# Text Clustering as Classification with LLMs

**Anonymous ACL submission**

## Abstract

Text clustering remains valuable in real-world applications where manual labeling is cost-prohibitive. It facilitates efficient organization and analysis of information by grouping similar texts based on their representations. However, implementing this approach necessitates fine-tuned embedders for downstream data and sophisticated similarity metrics. To address this issue, this study presents a novel framework for text clustering that effectively leverages the in-context learning capacity of Large Language Models (LLMs). Instead of fine-tuning embedders, we propose to transform the text clustering into a classification task via LLM. First, we prompt LLM to generate potential labels for a given dataset. Second, after integrating similar labels generated by the LLM, we prompt the LLM to assign the most appropriate label to each sample in the dataset. Our framework has been experimentally proven to achieve comparable or superior performance to state-of-the-art clustering methods that employ embeddings, without requiring complex fine-tuning or clustering algorithms. We make our code available to the public for utilization[1].

## 1 Introduction

Text clustering is a fundamental task within the realm of natural language processing (NLP) and holds significant importance in various practical situations, especially when manual annotation is prohibitively costly. In particular, it identifies and categorizes texts that share common themes or topics based on their content similarity and plays a crucial role in improving community detection results in social media (Qi et al., 2012), identifying new topics (Castellanos et al., 2017), analyzing extensive text datasets (Aggarwal and Zhai, 2012), structuring information (Cutting et al., 2017), and organizing documents to improve retrieval results (Anick and Vaithyanathan, 1997; Cutting et al., 1993). A typical method for text clustering is to apply clustering algorithms based on pre-trained embeddings (Devlin et al., 2018; Muennighoff et al., 2023a; Wang et al., 2022; Su et al., 2022). The embedding captures semantic relationships between words and phrases, providing a dense and continuous representation of text that is well-suited for clustering tasks. However, these methodologies often require tailored fine-tuning to adapt to specific domains or datasets, which can be resource-intensive and time-consuming. Besides, the choice of clustering hyperparameters is greatly influenced by human expertise and can have a significant impact on the final outcomes.

Recent state-of-the-art LLMs, such as the GPT series (Brown et al., 2020; Ouyang et al., 2022; OpenAI, 2023), have showcased remarkable reasoning performance across a wide range of NLP tasks. However, GPT models restrict access to their outputs solely through an API, rather than permitting the fine-tuning of parameters to customize the embeddings for specific downstream tasks. This limitation of closed-source LLMs has prevented them from realizing their full potential in previous clustering methodologies. While several studies have attempted to harness the outputs of API-based LLMs to guide text clustering (Zhang et al., 2023a; Wang et al., 2023a; Viswanathan et al., 2024), they still rely on skeleton support from fine-tuning embedders such as BERT and E5, as well as improving clustering algorithms like K-means[2].

To this end, this study proposes a novel two-stage framework that transforms the text clustering task into a classification task by leveraging the formidable capabilities of the LLM. In Stage 1, we sequentially input the data in mini-batches and prompt the LLM with a label generation task to assign potential labels to the given data. In Stage

[1] https://anonymous.4open.science/r/Text-Clustering-via-LLM-E500

[2] See Appendix A for more discussions on related work

2, after obtaining the labels, we prompt the LLM to classify the given data based on these labels. This framework processes the dataset sequentially, rather than all at once, thereby circumventing the input length limitations of LLMs. Moreover, it leverages the exceptional generation and classification capabilities of LLM to effectively simplify the complexities of clustering and enhance overall clustering performance.

We extensively evaluate our framework on five datasets encompassing diverse tasks such as topic mining, emotion detection, intent discovery, and domain discovery, with granularities ranging from 18 to 102 clusters. Our results demonstrate that the proposed framework achieves comparable and even better outcomes compared with state-of-the-art clustering methods that employ tailored embedders or cluster algorithms. Notably, our approach eliminates the need for a fine-tuning process for different datasets as well as tricky hyper-parameter settings, thereby saving significant time and computational resources.

## 2 Methodology

In this work, we propose a two-stage framework that utilizes a single LLM for text clustering tasks. To better leverage the generative and classification capabilities of LLMs, we transform the clustering task into a label-based classification task, allowing the LLM to process the data more effectively. As illustrated in Figure 1, unlike previous text clustering methods such as ClusterLLM (Zhang et al., 2023b) that calculate distances between data points in vector space, our approach does not require fine-tuning for better representation or a pre-assigned cluster number $K$. We first prompt the LLM to generate potential labels for the data. After merging similar labels, we then prompt the LLM to classify the input data based on these generated labels. We will introduce our method in detail in the following sections.

