings and algorithms

Model Type	**Model**	**Clustering Algorithm**	F1S	NMI	ARI	FMI	Silhouette	Davies-Bouldin
Baseline	TF-IDF	KMeans (Euclidean)	0.946	0.835	0.867	0.905	0.068	4.944
		KMeans (Spherical)	0.946	0.835	0.867	0.905	0.068	4.944
		Agglomerative	0.878	0.646	0.712	0.794	0.061	4.921
		HDBSCAN	0.417	0.183	0.042	0.411	0.002	6.902
Encoder-only	MPNet	KMeans (Euclidean)	0.765	0.612	0.675	0.765	0.192	3.058
		KMeans (Spherical)	0.750	0.614	0.673	0.764	0.196	3.110
		Agglomerative	0.726	0.565	0.620	0.744	0.194	2.199
		HDBSCAN	0.381	0.152	0.016	0.430	-0.058	3.982
		Spectral	0.726	0.597	0.635	0.747	0.116	2.523
	Distillbert	KMeans (Euclidean)	0.619	0.401	0.437	0.601	0.181	2.472
		KMeans (Spherical)	0.613	0.408	0.425	0.591	0.179	2.482
		Agglomerative	0.414	0.035	0.008	0.528	0.376	0.726
		HDBSCAN	0.464	0.172	0.052	0.518	-0.268	2.972
		Spectral	0.696	0.533	0.545	0.683	0.143	2.753
Decoder-only	Qwen2.5	KMeans (Euclidean)	0.699	0.491	0.557	0.684	0.260	2.237
		KMeans (Spherical)	0.705	0.493	0.570	0.694	0.275	2.227
		Agglomerative	0.405	0.026	-0.002	0.520	0.464	1.036
		HDBSCAN	0.506	0.382	0.198	0.496	0.007	2.541
		Spectral	0.688	0.506	0.521	0.661	0.234	2.138
	GPT-2	KMeans (Euclidean)	0.369	0.052	0.044	0.315	-0.085	1.070
		KMeans (Spherical)	0.417	0.034	0.041	0.400	0.369	1.391
		Agglomerative	0.417	0.025	0.007	0.524	0.787	0.782
		HDBSCAN	0.506	0.293	0.146	0.532	-0.355	2.434
		Spectral	0.387	0.066	0.052	0.343	-0.232	0.964
Encoder-Decoder	Bart	KMeans (Euclidean)	0.464	0.153	0.117	0.368	0.232	1.898
		KMeans (Spherical)	0.446	0.151	0.114	0.360	0.220	2.031
		Agglomerative	0.417	0.028	0.006	0.528	0.573	0.563
		HDBSCAN	0.467	0.087	0.079	0.501	0.394	2.305
		Spectral	0.482	0.227	0.193	0.416	0.191	1.973
	T5	KMeans (Euclidean)	0.646	0.375	0.456	0.620	0.198	2.639
		KMeans (Spherical)	0.667	0.398	0.484	0.636	0.192	2.749
		Agglomerative	0.417	0.028	0.006	0.528	0.453	0.749
		HDBSCAN	0.420	0.301	0.072	0.477	-0.137	3.110
		Spectral	0.616	0.370	0.358	0.544	0.137	3.052

Table 9: Clustering performance of MN-DS with different embeddings and algorithms

