Beyond words: a comparative analysis of LLM embeddings for effective clustering

Imed Keraghel, Stanislas Morbieu, Mohamed Nadif

Beyond words: a comparative analysis of LLM embeddings for effective clustering

Imed Keraghel[1,2], Stanislas Morbieu[2], and Mohamed Nadif[1]

[1] Centre Borelli UMR9010, Université Paris Cité, 75006 Paris, France
[2] Kernix Software, 75014 Paris, France

Abstract. The document clustering process involves the grouping of similar unlabeled textual documents. This task relies on the use of document embedding techniques, which can be derived from various models, including traditional and neural network-based approaches. The emergence of Large Language Models (LLMs) has provided a new method of capturing information from texts through customized numerical representations, potentially enhancing text clustering by identifying subtle semantic connections. The objective of this paper is to demonstrate the impact of LLMs of different sizes on text clustering. To accomplish this, we select five different LLMs and compare them with three less resource-intensive embedding methods. Additionally, we utilize six clustering algorithms. We simultaneously assess the performance of the embedding models and clustering algorithms in terms of clustering quality, and highlight the strengths and limitations of the models under investigation.

Keywords: Large Language Models, Embeddings, Clustering.

1 Introduction

In the rapidly evolving area of natural language processing (NLP), the advent of Large Language Models (LLMs) such as the GPT series [1–4] has significantly enhanced our ability to process and analyze large volumes of texts. These models function by transforming texts into high-dimensional vectors called embeddings, which are commonly used for tasks such as translating or answering questions. Using them for clustering is quite new and has not yet been extensively explored.

The objective of this article is to explore the emerging field of embeddings generated by Large Language Models (LLMs) through a thorough comparative analysis. Our study encompasses various models, including both smaller-scale models and larger architectures like the GPT series. The main goal is to examine the impact of a language model's size on the quality of its embeddings for clustering similar texts.

Clustering, categorizing text based on similarity, is crucial in NLP. LLM-generated embeddings could enhance this by capturing text's semantic nuances. Our core hypothesis posits that there might be a direct correlation between the size of an LLM and its proficiency in creating effective embeddings. This could imply that larger models yield more accurate embeddings, thereby improving clustering results. On the other hand, there is a potential for a point of diminishing returns where increases in model size no longer contribute to significant enhancements in clustering performance. In fact, overly

large models may even detract from clustering quality due to overfitting on training tasks.

This study compares small and large language models to find out if the larger ones create better text groupings or if the smaller ones are equally effective while using fewer resources. This research will help inform future work and real-world uses of these models to sort and understand large textual datasets.

2 Related Work

The use of textual embeddings in NLP tasks has been extensively studied, with a primary focus on supervised tasks such as classification [5]. However, the use of textual embeddings, especially those derived from LLMs, in unsupervised tasks such as clustering remains underexplored. Despite the proven capabilities of LLMs in capturing nuanced semantic relationships within text, their potential for clustering has not been fully exploited or understood within the academic domain.

Previous research, including studies conducted by [6, 7], has focused on investigating the effect of different embedding techniques on clustering results. These studies have shown that the choice of embedding significantly affects the performance of algorithms. Research in this area has mainly focused on comparing traditional embeddings, which are static and contextualized embeddings. The contextualized embeddings are often limited to BERT-based transformer models. The results of these studies have been diverse, with some suggesting that traditional methods perform similarly or even better than newer methods in specific situations.

Recent studies such as [8] acknowledge the strength of LLMs in enhancing clustering but stop short of employing their embeddings directly, opting instead for keyword enrichment strategies. This highlights a gap in the field, indicating that the full capabilities of LLM embeddings have not been entirely harnessed for unsupervised clustering applications. In [9], the authors benchmarked numerous models, including LLMs, for clustering, providing a broad overview of performance but without comparing clustering algorithms or delving into details.

This paper addresses this lacuna by comparing the clustering efficacy of embeddings from variously sized LLMs. Our main objective is to determine whether these advanced embeddings can greatly enhance clustering tasks.