### 2.1 Task Definition

For text clustering, given an unlabeled dataset $\mathcal{D} = \{d_i\}_{i=1}^{N}$, where $N$ is the size of the corpus, the goal is to output $K$ subsets of $\mathcal{D}$ as $\mathcal{C} = \{c_j\}_{j=1}^{K}$, where $K$ represents the number of clusters and each $c_j$ represents a cluster, such that $d_1 \in c_j$ and $d_2 \in c_j$ if $d_1$ and $d_2$ belongs to the same cluster. We transform text clustering task into classification task in this work. Specifically, given the dataset

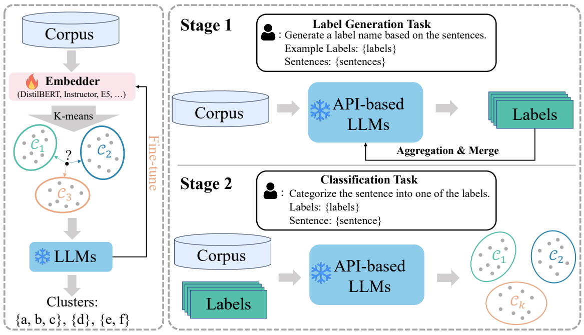


Figure 1: A comparison between other methods using LLMs (left) and our method (right) for text clustering. Our method transforms the clustering task into a text classification task by generating potential labels (Stage 1) and classifying input sentences according to the labels (Stage 2) using LLMs.

$\mathcal{D}$, the model first generates a set of labels $\mathcal{L} = \{l_k\}_{k=1}^{K'}$ based on the content of the dataset, where $K'$ is the number of labels. Subsequently, each data $d_i \in \mathcal{D}$ will be classified into one of the labels $l \in \mathcal{L}$ and the input dataset will be clustered into $K'$ clusters $\mathcal{C}' = \{c'_j\}_{j=1}^{K'}$.

### 2.2 Label Generation Using LLMs

In this section, we explore the process of forming a label-generation task to obtain potential labels for clusters using LLMs. Given the few-shot capabilities of LLMs (Brown et al., 2020), we will provide several example label names to fully utilize the in-context learning ability of LLMs.

#### 2.2.1 Potential Label Generation

Since inputting an entire dataset into LLMs is impractical due to context length limitations, we input the dataset in mini-batches and then aggregate the potential labels. Subsequently, we prompt the model to merge similar labels to adjust the granularity of the clusters. Specifically, given a batch size $B$, we will first prompt the LLM with $B$ instances along with $n$ example label names to generate potential labels for the input data using a prompt $\mathcal{P}_g$, where the dataset is divided into $\frac{N}{B}$ mini-batches for processing:

$$\mathcal{L}' = \mathcal{P}_g(\mathcal{I}_{\text{generate}}, \mathcal{D}', l) \tag{1}$$

where $\mathcal{I}_{\text{generate}}$ is the label generation task instruction, $\mathcal{D}' = \{d_i\}_{i=1}^{B}$ is the input data in mini-batches of the size $B$, and $l$ represents the $n$ given label names.

#### 2.2.2 Potential Labels Aggregation and Mergence

After obtaining all the potential labels from LLMs, we aggregate the labels generated from each mini-batch together:

$$\mathcal{L}_{\text{unique}} = \{l \mid l \in \mathcal{L}'\} \tag{2}$$

To avoid redundant duplication of final clusters caused by the LLM producing different descriptions for the same label, we further prompt the LLM to merge labels with similar expressions:

$$\mathcal{L} = \mathcal{P}_m(\mathcal{I}_{\text{merge}}, \mathcal{L}_{\text{unique}}) \tag{3}$$

where $\mathcal{I}_{\text{merge}}$ is the instructions of the merging task.

### 2.3 Given Label Classification

Given the potential labels for the entire dataset, we can now obtain the final clusters by performing label classification using LLMs. For each input instance, we prompt the LLM to assign a label from the previously generated potential labels:

$$c'_j = \mathcal{P}_a(\mathcal{I}_{\text{assign}}, d_j, \mathcal{L}) \tag{4}$$

where $c'_j$ is the cluster that the LLM classifies $d_j$ into and $\mathcal{I}_{\text{assign}}$ is the instruction of the assigning task. After assigning all the data in the dataset according to the labels, we finally get the text clustering result $\mathcal{C}' = \{c'_j\}_{j=1}^{K'}$.

For the detailed prompt template and instructions $\mathcal{I}_{\text{generate}}$, $\mathcal{I}_{\text{merge}}$, and $\mathcal{I}_{\text{assign}}$, please refer to Appendix B.

## 3 Experiment

Following Zhang et al.(2023b), we evaluate our method on five datasets covering diverse tasks with different granularities. See Appendix C for dataset details.

### 3.1 Implementation Details

#### 3.1.1 Query LLMs

We use GPT-3.5-turbo as the query LLM for label generation and given label classification. Responses are controlled by adding a postfix: "Please response in *JSON* format". Detailed prompts and instructions are provided in Appendix B. We then extract the labels from the list in the JSON response.