Model Type	Model	Clustering Algorithms	Metrics					
			F1S	**NMI**	**ARI**	**FMI**	**Silhouette**	**Davies-Bouldin**
Baseline	TF-IDF	KMeans (Euclidean)	0.343	0.538	0.182	0.196	0.150	3.191
		KMeans (Spherical)	0.343	0.538	0.182	0.196	0.150	3.191
		Agglomerative	0.258	0.490	0.159	0.197	0.093	2.797
		HDBSCAN	0.131	0.267	0.003	0.087	-0.024	2.300
Encoder-only	MPNet	KMeans (Euclidean)	0.364	0.557	0.231	0.239	0.113	2.964
		KMeans (Spherical)	0.376	0.560	0.242	0.249	0.120	2.943
		Agglomerative	0.181	0.454	0.081	0.155	0.044	2.379
		HDBSCAN	0.020	0.054	0.003	0.085	0.036	6.804
		Spectral	0.300	0.512	0.043	0.102	0.039	2.541
	Distillbert	KMeans (Euclidean)	0.269	0.461	0.140	0.149	0.087	2.800
		KMeans (Spherical)	0.275	0.469	0.156	0.165	0.097	2.764
		Agglomerative	0.022	0.047	0.000	0.092	0.243	1.097
		HDBSCAN	0.078	0.070	0.003	0.083	-0.099	5.179
		Spectral	0.250	0.456	0.048	0.091	0.003	2.413
Decoder-only	Qwen2.5	KMeans (Euclidean)	0.292	0.490	0.165	0.173	0.134	2.464
		KMeans (Spherical)	0.290	0.490	0.165	0.173	0.134	2.454
		Agglomerative	0.098	0.335	0.032	0.115	0.020	1.408
		HDBSCAN	0.067	0.158	0.002	0.094	-0.293	1.781
		Spectral	0.260	0.474	0.050	0.095	0.002	2.172
	GPT-2	KMeans (Euclidean)	0.092	0.243	0.027	0.037	-0.072	2.065
		KMeans (Spherical)	0.146	0.328	0.058	0.068	0.102	3.819
		Agglomerative	0.051	0.144	0.003	0.076	-0.017	1.918
		HDBSCAN	0.030	0.066	0.001	0.093	-0.375	2.203
		Spectral	0.117	0.297	0.034	0.049	-0.134	2.092
Encoder-Decoder	Bart	KMeans (Euclidean)	0.313	0.503	0.175	0.184	0.096	2.887
		KMeans (Spherical)	0.299	0.504	0.174	0.183	0.094	2.957
		Agglomerative	0.019	0.034	0.000	0.093	0.235	0.949
		HDBSCAN	0.064	0.145	0.002	0.093	-0.245	2.070
		Spectral	0.254	0.468	0.030	0.084	0.001	2.514

Table 10: Clustering performance of BBC with different embeddings and algorithms

Model Type	Model	Clustering Algorithms	Metrics					
			F1S	**NMI**	**ARI**	**FMI**	**Silhouette**	**Davies-Bouldin**
Baseline	TF-IDF	KMeans (Euclidean)	0.769	0.651	0.522	0.635	0.043	6.178
		KMeans (Spherical)	0.769	0.651	0.522	0.635	0.043	6.178
		Agglomerative	0.599	0.565	0.411	0.614	0.028	5.103
		HBDSCAN	0.270	0.120	0.018	0.353	-0.007	6.499
Encoder-only	MPNet	KMeans (Euclidean)	0.963	0.887	0.913	0.931	0.130	3.344
		KMeans (Spherical)	0.964	0.889	0.916	0.933	0.130	3.350
		Agglomerative	0.616	0.672	0.460	0.655	0.089	2.849
		HBDSCAN	0.435	0.333	0.077	0.385	-0.041	2.936
		Spectral	0.537	0.647	0.501	0.643	0.098	3.331
	Distillbert	KMeans (Euclidean)	0.943	0.837	0.865	0.893	0.197	2.408
		KMeans (Spherical)	0.941	0.835	0.861	0.889	0.197	2.398
		Agglomerative	0.234	0.010	0.000	0.448	0.132	1.408
		HBDSCAN	0.347	0.199	0.032	0.406	-0.167	2.794
		Spectral	0.530	0.563	0.331	0.552	0.219	2.246
Decoder-only	Qwen2.5	KMeans (Euclidean)	0.923	0.799	0.820	0.856	0.291	1.987
		KMeans (Spherical)	0.921	0.793	0.815	0.852	0.291	1.986
		Agglomerative	0.605	0.615	0.506	0.664	0.237	1.736
		HBDSCAN	0.530	0.408	0.140	0.423	-0.121	2.085
		Spectral	0.768	0.783	0.680	0.767	0.244	1.924
	GPT-2	KMeans (Euclidean)	0.369	0.132	0.075	0.282	-0.028	1.535
		KMeans (Spherical)	0.515	0.255	0.193	0.359	0.247	2.417
		Agglomerative	0.229	0.004	0.000	0.449	0.420	0.719
		HDBSCAN	0.308	0.253	0.017	0.373	-0.396	2.479
		Spectral	0.435	0.219	0.158	0.368	-0.023	1.392
Encoder-Decoder	Bart	KMeans (Euclidean)	0.446	0.208	0.154	0.342	0.098	2.437
		KMeans (Spherical)	0.489	0.286	0.220	0.397	0.198	2.320
		Agglomerative	0.255	0.061	-0.001	0.425	0.312	1.254
		HDBSCAN	0.313	0.102	0.044	0.403	0.041	1.595
		Spectral	0.640	0.545	0.410	0.572	0.105	2.351
	T5	KMeans (Euclidean)	0.869	0.708	0.694	0.757	0.188	2.568
		KMeans (Spherical)	0.886	0.728	0.732	0.787	0.187	2.601
		Agglomerative	0.238	0.022	0.001	0.444	0.027	2.044
		HDBSCAN	0.296	0.105	0.037	0.367	-0.049	4.879
		Spectral	0.573	0.590	0.351	0.557	0.139	2.276