3 Models and algorithms

3.1 Embedding models

Textual embeddings, a fundamental component in NLP for converting texts into numerical vectors, are varied in type and derivation method. Traditional embeddings use simple techniques like `TF-IDF`. Static embeddings, exemplified by `Word2Vec` [10], maintain a consistent vector per word regardless of context. In contrast, contextual embeddings, such as those of `BERT` [11] and `GPT` [3], adjust vectors based on the context of word usage, allowing nuanced language model training. In our contribution, we aim to evaluate the effectiveness of textual embeddings derived from five LLMs, namely `BLOOMZ-560m`,

BLOOMZ-3B [12], Mistral-7B [13], Llama-2-13B [14], and GPT besides a compact transformer-based model, MiniLM [15], which serves as a standard for comparison. BERT and JoSE [16] have also been considered in this study to determine whether the use of energy-intensive models (LLMs) is justified or not. We study this effectiveness in the context of unsupervised learning, particularly in the objective of document clustering.

The properties of the studied models are given in Table 1, with a comprehensive examination presented in the following sections.

Table 1: Description of models.

Model	#Parameters	#Layers	#Embedding size
JoSE	-	-	100
MiniLM-L12-v2	33 million	12	384
BERT	110 million	12	768
BLOOMZ-560m	560 million	24	1024
BLOOMZ-3B	3 billion	30	2560
Mistral	7.3 billion	32	4096
Llama 2	13 billion	40	5120
text-embedding-ada-002	-	-	1536

Mistral 7B is a LLM developed by Mistral AI[3] with 7.3 billion parameters. It outperforms competitors like Llama-2-13B on various evaluation criteria [13]. This model employs innovative attention mechanisms such as Grouped Query Attention (GQA) for faster inferences [17] and Sliding Window Attention (SWA) to handle longer sequences more efficiently [18].

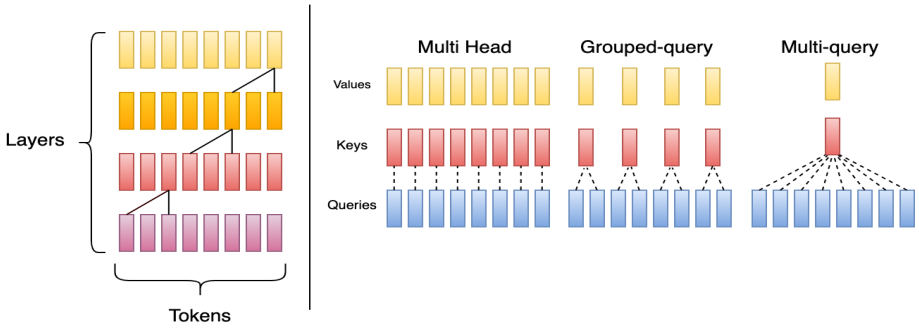

Fig. 1: Visualizing dual attention mechanisms: Grouped-Query and Sliding Window.

The SWA mechanism enhances efficiency by allowing each model layer to concentrate on a data segment within a sliding window. This approach conserves computing resources, scaling with window size and data length. Additionally, SWA aids in retaining and utilizing information beyond the current window through data layering.

The GQA mechanism allows for simultaneous focus on critical text parts. It organizes question words into groups, guiding attention to text areas essential for answering. Per Figure 1, GQA can process several questions at once by forming distinct groups of vectors of the question word. This multigroup approach applies attention to input tokens,

[3] https://mistral.ai/

enabling the model to efficiently address multiple questions by concentrating on relevant text portions simultaneously.

BLOOMZ is an enhanced iteration of the BLOOM model [19]. The original Bloom model was developed through the collaboration of over a thousand AI experts with the aim of creating a freely available and widely accessible LLM. It was trained on a diverse dataset and its architecture is based on the Megatron-LM GPT-2 framework [20].

The BLOOMZ family of models undergoes a process known as instruction tuning or multitask fine-tuning [21]. This process involves the refinement of the pre-trained Bloom model with a diverse datasets across various NLP tasks such as question answering, summarization, and translation. The datasets serve as a means to enhance the model's ability to interpret and act upon instructions encapsulated within prompts, thereby improving its zero-shot task generalization capabilities. In essence, zero-shot learning enables the model to adeptly perform tasks it has not been explicitly trained on, solely based on its understanding derived from the prompts provided.