#### 3.1.2 Potential Label Generation

During label generation, label names are provided to the LLM as examples. We set the number of given label names to 20% of the total number of labels in the dataset. To account for context length limitations, we set the mini-batch size $B$ to 15, meaning the LLM receives 15 input sentences at a time to generate potential labels.

#### 3.1.3 Evaluation Metrics

Following (De Raedt et al., 2023) and (Zhang et al., 2023b), we use three metrics to evaluate clustering quality. The first metric is Accuracy (ACC), calculated by aligning true labels and predicted clusters using the Hungarian algorithm (Kuhn, 1955) and calculating the percentage of correct assignments. Second metric is Normalized Mutual Information (NMI), which uses mutual information to measure the similarity between the true and predicted clusters and normalize it by the average of the entropy. Lastly, we use Adjusted Rand Index (ARI), which takes into account the possibility of random cluster assignments by adjusting the Rand Index for the chance grouping of elements.

### 3.2 Compared Baselines

**K-means.** We directly apply K-means on embeddings extracted from E5-large (Wang et al., 2022) and Instructor-large (Su et al., 2022). We run the clustering 5 times with different seeds and calculate the average result.

**IDAS** (De Raedt et al., 2023) identifies prototypes that represent the latent intents and independently summarizes them into labels using LLMs. Then, it encodes the concatenation of sentences and summaries for clustering.

**PAS** (Wang et al., 2023b) develops a three-stage algorithm Propose-Assign-Select by prompting LLMs to generate goal-related explanations, determine whether each explanation supports each sample, and use integer linear programming to select clusters such that each sample belongs to one single cluster.

**ClusterLLM** (Zhang et al., 2023b) prompts LLM for insights on similar data points and fine-tunes small embedders using the LLM's choice. It also uses LLM to guide the clustering granularity by determining whether two data points belong to the same category. For comparison, we select the best performing model ClusterLLM-I-iter reported in the paper.

Additionally, we apply our method with gold labels given, which performs label classification using the dataset's ground truth cluster labels. This model represents the upper bound of the LLM's performance. For more implementation details on the baselines, please refer to Appendix D.

| Methods | ArxivS2S ACC | ArxivS2S NMI | ArxivS2S ARI | GoEmo ACC | GoEmo NMI | GoEmo ARI | Massive-I ACC | Massive-I NMI | Massive-I ARI | Massive-D ACC | Massive-D NMI | Massive-D ARI | MTOP-I ACC | MTOP-I NMI | MTOP-I ARI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **K-means (E5)** | 31.21 | 54.47 | 17.01 | 22.14 | 21.26 | 9.64 | 52.79 | 70.76 | 39.03 | 62.21 | 65.42 | 47.69 | 34.48 | 71.47 | 26.35 |
| **K-means (Instructor)** | 25.11 | 48.48 | 12.39 | 25.19 | 21.54 | 17.03 | 56.55 | 74.49 | 42.88 | 60.41 | 67.31 | 43.90 | 33.04 | 71.46 | 26.72 |
| **IDAS** | 16.79 | 41.56 | 6.68 | 15.24 | 12.00 | 5.43 | 51.33 | 68.38 | 38.29 | 54.65 | 57.32 | 42.49 | 33.91 | 68.70 | 27.90 |
| **PAS** | 36.50 | 16.37 | 18.15 | 11.34 | 2.84 | 10.14 | 19.62 | 28.99 | 9.56 | 40.63 | 30.99 | 22.80 | 50.88 | 64.88 | 41.83 |
| **ClusterLLM** | 26.34 | 50.45 | 13.65 | 26.75 | 23.89 | **17.76** | 60.69 | 77.64 | 46.15 | 60.85 | **68.67** | 45.07 | 35.04 | 73.83 | 29.04 |
| **Ours** | **38.78** | **57.43** | **20.55** | **31.66** | **27.39** | 13.50 | **71.75** | **78.00** | **56.86** | **64.12** | 65.44 | **48.92** | **72.18** | **78.78** | **71.93** |
| **LLM_known_labels** | 41.50 | 57.59 | 20.67 | 38.97 | 28.85 | 18.94 | 75.25 | 78.19 | 58.01 | 69.77 | 69.27 | 55.26 | 73.25 | 80.88 | 73.93 |

Table 1: Experiment results of text clustering on five datasets, evaluated using Accuracy, NMI and ARI. Best results are highlighted in bold. *LLM_known_labels* represents the theoretical upper bound for LLMs in this task. Results of t-test has shown significant improvements of our method.