Table 11: Clustering performance of BBC-Sports with different embeddings and algorithms

Model Type	Model	Clustering Algorithms	Metrics					
			F1S	**NMI**	**ARI**	**FMI**	**Silhouette**	**Davies-Bouldin**
Baseline	TF-IDF	KMeans (Euclidean)	0.963	0.886	0.902	0.925	0.053	5.668
		KMeans (Spherical)	0.963	0.886	0.902	0.925	0.053	5.668
		Agglomerative	0.769	0.749	0.681	0.766	0.041	5.825
		HDBSCAN	0.410	0.187	0.056	0.440	-0.020	5.756
Encoder-only	MPNet	KMeans (Euclidean)	0.973	0.923	0.924	0.942	0.214	2.451
		KMeans (Spherical)	0.973	0.923	0.924	0.942	0.214	2.451
		Agglomerative	0.788	0.777	0.716	0.797	0.187	2.194
		HDBSCAN	0.484	0.309	0.129	0.452	-0.032	3.126
		Spectral	0.754	0.795	0.627	0.753	0.182	2.308
	Distillbert	KMeans (Euclidean)	0.792	0.718	0.675	0.752	0.257	2.137
		KMeans (Spherical)	0.795	0.730	0.683	0.758	0.256	2.140
		Agglomerative	0.482	0.388	0.156	0.533	0.160	1.645
		HDBSCAN	0.338	0.145	0.047	0.373	0.000	3.479
		Spectral	0.753	0.737	0.562	0.694	0.271	1.894
Decoder-only	Qwen2.5	KMeans (Euclidean)	0.478	0.268	0.189	0.378	0.218	2.178
		KMeans (Spherical)	0.494	0.272	0.203	0.388	0.221	2.172
		Agglomerative	0.389	0.093	0.022	0.469	0.110	1.451
		HDBSCAN	0.427	0.122	0.047	0.433	-0.144	3.854
		Spectral	0.612	0.474	0.284	0.532	0.183	2.109
	GPT-2	KMeans (Euclidean)	0.259	0.032	0.019	0.240	0.005	1.457
		KMeans (Spherical)	0.313	0.059	0.041	0.258	0.259	2.139
		Agglomerative	0.326	0.028	-0.017	0.440	0.206	1.314
		HDBSCAN	0.313	0.033	0.010	0.342	-0.183	3.429
		Spectral	0.307	0.034	0.034	0.262	-0.018	1.375
Encoder-Decoder	Bart	KMeans (Euclidean)	0.303	0.048	0.026	0.259	0.176	2.010
		KMeans (Spherical)	0.309	0.051	0.033	0.261	0.197	2.027
		Agglomerative	0.318	0.028	0.004	0.375	0.235	1.378
		HDBSCAN	0.354	0.060	0.016	0.376	-0.018	1.934
		Spectral	0.419	0.196	0.135	0.364	0.127	2.153
	T5	KMeans (Euclidean)	0.411	0.222	0.148	0.338	0.184	2.676
		KMeans (Spherical)	0.414	0.230	0.152	0.341	0.185	2.672
		Agglomerative	0.357	0.025	-0.002	0.470	0.126	1.774
		HDBSCAN	0.350	0.085	0.038	0.368	0.067	3.939
		Spectral	0.501	0.338	0.196	0.472	0.141	2.189

B Analysis of Model Depth

Table 12: Clustering performance metrics for different Qwen2.5 models (Reuters dataset).

