Co-Evolving LLMs and Embedding Models via Density-Guided Preference Optimization for Text Clustering

Zetong Li[1], **Qinliang Su**[1,2*], **Minhua Huang**[3], **Yin Yang**[3]
[1]School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
[2]Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, China
[3]China Mobile Internet Company Ltd
{lizt9@mail2,suqliang@mail}.sysu.edu.cn
{huangminhua,yangyin}@cmic.chinamobile.com

Abstract

Large language models (LLMs) have shown strong potential in enhancing text clustering when combined with traditional embedding models. However, existing methods predominantly treat LLMs as static pseudo-oracles, *i.e.*, unidirectionally querying them for similarity assessment or data augmentation, while never seeking feedback from embedding models to improve them. In this work, we propose a training framework that enables bidirectional refinement between LLMs and embedding models. We first design task-aware prompts to guide the LLM in generating interpretations for the input texts. These interpretations are projected into the embedding space, in which interpretations that are preferred by the embedding model are selected based on their distribution densities. The selected interpretations are then used to fine-tune the LLM via preference optimization to prioritize the generation of helpful interpretations. Meanwhile, we enhance the embedding model via contrastive learning on the generated interpretations and perform clustering on the output embeddings, leading to iterative co-training between the LLM and the embedding model. Experiments on 14 benchmark datasets across 5 tasks demonstrate the effectiveness of our method.

1 Introduction

Text clustering is a fundamental task in natural language processing (NLP), with applications ranging from news recommendation (Bouras and Tsogkas, 2017), dialogue systems (Liu et al., 2019a) to opinion monitoring (Stieglitz et al., 2018). Traditional methods generally rely on statistical features like TF-IDF (Spärck Jones, 2004) or contextual embeddings from embedding models like Sentence-BERT (Reimers and Gurevych, 2019) with algorithms such as KMeans, DEC (Xie et al., 2016) and pseudo-labelling (YM. et al., 2020), *etc.* Since these methods still struggle to understand complex texts with diverse meanings and perspectives, recent studies have begun to leverage the powerful capabilities of LLMs for text clustering. A most direct approach (Huang and He, 2024) is to first ask LLMs for a candidate set of potential labels and then query LLMs about categories for each text. Since the generated label set is hard to coincide with the ground-truth one, this LLMs-only method does not perform well. Differently, most methods tend to use LLMs to assist embedding models for better clustering. For instance, (Zhang et al., 2023) mine boundary samples in embedding space and query LLMs for their similarities with samples from nearby clusters, as if LLMs are pseudo-oracles. The similarities between texts are used to construct positive and negative pairs for the contrastive training of embedding models. Another popular way is to regard LLMs as static augmenters. By querying for either keyphrases for each text or descriptions for each cluster, these methods (Viswanathan et al., 2024; De Raedt et al., 2023) concatenate or add the embeddings of original texts and augmented contents from LLMs, and then use KMeans to perform clustering on them.

Current approaches predominantly treat LLMs as static oracles or augmenters, *i.e.*, merely leveraging them through unidirectional interaction for similarity assessment or data augmentation without fine-tuning them to adapt to target datasets. However, since textual meaning is context-dependent, a frozen LLM queried passively by embedding models can only provide subjective interpretations based on its pretrained knowledge, failing to capture dataset-specific semantics. This limitation is particularly critical in text clustering, where understanding the underlying data distribution is essential. While directly incorporating all data into LLM prompts is infeasible, current methods lack effective mechanisms to transfer the data distribution knowledge into LLMs for improved contextual

*Corresponding author.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 4797–4809
November 4-9, 2025

understanding of texts.

To address the limitations above, in this paper, we propose a training framework that mutually trains the LLM and the embedding model for text clustering. We first design task-aware prompts to instruct the LLM to generate semantically rich and helpful interpretations for original texts. Then, we project these interpretations into the embedding space and select those locating at high-density region as the preferred ones, based on the intuition that high density implies that the interpretation is more likely to be semantically helpful under the dataset. We use the collected preference samples to fine-tune the LLM via preference optimization, thus effectively transfering the embedding model's understanding of data distribution into the LLM, making it more likely to generate helpful interpretations. Concurrently, we also enhance the embedding model through contrastive learning on the generated interpretations, thus forming a training circle that improves the two models mutually. Extensive experiments show that our method can effectively use the LLM and the embedding model for better clustering performance.

2 Related Work

Traditional Methods Previous works mainly rely on vector representations like TF-IDF features (Spärck Jones, 2004) and contextual embeddings from embedding models (Devlin et al., 2019; Liu et al., 2019b; Reimers and Gurevych, 2019). While early studies (Banerjee et al., 2007; Hu et al., 2009) tend to directly apply KMeans or Agglomerative Hierarchical Clustering (AHC), recent studies (Guo et al., 2017; Yang et al., 2017; Xu et al., 2017; Hadifar et al., 2019; Huang et al., 2020; Zhang et al., 2021b; Yin et al., 2022; Zheng et al., 2023; Li et al., 2024) use autoencoder or contrastive learning with DEC (Xie et al., 2016) or pseudo labelling (YM. et al., 2020) to achieve clustering. Among them, some works also introduce multiple models for text clustering like us. Different from attempts to combine TF-IDF features and embedding models (Li et al., 2024), our work focuses on leveraging LLMs together with embedding models.