Llama 2 model, part of Meta's Llama family of open-source LLMs, is an autoregressive language model that uses an optimized transformer architecture and has parameter counts ranging from 7 billion to 70 billion [14]. The largest variant, which incorporates GQA (like Mistral), enhances inference speed without compromising quality. LLama models have exhibited strong competitiveness compared to existing open-source models and have achieved performance levels comparable to some proprietary models. However, it is important to note that they still fall behind more advanced models like GPT-4.

text-embedding-ada-002 is a specialized variant of the GPT-3 architecture developed by OpenAI for generating text embeddings[4]. It uses the transformer framework to create vector representations of text that capture semantic meaning. This model is pre-trained on a vast corpus of text data and subsequently fine-tuned for the specific purpose of text embedding. Although it has fewer parameters than the more extensive GPT-3 models, it retains the capability of the GPT framework, making it accessible to a wide range of applications via API access.

MiniLM is a different kind of language model that emphasizes a smaller and more efficient design compared to LLMs [15]. The main innovation in MiniLM is in its training approach. It involves distilling the knowledge of a powerful, large teacher-language model into a much smaller student model. This process, known as self-attention distillation, involves training the student to imitate the behavior of the teacher as closely as possible. The student model does this by learning to predict the teacher's outputs from its intermediate layers rather than just its final output. Despite its smaller size, MiniLM manages to maintain a high level of language understanding and generation capabilities.

3.2 Clustering algorithms

In unsupervised learning, clustering methods are routinely employed in data embedding to make new discoveries from large and complex data sets. Several clustering algorithms

[4] https://platform.openai.com/docs/guides/embeddings

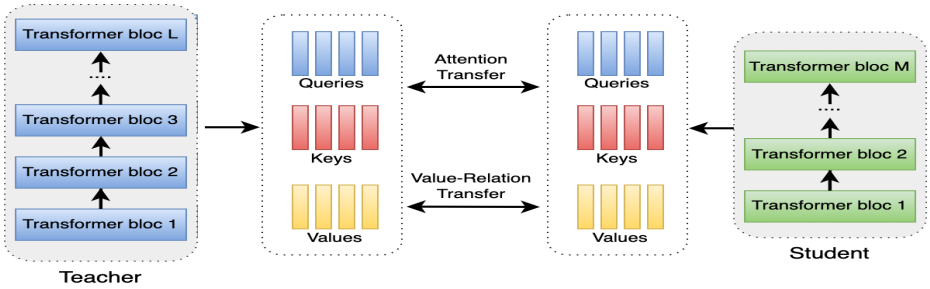

Fig. 2: Deep self-attention distillation: student model training through advanced imitation of teacher model's final layer with enhanced value-relation transfer for self-attention mechanisms, as implemented in MiniLM models.

exist; in our study, we use five algorithms in order to evaluate the different models. These algorithms were chosen for their popularity in the community or for their use when combining them with dimensionality reduction methods. Specifically, we focus on deep clustering variants, which have demonstrated impressive results in image clustering [22]. Therefore, the selected clustering algorithms are the following. `K-means` from Scikit-learn [23], `Spherical K-means` from the coclust framework [24], `CAEclust` from [25] `Deep K-means` and `Deep Clustering Network` from [26].

K-means is a widely used algorithm for clustering. It aims to partition a dataset into k non-overlapping clusters, minimizing the sum of squared distances between data points and their respective cluster centroids. The process involves initializing centroids, assigning points to the nearest centroid, updating centroids based on mean values, and iterating until convergence.

Spherical K-means (SK-means) is a variant of `K-means` designed for data residing on a unit hypersphere. Unlike `K-means` that uses Euclidean distance, Spherical `K-means` employs cosine distance to measure dissimilarity between data points considering the angle between them. This makes it suitable for scenarios where only direction or relative orientation matters, such as text clustering.

A survey [27] describes a wide range of different deep representation learning methods. Here, we focus on the AEs and its variants because combined with clustering methods, they can lead to interesting results in various fields.

CAEclust is a Python package developed to implement consensus clustering [25]. It relies on Autoencoders (AEs) and spectral clustering. It aims to facilitate deep clustering by merging representations from various Denoising Autoencoders (DAEs), as shown in Figure 3, without the need to define a specific architecture, as proposed by the same authors in [22]. `CAEclust` has been successfully evaluated for image datasets. Thus, we plan to test `CAEclust`'s performance on text data.