# 4 Results

## 4.1 Text Clustering Results

We present our text clustering results in Table 1. Firstly, our method consistently improves text clustering results over other baseline methods across all datasets, with very few exceptions. For instance, our method increases accuracy by 12.44% on ArxivS2S, and MTOP-I even witnesses a performance doubling. This demonstrates the effectiveness of using LLMs exclusively in text clustering. Besides, the improvements across three different evaluation metrics indicate that our method comprehensively enhances text clustering results from different aspects. It not only effectively identifies and differentiates between distinct categories but also captures the intrinsic structures and characteristics of the data points. What's more, the performance of our method is close to that of the upper bond *LLM_known_labels*, which uses ground truth cluster labels for classification. This comparable performance shows the effectiveness of our approach in generating potential labels and merging similar labels to determine cluster granularity.

## 4.2 Granularity Analysis

To assess the granularity of the output clusters, we compare the final cluster number generated by our method with those produced by ClusterLLM. Table 2 shows that our method outputs cluster counts that are closer to the true number of clusters, indicated by a smaller absolute difference. This closer alignment with the actual cluster distribution highlights our method's ability in more accurately capturing the underlying structure of the data through merging labels that have similar semantic meanings. Consequently, this leads to improved cluster coherence and validity. The ablation test regarding label merging task in Appendix F supports this conclusion. It compares the cluster granularity before and after the merging task and shows that performing label merging task can help the model aggregate similar labels and output a cluster number that is closer to the ground truth.

| Method | ArxivS2S | GoEmo | Massive-I | Massive-D | MTOP-I |
|---|---|---|---|---|---|
| GT #clusters | 93 | 27 | 59 | 18 | 102 |
| ClusterLLM | 16 (-77) | 56 (+29) | 43 (-16) | 90 (+72) | 43 (-59) |
| Ours | 122 (+29) | 52 (+25) | 71 (+12) | 24 (+6) | 83 (-19) |

Table 2: Granularity analysis. The results are presented in the format of "[#clusters](difference)", where a positive difference means the model generate more #clusters than ground truth and vice versa.

## 4.3 Analysis on Few-shot Label Generation

To demonstrate that using few-shot examples can help LLM improve its performance, we conduct experiments with different percentage of gold labels given to the LLM. As shown in Appendix E, when provided with examples, the model improves its clustering result across all three evaluation metrics on all five datasets. This observation supports our method of providing example label names during label generation task in Section 2.2.1 and shows that our method can utilize the in-context learning ability of LLMs.

# 5 Conclusion

We explore using LLMs exclusively for text clustering without relying on additional embedders or traditional cluter algorithms by proposing a two-stage framework that transforms the text clustering problem into label generation and classification tasks. This framework adds interpretability of the clusters by assigning meaningful labels. Additionally, LLMs' comprehensive knowledge from pre-training data enhances domain adaption ability of our method in text clustering. Extensive experiments demonstrate the effectiveness of our framework in text clustering with better performance and granularity. We will explore more cost-efficient and fine-grained methods in text clustering using LLM in the future.

## Limitation

Our work has limitations in the following senses. First, as our work relies exclusively on LLMs for text clustering and does not fine-tune smaller embedders for better representation, more processing is required through LLMs. This results in increased API usage and higher associated costs. Since we use the LLM for given-label classification, the number of API calls is proportional to the dataset size. While the savings in computational costs can offset a significant portion of this API cost increase, this remains a cost limitation when dealing with large datasets. Second, while our method achieves better granularity in clustering results compared to other LLM-based methods like ClusterLLM, it still lacks fine-grain control. Our approach depends on LLMs to generate potential labels and merge similar concepts, making the final output labels heavily reliant on the LLMs' aggregation results. As a result, we only have general control over the granularity of the clusters. Given that the number of clusters can significantly influence the clustering outcomes, we aim to develop more precise mechanisms for controlling granularity in future work.

## Ethics Statement

This work employs LLMs through APIs (e.g. OpenAI API), it will be risky and unsafe to upload privacy information. Additional effort should be applied to remove sensitive information before uploading to LLMs if you are using this framework to deal with sensitive data.

## References

Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text clustering algorithms. *Mining text data*, pages 77–128.

Peter G Anick and Shivakumar Vaithyanathan. 1997. Exploiting clustering and phrases for context-based information retrieval. In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 314–323.

Florian Beil, Martin Ester, and Xiaowei Xu. 2002. Frequent term-based text clustering. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 436–442.

Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. 2020. Structural deep clustering network. In *Proceedings of the web conference 2020*, pages 1400–1410.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

David Carmel, Haggai Roitman, and Naama Zwerdling. 2009. Enhancing cluster labeling using wikipedia. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 139–146.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. 2018. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149.

Ángel Castellanos, Juan Cigarrán, and Ana García-Serrano. 2017. Formal concept analysis for topic detection: a clustering quality experimental analysis. *Information Systems*, 66:24–42.

Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. 2017. Deep adaptive image clustering. In *Proceedings of the IEEE international conference on computer vision*, pages 5879–5887.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan Boyd-Graber, and David Blei. 2009. Reading tea leaves: How humans interpret topic models. *Advances in neural information processing systems*, 22.

Douglass R Cutting, David R Karger, and Jan O Pedersen. 1993. Constant interaction-time scatter/gather browsing of very large document collections. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 126–134.

Douglass R Cutting, David R Karger, Jan O Pedersen, and John W Tukey. 2017. Scatter/gather: A cluster-based approach to browsing large document collections. In *ACM SIGIR Forum*, volume 51, pages 148–159. ACM New York, NY, USA.

Maarten De Raedt, Fréderic Godin, Thomas Demeester, Chris Develder, and Sinch Chatlayer. 2023. Idas: Intent discovery with abstractive summarization. In *The 5th Workshop on NLP for Conversational AI*, page 71.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. 2020. Goemotions: A dataset of fine-grained emotions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4040–4054.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, et al. 2023. Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Renchu Guan, Hao Zhang, Yanchun Liang, Fausto Giunchiglia, Lan Huang, and Xiaoyue Feng. 2020. Deep feature-based text clustering and its explanation. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3669–3680.

Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. 2017. Deep clustering with convolutional autoencoders. In *Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part II 24*, pages 373–382. Springer.

Peihao Huang, Yan Huang, Wei Wang, and Liang Wang. 2014. Deep embedding network for clustering. In *2014 22nd International conference on pattern recognition*, pages 1532–1537. IEEE.

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

Sehyun Kwon, Jaeseung Park, Minkyu Kim, Jaewoong Cho, Ernest K Ryu, and Kangwook Lee. 2023. Image clustering conditioned on text criteria. *arXiv preprint arXiv:2310.18297*.

Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. 2021. Mtop: A comprehensive multilingual task-oriented semantic parsing benchmark. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2950–2962.

Stuart Lloyd. 1982. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023a. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023b. Mteb: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037.

Roberto Navigli and Giuseppe Crisafulli. 2010. Inducing word senses to improve web search result clustering. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 116–126.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Sungwon Park, Sungwon Han, Sundong Kim, Danu Kim, Sungkyu Park, Seunghoon Hong, and Meeyoung Cha. 2021. Improving unsupervised image clustering with robust learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12278–12287.

Guo-Jun Qi, Charu C Aggarwal, and Thomas Huang. 2012. Community detection with edge content in social media networks. In *2012 IEEE 28th International conference on data engineering*, pages 534–545. IEEE.

Yazhou Ren, Ni Wang, Mingxia Li, and Zenglin Xu. 2020. Deep density-based image clustering. *Knowledge-Based Systems*, 197:105841.

Satu Elisa Schaeffer. 2007. Graph clustering. *Computer science review*, 1(1):27–64.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A Smith, Luke Zettlemoyer, and Tao Yu. 2022. One embedder, any task: Instruction-finetuned text embeddings. *arXiv preprint arXiv:2212.09741*.

Yaling Tao, Kentaro Takagi, and Kouta Nakata. 2021. Clustering-friendly representation learning via instance discrimination and feature decorrelation. *arXiv preprint arXiv:2106.00131*.

Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. 2014. Learning deep representations for graph clustering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28.

Pucktada Treeratpituk and Jamie Callan. 2006. Automatically labeling hierarchical clusters. In *Proceedings of the 2006 international conference on Digital government research*, pages 167–176.

Vijay Viswanathan, Kiril Gashteovski, Kiril Gashteovski, Carolin Lawrence, Tongshuang Wu, and Graham Neubig. 2024. Large language models enable few-shot clustering. *Transactions of the Association for Computational Linguistics*, 12:321–333.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Zihan Wang, Jingbo Shang, and Ruiqi Zhong. 2023a. Goal-driven explainable clustering via language descriptions. In *Proceedings of the 2023 Conference* *on Empirical Methods in Natural Language Processing*, pages 10626–10649, Singapore. Association for Computational Linguistics.

Zihan Wang, Jingbo Shang, and Ruiqi Zhong. 2023b. Goal-driven explainable clustering via language descriptions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10626–10649.

Jianlong Wu, Keyu Long, Fei Wang, Chen Qian, Cheng Li, Zhouchen Lin, and Hongbin Zha. 2019. Deep comprehensive correlation mining for image clustering. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8150–8159.

Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. 2015. Short text clustering via convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 62–69.

Jianwei Yang, Devi Parikh, and Dhruv Batra. 2016. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5147–5156.

Yi Yang, Dong Xu, Feiping Nie, Shuicheng Yan, and Yueting Zhuang. 2010. Image clustering using local discriminant models and global integration. *IEEE Transactions on Image Processing*, 19(10):2761–2773.

Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. 2017. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 555–564.