Model	Clustering Algorithm	F1S	NMI	ARI	FMI	Silhouette	Davies-Bouldin
Qwen2.5-0.5B-Instruct	KMeans (Euclidean)	0.1481	0.4503	0.0761	0.2133	0.0960	2.6341
	KMeans (Spherical)	0.1726	0.4556	0.0864	0.2296	0.1063	2.5987
	Agglomerative	0.6135	0.4864	0.5235	0.6470	0.1408	1.4761
	HDBSCAN	0.4262	0.0017	-0.0002	0.4998	0.8344	0.4988
	Spectral	0.2748	0.5000	0.1785	0.3293	0.0308	2.1902
Qwen2.5-1.5B-Instruct	KMeans (Euclidean)	0.1503	0.4391	0.0794	0.2177	0.1009	2.5151
	KMeans (Spherical)	0.1566	0.4511	0.0837	0.2250	0.1238	2.4050
	Agglomerative	0.5913	0.4484	0.5102	0.6332	0.1907	1.3419
	HDBSCAN	0.4294	0.0146	0.0155	0.5008	0.8128	0.4613
	Spectral	0.2874	0.4976	0.1794	0.3321	0.0724	2.2771
Qwen2.5-3B-Instruct	KMeans (Euclidean)	0.1627	0.4528	0.0910	0.2364	0.0985	2.5493
	KMeans (Spherical)	0.1616	0.4582	0.0865	0.2295	0.1079	2.5693
	Agglomerative	0.5627	0.3280	0.2496	0.5024	0.0912	1.3083
	HDBSCAN	0.4273	0.0059	0.0029	0.5005	0.8552	0.4645
	Spectral	0.2563	0.4994	0.1740	0.3303	0.0819	2.2249
Qwen2.5-7B-Instruct	KMeans (Euclidean)	0.1407	0.4332	0.0808	0.2204	0.1053	2.5459
	KMeans (Spherical)	0.1373	0.4193	0.0622	0.1834	0.1810	2.5770
	Agglomerative	0.5561	0.3298	0.3063	0.5192	0.1908	1.2831
	HDBSCAN	0.4349	0.0330	0.0573	0.4975	0.5212	0.6945
	Spectral	0.2578	0.4903	0.1927	0.3515	0.1004	2.2325
Qwen2.5-14B-Instruct	KMeans (Euclidean)	0.1382	0.4059	0.0672	0.1969	0.1249	2.3735
	KMeans (Spherical)	0.1412	0.4153	0.0735	0.2072	0.1375	2.4578
	Agglomerative	0.5646	0.3691	0.3536	0.5415	0.0631	1.3335
	HDBSCAN	0.4291	0.0170	0.0189	0.5018	0.6773	0.6985
	Spectral	0.2626	0.4719	0.1635	0.3152	0.0669	2.3298

C Analysis of Embedding Extraction Layers

Table 13: Analysis of embeddings extracted from different layers of LLMs.

Dataset	Model	Layer	Clustering	F1S	NMI	ARI
Reuters	Qwen2.5-0.5B-Instruct	0	KMeans (Spherical)	0.1583	0.3672	0.0547
			Agglomerative	0.4341	0.0766	0.0295
		6	KMeans (Spherical)	0.1475	0.3666	0.0900
			Agglomerative	0.4381	0.1676	0.1436
		12	KMeans (Spherical)	0.1395	0.3184	0.0858
			Agglomerative	0.4405	0.2103	0.2582
		18	KMeans (Spherical)	0.1341	0.3614	0.0715
			Agglomerative	0.3005	0.3342	0.2317
		24	KMeans (Spherical)	0.1511	0.4509	0.0820
			Agglomerative	0.6135	0.4864	0.5235
	MPNet	0	KMeans (Spherical)	0.1793	0.4404	0.0700
			Agglomerative	0.5739	0.2894	0.2231
		3	KMeans (Spherical)	0.1480	0.4421	0.0725
			Agglomerative	0.5789	0.3583	0.2438
		6	KMeans (Spherical)	0.1287	0.4106	0.0619
			Agglomerative	0.4466	0.3338	0.1719
		9	KMeans (Spherical)	0.1442	0.4321	0.0699
			Agglomerative	0.5760	0.3442	0.2293
		12	KMeans (Spherical)	0.2213	0.5459	0.1028
			Agglomerative	0.7396	0.6824	0.8384
BBC	Qwen2.5-0.5B-Instruct	0	KMeans (Spherical)	0.6692	0.4469	0.4190
			Agglomerative	0.2382	0.0193	-0.0011
		6	KMeans (Spherical)	0.6013	0.3906	0.3111
			Agglomerative	0.3204	0.1070	0.0588
		12	KMeans (Spherical)	0.5627	0.3356	0.2555
			Agglomerative	0.2890	0.1081	0.0377
		18	KMeans (Spherical)	0.8607	0.6996	0.6803
			Agglomerative	0.4072	0.3986	0.2539
		24	KMeans (Spherical)	0.9196	0.7905	0.8113
			Agglomerative	0.4436	0.4665	0.3268
	MPNet	0	KMeans (Spherical)	0.9200	0.7803	0.8139
			Agglomerative	0.2315	0.0097	-0.0005
		3	KMeans (Spherical)	0.9200	0.7976	0.8102
			Agglomerative	0.5825	0.5865	0.3643
		6	KMeans (Spherical)	0.8881	0.7208	0.7402
			Agglomerative	0.2333	0.0079	-0.0001
		9	KMeans (Spherical)	0.9200	0.7834	0.8145
			Agglomerative	0.2328	0.0074	0.0000
		12	KMeans (Spherical)	0.9640	0.8899	0.9154
			Agglomerative	0.6157	0.6724	0.4596