Representation learning followed by cluster analysis is often helpful in data science. The `K-means` algorithm applied to data embedding derived from `UMAP` [28] or AEs, for example, is a popular approach. This procedure is performed sequentially and is referred

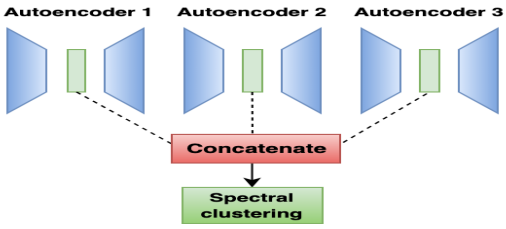

Fig. 3: Overview of CAEclust.

to as the tandem approach. However, AE may sometimes be an unsuitable method to reduce the dimension before clustering; it can fail to retain information that could be valuable for the clustering task. Thus, jointly optimizing for both tasks –RL and clustering– is a good alternative. Thus, we propose two popular algorithms.

Deep K-means (DKM) combines jointly representation learning and clustering [26]. Its aim is to minimize both the reconstruction error and the clustering error in the learned embedded space, by iteratively updating the autoencoder parameters and cluster representatives. Figure 4 shows the steps of this algorithm.

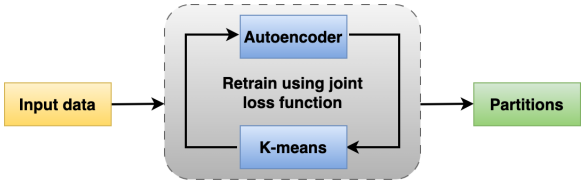

Fig. 4: Overview of methods based on retraining using an AE, `K-means`, and a joint loss function.

Deep Clustering Network (DCN) algorithm [29] also performs unsupervised clustering using a deep autoencoder, but does not use an auxiliary distribution. Instead, it uses an optimization objective that is a direct combination of a reconstruction error and a clustering error, as depicted in Figure 5.

4 Numerical experiments

In this section, we elucidate the various components of our experimental setup.

4.1 Datasets

In the remainder of our experiments, we use the following datasets:

– BBC News[5]: a dataset sourced from the BBC News, encompasses a collection of 2,225 articles labeled across five categories: business, entertainment, politics, sport, and tech.

[5] http://mlg.ucd.ie/datasets/bbc.html

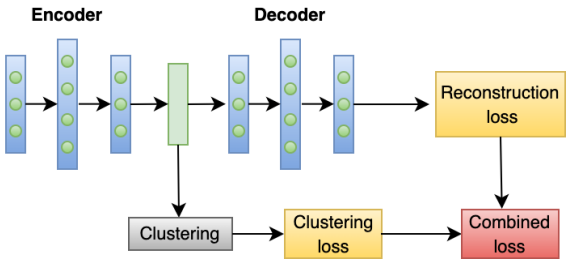

Fig. 5: A typical network architecture for a deep clustering algorithm. Such an algorithm uses an optimization objective that combines the reconstruction error and the clustering error (joint optimization approach).

- 20 Newsgroups[6]: consists of 18,846 newsgroup documents, distributed across 20 different topical newsgroups. Originating from Usenet newsgroups in the late 1990s, it encompasses a broad spectrum of themes such as politics, religion, and science.
- IMDb : a collection of movie reviews retrieved from the IMDb website used for binary sentiment classification [30]. The dataset serves as a cornerstone for NLP research and machine learning studies, designed to provide a substantial and balanced collection of positive and negative reviews.
- Web Content[7]: a compilation of data created through an extensive scraping of various websites. By extracting text from a myriad of web pages, this dataset offers a diverse and rich set of information from different domains and types of content. This dataset comprises a collection of 1,408 samples, meticulously categorized into 16 distinct classifications reflecting a broad spectrum of Web Content. The categories span a variety of sectors and interests, from education and news to e-commerce and sport.

Data preprocessing We select four datasets originating from various fields, each with its own distinct attributes, thus necessitating customized preprocessing strategies. The 20 Newsgroups and IMDb datasets undergo pre-processing, including removing apostrophes, HTML tags, special characters, punctuation, URLs, and converting texts to lowercase. In contrast, the BBC News dataset, which needs less cleaning, undergoes only a conversion to lowercase. The Web Content dataset, already preprocessed, receives no additional treatment. The characteristics of these datasets are detailed in Table 2.

Table 2: Description of datasets. The balance represents the ratio between the smallest and largest class. #Tokens indicates the mean token count.