Chao Zhang, Fangbo Tao, Xiusi Chen, Jiaming Shen, Meng Jiang, Brian Sadler, Michelle Vanni, and Jiawei Han. 2018. Taxogen: Unsupervised topic taxonomy construction by adaptive term embedding and clustering. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2701–2709.

Yuwei Zhang, Zihan Wang, and Jingbo Shang. 2023a. ClusterLLM: Large language models as a guide for text clustering. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13903–13920, Singapore. Association for Computational Linguistics.

Yuwei Zhang, Zihan Wang, and Jingbo Shang. 2023b. Clusterllm: Large language models as a guide for text clustering. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13903–13920.

Sheng Zhou, Hongjia Xu, Zhuonan Zheng, Jiawei Chen, Jiajun Bu, Jia Wu, Xin Wang, Wenwu Zhu, Martin Ester, et al. 2022. A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions. *arXiv preprint arXiv:2206.07579*.

Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2009. Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment*, 2(1):718–729.

# Appendix

## A Related Work

### A.1 Clustering

Clustering as a fundamental task in machine learning, has been applied to various data types, including texts (Beil et al., 2002; Aggarwal and Zhai, 2012; Xu et al., 2015), images (Yang et al., 2010; Chang et al., 2017; Wu et al., 2019; Ren et al., 2020; Park et al., 2021), and graphs (Schaeffer, 2007; Zhou et al., 2009; Tian et al., 2014; Yin et al., 2017). Recent studies paid much attention to utilizing deep neural networks in clustering, which models the similarity among instances using learned representations (Huang et al., 2014; Guo et al., 2017; Bo et al., 2020; Zhou et al., 2022). For example, Yang et al. (2016) propose a recurrent network for joint unsupervised learning of deep representations in clustering. Caron et al. (2018) jointly learns the parameters of neural networks and the cluster assignments of the resulting features. Tao et al. (2021) combines instance discrimination and feature decorrelation to present a clustering-friendly representation learning method. All these methods require additional training process to obtain the featured representations, and then apply standard clustering algorithms, such as K-means (Lloyd, 1982), to obtain the final cluster results (Guan et al., 2020). We argue that this kind of method has limited data adaption ability and has to train the model on new datasets, which result in high computational cost.

### A.2 Adding Explanations to Text Clusters

While previous clustering algorithms do not necessarily produce interpretable clusters (Chang et al., 2009), studies pay attention to explaining the clusters with semantically meaningful expressions. Treeratpituk and Callan (2006) assigns labels to hierarchical clusters and assesses potential labels by utilizing information from the cluster itself, its parent cluster, and corpus statistics; Carmel et al. (2009) proposes a framework that selects candidate labels from external resources like Wikipedia to represent the content of the cluster; Navigli and Crisafulli (2010) induce word senses when clustering the result based on their semantic similarity; Zhang et al. (2018) iteratively identifies general terms and refines the sub-topics during clustering to split coarse topics into fine-grained ones. However, label or phrase level added information is limited in describing a complex cluster (Wang et al., 2023b). Thus, more in-depth expressions are required to make clusters more explainable.

### A.3 Text Clustering using LLMs

Recent rapid development of Large Language Models (LLMs), such as GPT series (Brown et al., 2020; Ouyang et al., 2022; OpenAI, 2023), has demonstrated the powerful comprehensive language capability of LLMs and some works has been using LLMs in text clustering task. Wang et al. (2023b) utilze LLMs to propose explanations for the cluster and classify the samples based on the generated explanations; De Raedt et al. (2023) collects descriptive utterance labels from LLMs with well-chosen prototypical utterances to bootstrap in-context learning; Kwon et al. (2023) use LLMs to label the description of input data and cluster the labels with given K. Besides explanation and label generation, Viswanathan et al. (2024) expand documents' keyphrases, generate pairwise constraints and correct low-confidence points in the clusters via LLMs, Zhang et al. (2023b) leverage feedbacks from LLMs to improve smaller embedders, such as Instructor (Su et al., 2022) and E5 (Wang et al., 2022), and prompt LLMs for helps on clustering granularity. All these methods use LLMs in an indirect way that LLMs only process part of the input data and do not see the whole dataset. We argue that this approach does not take full advantage of the powerful linguistic capabilities of LLMs.

## B Prompt template

We design different prompt template ($\mathcal{P}_g$, $\mathcal{P}_m$, $\mathcal{P}_a$) and instructions ($\mathcal{I}_{generate}$, $\mathcal{I}_{merge}$, $\mathcal{I}_{assign}$) for label generation, aggregating & merging labels and given label classification tasks. Table 3 demonstrates the prompt template and the instructions used in each task. In order to get better response from LLMs for further data process, we add format control related prompt into the instructions, such as "Please return in json format" with a json example for LLMs to better understand how to response in a better way. We present a case study in MTOP-I dataset for each task in Table 4.