Datasets	Characteristics			
	#Documents	#Clusters	Balance	#Tokens
BBC News	2,225	5	0.75	390
20 Newsgroup	18,846	20	0.63	284
IMDb	50,000	2	1	231
Web Content	1,408	16	0.14	747

[6] http://qwone.com/ jason/20Newsgroups/
[7] https://www.kaggle.com/datasets/hetulmehta/website-classification

4.2 Evaluation metrics

Using labeled datasets, we evaluate clustering algorithms performance with external indices: Accuracy (ACC), Normalized Mutual Information (NMI) [31], and Adjusted Rand Index (ARI) [32]. The ACC quantifies the degree to which each cluster contains data points corresponding to their respective classes. NMI, which ranges from 0 to 1, evaluates the information commonality between the suggested clustering and the ground truth labels. Finally, ARI measures the similarity between two clustering, taking into account both the cluster assignments and the ground truth labels, when available. The ARI metric ranges from -1 to 1, where a higher value indicates better agreement between the true labels and the clustering. Intuitively, NMI quantifies how much the estimated clustering is informative about the true clustering, while the ARI measures the degree of agreement between the estimated clustering and the reference partition. Both NMI and ARI are equal to 1 if the resulting clustering partition is identical to the ground truth.

4.3 Experimental Settings

The experiments were conducted on a professional workstation with specific hardware specifications: an Intel® Core™ i9-12950HX CPU running at 2.6GHz and 64 GB of DDR5 memory operating at 4,800 MHz. It is important to note that the extraction of embeddings for LLMs required GPU usage, which was carried out on the Pro version of Google Colab equipped with an Nvidia A100 40 GB.

When it comes to clustering algorithms, `K-means` initialize using *the K-means++* technique, which chooses the starting centroids by sampling based on their contribution to total inertia. We cap the iterations at 300 and run 10 initial setups to strengthen the clustering stability. The same parameters, 300 iterations and 10 starts, are applied to `Spherical K-means`. As we are in an unsupervised setting, we employ default values for `Deep K-means`, `DCN`, and `CAEclust` to maintain consistency. The performances of the algorithms are detailed in Table 3.

For each experiment, the model is launched 10 times, and the best result according to the objective function to optimize was selected (Inertia for `KMeans`, combined loss for `DCN`, etc.). For the embeddings, a maximum size of 2,000 tokens was set for all models except for `BERT` and `MiniLM`, where it was fixed at 512. Finally, as baselines, we introduce two other models, namely `BERT` [11] and Joint Spherical Embedding (JoSE) [16]. JoSE has been chosen for its parsimony and efficiency, outperforming popular models such as `Word2Vec`, `GloVe` and `BERT` in tasks of clustering and textual similarity while being simple and energy-efficient. To represent documents, an average of the vectors produced by the embedding models was computed for each token. To access the `GPT-3` embeddings, we use the `text-embedding-ada-002` model through OpenAI's API, which charges $0.0001 for every 1,000 tokens.

4.4 Results and discussion

Below are the essential points that result from numerous experiments:

– Undeniably, the `GPT` model is the best regardless of the nature of the datasets. This is verified regardless of the clustering algorithms used. Furthermore, note that

Table 3: Experimental results: For each dataset, the best score of each clustering algorithm applied with all embedding models, is highlighted in bold, and second-best is underlined. For instance for the Website content dataset, concerning K-means, the best score in term of ACC is **83.85** obtained with GPT and the second 73.08 is obtained with MiniLM.

		Website content			20 Newsgroup			IMDb			BBC News		
Emb.	Clus.	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
JoSE	K-means	72.87	71.56	60.27	56.71	57.05	41.98	68.84	10.57	14.18	95.46	85.55	89.26
	SK-means	**77.41**	73.30	65.12	53.73	55.41	40.59	69.08	11.34	14.55	69.17	66.98	60.81
	UMAP K-means	77.63	76.26	68.00	52.81	61.03	46.16	50.54	00.01	00.01	95.70	86.98	89.99
	DKM	76.28	70.60	62.91	56.73	56.07	42.87	55.96	01.06	01.40	90.52	74.77	77.90
	DCN	66.62	59.60	48.83	56.78	54.57	41.90	61.08	03.69	04.89	81.35	60.68	61.12
	CAEclust	72.80	70.04	60.63	44.68	51.90	33.53	51.36	00.04	00.05	95.60	86.59	89.60
BERT	K-means	41.05	40.08	22.21	34.94	38.64	20.45	54.08	00.51	00.65	94.29	83.43	86.82
	SK-means	42.12	39.26	23.49	35.88	40.05	21.24	54.18	00.54	00.68	93.57	81.78	85.03
	UMAP K-means	54.55	55.52	37.49	50.49	52.02	35.59	55.24	00.83	01.08	93.30	82.53	84.36
	DKM	52.20	48.97	32.73	46.06	50.17	33.28	53.02	00.33	00.35	93.03	81.16	84.22
	DCN	52.06	47.93	30.59	48.73	49.16	33.72	52.44	00.23	00.22	**91.73**	77.70	**80.68**
	CAEclust	54.97	51.28	34.77	44.6	51.88	33.49	54.97	51.28	34.77	92.90	80.50	83.73
MiniLM	K-means	73.08	71.41	60.43	59.34	60.94	45.14	57.28	01.54	02.10	95.87	87.54	90.31
	SK-means	66.48	67.91	55.33	58.09	60.12	43.61	57.30	01.62	02.11	72.09	64.47	60.37
	UMAP K-means	82.60	77.91	72.34	64.56	66.29	50.29	60.52	03.27	04.41	95.64	86.94	89.55
	DKM	77.34	72.30	63.77	61.72	61.06	47.40	58.82	02.26	03.09	91.28	78.41	80.73
	DCN	74.01	67.91	59.24	60.20	59.48	45.89	64.54	06.77	08.44	77.21	53.47	55.16
	CAEclust	74.36	72.42	63.68	60.72	60.68	44.52	60.46	03.19	04.36	95.06	85.71	88.47
BLOOMZ-560M	K-means	24.64	27.43	13.48	24.54	27.90	12.22	60.48	03.20	04.37	89.12	73.84	74.81
	SK-means	30.33	30.55	15.46	25.02	28.53	12.50	60.42	03.16	04.32	89.21	74.04	75.01
	UMAP K-means	56.32	55.34	38.86	34.32	38.84	20.50	64.40	06.07	08.28	94.97	84.82	88.18
	DKM	17.68	12.62	04.20	07.38	00.93	00.28	53.00	00.26	00.34	31.96	04.57	04.03
	DCN	15.77	12.43	04.50	25.64	29.60	13.72	63.74	05.56	07.53	70.56	54.58	47.35
	CAEclust	63.35	65.03	49.55	55.86	56.03	38.70	**86.72**	**45.04**	**53.93**	95.69	87.39	89.82
BLOOMZ-3B	K-means	43.75	40.23	23.38	29.25	33.76	15.00	58.72	02.30	03.02	89.62	74.06	75.72
	SK-means	45.03	41.35	26.38	28.98	33.83	15.13	58.76	02.32	03.05	89.57	73.97	75.61
	UMAP K-means	55.68	53.68	37.41	38.45	43.94	24.33	58.40	02.09	02.80	95.42	86.77	89.38
	DKM	47.94	46.42	28.84	15.14	23.18	07.78	57.12	01.66	02.01	92.27	78.83	81.77
	DCN	39.63	36.43	20.40	34.54	39.72	20.57	58.80	02.33	03.08	89.48	72.32	75.82
	CAEclust	63.42	63.40	48.78	44.30	50.50	31.30	83.72	36.24	45.47	70.88	72.88	62.70
Mistral-7B	K-means	56.46	56.42	41.05	44.53	48.24	26.62	63.04	05.02	06.78	96.36	88.51	91.49
	SK-means	58.10	59.60	43.96	42.98	47.30	26.10	63.00	04.99	06.74	71.15	69.94	64.03
	UMAP K-means	72.94	72.75	61.32	54.75	60.65	41.44	69.08	10.77	14.54	**97.12**	**91.23**	**93.13**
	DKM	65.70	64.85	51.39	56.34	59.34	41.00	64.10	08.41	07.94	87.33	77.94	73.80
	DCN	55.18	54.89	38.11	53.92	55.82	36.23	62.02	04.49	05.76	89.84	77.01	77.23
	CAEclust	59.80	60.56	44.10	40.73	48.64	29.22	77.18	22.59	29.54	94.65	87.19	88.42
Llama-2-13B	K-means	55.33	54.12	36.62	45.60	47.98	27.13	55.82	00.99	01.34	**97.35**	**91.30**	**93.76**
	SK-means	47.73	53.12	37.40	43.39	47.66	25.79	55.76	00.97	01.31	88.63	80.68	76.97
	UMAP K-means	72.16	70.54	58.73	54.18	61.45	42.52	54.30	00.55	00.72	71.37	80.33	66.29
	DKM	70.10	68.17	55.84	56.85	60.69	39.74	52.64	01.15	00.27	88.09	79.78	75.62
	DCN	55.61	56.43	41.24	51.40	56.82	37.39	60.66	03.36	04.53	87.37	**78.23**	74.14
	CAEclust	56.61	58.82	41.56	53.19	55.25	35.00	74.48	18.04	23.96	**97.35**	**92.04**	**93.70**
GPT	K-means	**83.85**	**82.67**	**76.69**	**65.33**	**67.63**	**53.45**	**80.12**	**29.69**	**36.28**	97.08	90.88	93.19
	SK-means	74.34	**79.09**	**67.81**	**64.28**	**68.04**	**52.22**	**80.58**	**30.71**	**37.39**	**96.99**	**90.82**	**92.99**
	UMAP K-means	**84.20**	**83.78**	**78.18**	**70.77**	**72.17**	**60.02**	**71.74**	**14.20**	**18.89**	96.99	90.49	92.79
	DKM	**82.63**	**83.24**	**76.09**	**67.68**	**69.07**	**56.48**	**74.30**	**17.93**	**23.60**	**96.04**	**89.42**	**90.93**
	DCN	**78.98**	**78.32**	**70.09**	**63.93**	**66.84**	**52.58**	**75.54**	**19.76**	**26.08**	88.90	75.37	75.80
	CAEclust	**75.41**	**75.79**	**63.96**	**73.32**	**74.83**	**62.75**	76.66	22.27	28.42	96.67	90.05	92.21