## C Dataset Description

We extensively evaluate our framework on five datasets encompassing diverse tasks, including topic mining, emotion detection, intent discovery and domain discovery. Each dataset has different granularities, ranging from 18 to 102 clusters.

| Task | Prompt template with instruction $\mathcal{I}$ |
|---|---|
| Label Generation<br>$\mathcal{P}_g$ | Given the labels, under a text classification scenario, can all these text match the label given?<br>If the sentence does not match any of the label, please generate a meaningful new label name.<br>Labels: {given_labels}<br>Sentences: {sentence_list}<br>You should NOT return meaningless label names such as 'new_label_1' or 'unknown_topic_1' and<br>only return the new label names, please return in json format like: {json_example} |
| Aggregating and merging labels<br>$\mathcal{P}_m$ | Please analyze the provided list of labels to identify entries that are similar or duplicate, considering<br>synonyms, variations in phrasing, and closely related terms that essentially refer to the same concept.<br>Your task is to merge these similar entries into a single representative label for each unique concept<br>identified. The goal is to simplify the list by reducing redundancies without organizing it into subcategories<br>or altering its fundamental structure. Here is the list of labels for analysis and simplification:{label_list}.<br>Produce the final, simplified list in a flat, JSON-formatted structure without any substructures or<br>hierarchical categorization like: {json_example} |
| Given label classification<br>$\mathcal{P}_a$ | Given the label list and the sentence, please categorize the sentence into one of the labels.<br>Label list: {label_list}<br>Sentence: {sentence}<br>You should only return the label name, please return in json format like: {json_example} |

Table 3: Prompt template and instructions used in this paper. In this template, words inside { } should be replaced by corresponding variables during experiments.

| Task | Question and Answer | |
|---|---|---|
| Label Generation | Prompt | Given the labels, under a text classification scenario, can all these text match the label given?<br>If the sentence does not match any of the label, please generate a meaningful new label name.<br>Labels: [ 'update call', 'get weather', 'update method call', ... , 'resume music', 'get stories<br>news', 'get reminder location' ]<br>Sentences: ['do not play any song by lil Yachty', 'show me video messages from my grandma', ... ,<br>'please add a one and a half minute timer', 'read national news headlines']<br>You should NOT return meaningless label names such as 'new_label_1' or 'unknown_topic_1'<br>and only return the new label names, please return in json format like: {'labels': ['label name', 'label name']} |
| | LLM response | {<br>"labels": ["get info contact", "get stories news", "create reminder"]<br>} |
| Aggregating and merging labels | Prompt | Please analyze the provided list of labels to identify entries that are similar or duplicate, considering<br>synonyms, variations in phrasing, and closely related terms that essentially refer to the same concept.<br>Your task is to merge these similar entries into a single representative label for each unique concept<br>identified. The goal is to simplify the list by reducing redundancies without organizing it into subcategories<br>or altering its fundamental structure.<br>Here is the list of labels for analysis and simplification:['update call', 'get weather', ... , 'get reminder<br>location', 'create reminder'].<br>Produce the final, simplified list in a flat, JSON-formatted structure without any substructures or<br>hierarchical categorization like: {'merged_labels': ['label name', 'label name']} |
| | LLM response | {<br>"merged_labels": [<br>"update call",<br>"get weather",<br>"update method call",<br>...<br>"get stories news",<br>"get reminder location",<br>"create reminder"<br>]<br>} |
| Given label classification | Prompt | Given the label list and the sentence, please categorize the sentence into one of the labels.<br>Label list: [ 'update call', 'get weather',  ... , 'get reminder location', "create reminder" ]<br>Sentence: "Show me dates for music festivals in 2018."<br>You should only return the label name, please return in json format like: {'label_name': 'label'} |
| | LLM response | {<br>"label_name": "get event"<br>} |

Table 4: Case study in the MTOP-I dataset for different tasks. The '...' in the prompts and LLM responses indicate omitted labels to provide a clear presentation of the case study.

| Task | Name | #clusters | #data |
|---|---|---|---|
| Topic | ArxivS2S | 93 | 3674 |
| Emotion | GoEmo | 27 | 5940 |
| Domain | Massive-D | 18 | 2974 |
| Intent | Massive-I | 59 | 2974 |
| | MTOP-I | 102 | 4386 |

Table 5: Dataset stasistics

**ArxivS2S** (Muennighoff et al., 2023b) is a text clustering dataset in the domain of academic, it contains sentences describing a certain domain. **GoEmo** (Demszky et al., 2020) is a fine-grained dataset for emotion detection, multi-label or neutral instances are removed for text clustering purpose. **Massive-I/D** (FitzGerald et al., 2023) and **MTOP-I** (Li et al., 2021) are datasets originally used for classification but adapted for text clustering. "I" denotes intent and "D" denotes domain. Following Zhang et al. (2023b), all the datasets are splitted into large- and small-scale versions with the same number of clusters. Dataset statistics summary is shown in Table 5. We use small-scale version of datasets to reduce cost.