whatever datasets used, with GPT, K-means appears a simple and effective solution for clustering. This was confirmed by a ranking test, which revealed that K-means is the most efficient algorithm on GPT embeddings with an average rank of 2.4. It is followed by UMAP K-means and SK-means, with ranks of 2.9 and 3.2, respectively.

- Unlike the BLOOMZ-560M, BLOOMZ-3B, Mistral-7B and Llama-2-13B models that are sensitive to class imbalance, GPT and also MiniLM remain robust. This can be seen on the Website Content dataset whose classes are very unbalanced, the performance of the clustering methods remains very modest compared to GPT and MiniLM. Furthermore, note that for BLOOMZ-560M, BLOOMZ-3B, Mistral-7B and Llama-2-13B, all clustering algorithms fail, except for CAEclust.
- The performance of the BLOOMZ-560M, BLOOMZ-3B, Mistral-7B and Llama-2-13B models fluctuates depending on the datasets, unlike the GPT model and MiniLM model to a lesser extent. This is probably due to the imbalance and the degree of overlap of the clusters. Indeed, these models seem only suitable when the classes are easily distinguishable, as is the case with BBC News.
- The developers of the Mistral-7B model claim that it compares well to Llama-2-13B, although it has fewer parameters. This claim is substantiated by comparing the performance of clustering algorithms using embeddings from both models.

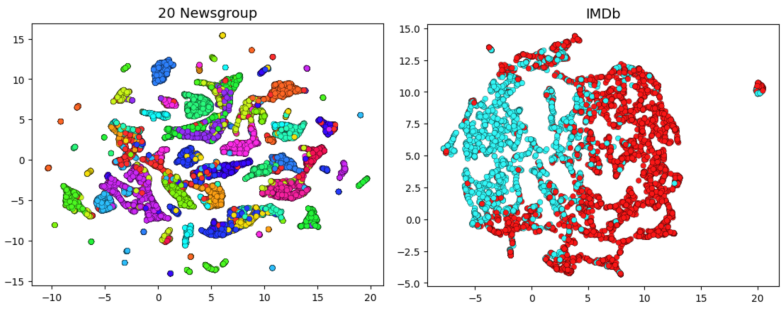

Fig. 6: UMAP: 2D Plot of obtained clusters by K-means using GPT embeddings.

- It does not appear that there is a relationship or correlation between the *size* of the model and the effectiveness of these embeddings. Indeed, it's observed that Mistral-7B exhibits performance comparable to that of Llama-2-13B and that the *smaller* model MiniLM outperforms models that are larger than it.
- One might wonder about the need to use all the dimensions generated by the models. Indeed, this represents a cost which can be reduced by a non-linear dimensionality reduction technique such as UMAP [28]; for all models, we fixed the dimension at 20 and the number of neighbors equal to 50. Thus, we can measure the impact of K-means applied on the reduced matrix. In Table 3 we observe that for all models except GPT and Llama-2-13B, the use of UMAP often leads to improved performance, as illustrated in Figure 6. However, we noted that by increasing the number of neighbors to 90, we can improve the clustering for GPT and Llama-2-13B. Such a suitable number of neighbors was detected based on the within-cluster sum of squares minimized by K-means as a function of the neighbours. In this way, we respect the context of unsupervised learning.

- Note that the performance of the JoSE Model is interesting; it is often better than BERT while embeddings do not require large resources.
- Finally, except for the GPT model, the deep clustering methods DKM, DCN and CAEclust are not always more efficient than a simple K-means.

5 Conclusion and perspectives

In this paper, our objective was to evaluate several LLM models in an unsupervised learning context. We evaluated six clustering algorithms from benchmarks with different characteristics in terms of the degree of mixing (class distribution), the number of classes that can be balanced or not. Unquestionably, the GPT model is the best followed by Mistral-7B and MiniLM; all clustering algorithms record very good scores compared to the other models. Even if our objective is not to compare clustering algorithms, we can note that despite its simplicity, K-means can be used without concern for such embeddings. Furthermore, we noted that the deep CAEclust algorithm also proves interesting on such data. On the other hand, since we chose BERT and JoSE as Baselines models, it is interesting to emphasize the good performance of the JoSE model, which is a parsimonious model.

References

1. Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.
2. Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
3. Brown et al. Language models are few-shot learners. *Neurips*, 33:1877–1901, 2020.
4. OpenAI. Gpt-4 technical report. 2023.
5. Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *18th China National Conference*, pages 194–206. Springer, 2019.
6. Rohan Saha. Influence of various text embeddings on clustering performance in nlp. *arXiv preprint arXiv:2305.03144*, 2023.
7. Jayasree Ravi and Sushil Kulkarni. Text embedding techniques for efficient clustering of twitter data. *Evolutionary Intelligence*, pages 1–11, 2023.
8. Vijay Viswanathan, Kiril Gashteovski, Carolin Lawrence, Tongshuang Wu, and Graham Neubig. Large language models enable few-shot clustering. *arXiv:2307.00524*, 2023.
9. Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022.
10. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv:1301.3781*, 2013.
11. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2018.
12. Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. Crosslingual generalization through multitask finetuning. *arXiv:2211.01786*, 2022.
13. Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv:2310.06825*, 2023.

14. Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv:2307.09288*, 2023.
15. Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Neurips*, 33:5776–5788, 2020.
16. Yu Meng, Jiaxin Huang, Guangyuan Wang, Chao Zhang, Honglei Zhuang, Lance Kaplan, and Jiawei Han. Spherical text embedding. *Neurips*, 32, 2019.
17. Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv:2305.13245*, 2023.
18. Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.
19. Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv:2211.05100*, 2022.
20. Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv:1909.08053*, 2019.
21. Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*, 2022.
22. Séverine Affeldt, Lazhar Labiod, and Mohamed Nadif. Spectral clustering via ensemble deep autoencoder learning (SC-EDAE). *Pattern Recognition*, 108:107522, 2020.
23. Buitinck et al. Api design for machine learning software: experiences from the scikit-learn project. *arXiv:1309.0238*, 2013.
24. François Role, Stanislas Morbieu, and Mohamed Nadif. Coclust: A python package for co-clustering. *Journal of Statistical Software*, 88(7):1–29, 2019.
25. Séverine Affeldt, Lazhar Labiod, and Mohamed Nadif. Caeclust: A consensus of autoencoders representations for clustering. *Image Processing On Line*, 12:590–603, 2022.
26. Maziar Moradi Fard, Thibaut Thonet, and Eric Gaussier. Deep k-means: Jointly clustering with k-means and learning representations. *Pattern Recognition Letters*, 138:185–192, 2020.
27. Md Rezaul Karim, Oya Beyan, Achille Zappa, Ivan G Costa, Dietrich Rebholz-Schuhmann, Michael Cochez, and Stefan Decker. Deep learning-based clustering approaches for bioinformatics. *Briefings in Bioinformatics*, pages 1–23, 2020.
28. Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv:1802.03426*, 2018.
29. Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *ICML*, pages 3861–3870, 2017.
30. Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.
31. Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.
32. Douglas Steinley. Properties of the hubert-arable adjusted rand index. *Psychological methods*, 9(3):386, 2004.