## D Baseline Implementation Details

Since different models all evaluated on different datasets, to better compare the performance of baseline models and our model, we implement the baseline models on the five datasets using the source code provided by the authors.

**K-means.** We use embeddings extracted from E5-large (Wang et al., 2022) and Instructor-large (Su et al., 2022) and apply K-means algorithm to obtain the text clustering result. We run the clustering five times with different seeds and calculate the average result as the final result.

**IDAS[3].** Following (De Raedt et al., 2023) ,we first generate labels using GPT-3 (text-davinci-003) for the five datasets used in this paper. For each test set, 5 JSON files are generated with different sample order, with the nearest neighbors $topk = 8$. After that, we produce the result with the generated labels and calculate the evaluation metrics.

**PAS[4].** We use the same experiment settings as (Wang et al., 2023b) and use GPT-3.5-turbo as the proposers and google/flan-t5-xl[5] as the assigners. For *cluster_num* parameter, we set it as the number of labels in the datasets.

**ClusterLLM[6].** Since ClusterLLM does not present its results in the ARI metric, we also reproduce its results on the five datasets. We choose the best performing model *ClusterLLM-I-iter* for comparison. This model adopts Instructor[7] as the embedder and applies the framework twice in an iterative way by using the previously fine-tuned model as initialization. The LLM used for triplet sampling and pairwise hierarchical sampling is GPT-3.5-turbo. We also re-perform the framework on ArxivS2S and GoEmo datasets to obtain the #clusters result in granularity analysis in Section 4.2, which is not presented in the original paper. The #clusters result of dataset Massive-I, Massive-D and MTOP-I is taken directly from the paper (Zhang et al., 2023b).

## E Given Label Percentage Experiment

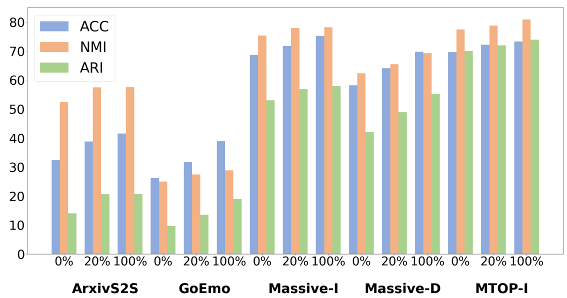


Figure 2: ACC, NMI, ARI of our method on five dataset with different percentage of given labels. 0% means no label is provided to the LLM, 20% means we give 20% of the total gold labels to the LLM during label generation and 100% means LLM is provided with all true labels and directly performs classification.

We provide the LLM with few-shot examples in label generation task to fully utilize its in-context learning ability. To demonstrate that using several examples can help LLM improve its performance, we conduct experiments with different percentage of gold labels given to the LLM. As shown in Figure 2, when provided with a few examples, the model improves its clustering result across all three evaluation metrics on all five datasets. Note that in the 100% case, the model is given all true labels and directly performs classification, which represents the theoretical upper bond "Ours (with gold labels)" introduced in Section 3.2.

[3] https://github.com/maarten-deraedt/IDAS-intent-discovery-with-abstract-summarization.

[4] https://github.com/ZihanWangKi/GoalEx.

[5] https://huggingface.co/google/flan-t5-xl.

[6] https://github.com/zhang-yu-wei/ClusterLLM

[7] https://huggingface.co/hkunlp/instructor-large

## F Label Merging Granularity Analysis

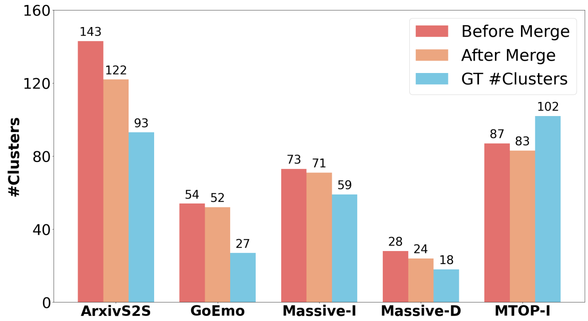


Figure 3: Label merging granularity on five datasets. "GT #Clusters" means the ground truth number of clusters in the dataset.

To justify the effectiveness of label merging task in our method, we conduct an comparative analysis on granularity before and after the merging task. Figure 3 shows that merging similar labels helps the model aggregate labels with same meanings, resulting in a cluster number closer to the the ground truth clusters. This merging method is especially effective when the number of labels is larger. For example, it aggregates 21 similar labels in the ArxivS2S dataset. Since the number of clusters can heavily impact the final clustering result, this method of improving the granularity is necessary.