LLM-Assisted Methods LLMs now have been increasingly used to enhance text clustering. Assuming that partial labels are known in advance, (Huang and He, 2024) use LLMs to generate a candidate label set and then achieve clustering via querying LLMs for text categories. Apart from obtaining labels, some methods (Wang et al., 2023; De Raedt et al., 2023) use LLMs to generate summarizations for clusters, which will then be regarded as label descriptions to assisting clustering. Differently, most methods (Zhang et al., 2023; An et al., 2024; Liang et al., 2024a,b; Viswanathan et al., 2024) tend to use LLMs as pseudo-oracles, by mining boundary samples through various criteria and then querying LLMs for their similarities. These additional information will then be used for training embedding models via contrastive learning. What's more, (Viswanathan et al., 2024) proposes to use LLMs to generate keyphrases for texts as augmentations, which is the most similar one to our method. However, we use a lightweight LLM to generate detailed interpretations and the used LLM will be trained to adapt to target datasets.

3 Method

3.1 Task-Aware Prompt for Text Interpretation

Label information such as label names or descriptions is generally unavailable in text clustering, but the overall clustering goal of the target data can often be inferred. For example, in dataset ArxivS2S, a rough inspection of the data suggests that it likely involves mining topics organized by subject areas. Thus, we refer to this high-level understanding as a task description DESC="topic: detailed subject area", and assume that it is available beforehand.

Different from existing works that use the LLM to query text similarities, we leverage the LLM for comprehensive interpretation of the input text. Specifically, for the target data to be clustered, we define a task description DESC="⟨TASK⟩: ⟨STANDARD⟩", where ⟨TASK⟩ means the overall task (*e.g.* topic, intent), and ⟨STANDARD⟩ means the detailed clustering standard (*e.g.* subject area, customer purpose). Given an input text TEXT, we then design the following prompt template T_{DESC}:

Prompt Template T_{DESC}

You are very good at natural language understanding.

For a given text, analyze its underlying meaning to present a better understanding. Remember that you should dissect the text in terms of **{TASK}**. You need to infer what **{STANDARD}** the text is about. Show your reasoning process and summarize by listing

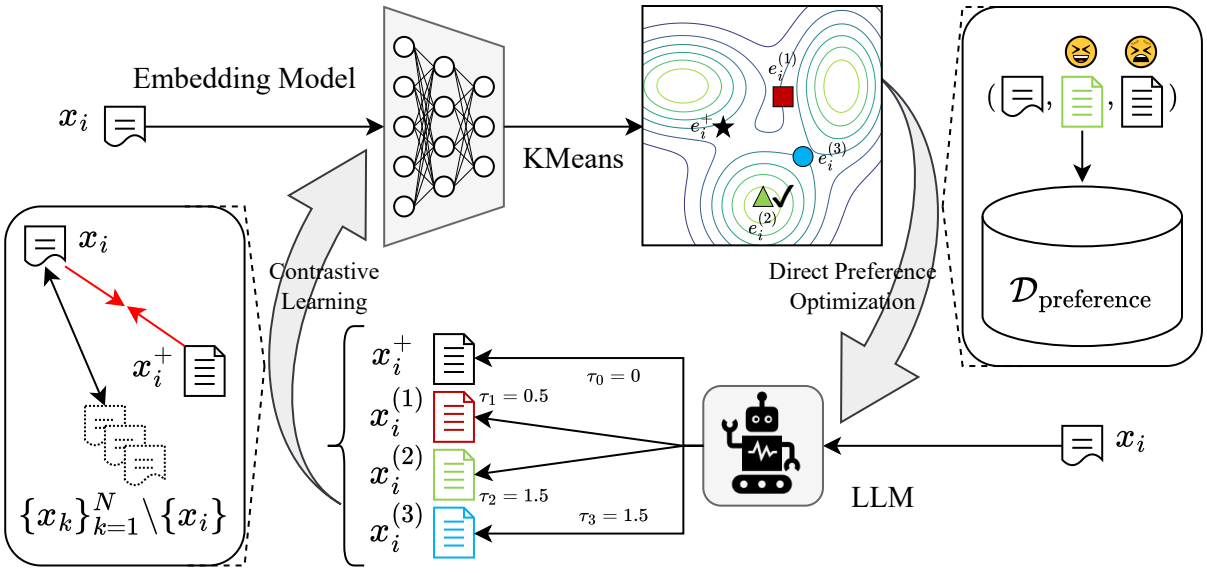

Figure 1: The training framework between the LLM and the embedding model. In this framework, the embedding model guides the LLM through preference optimization for more helpful interpretations, and benefit from these interpretations to improve clustering.

at most 8 keywords that reflect it. Do not be too lengthy and strictly follow the specified format:
Reasoning: ...
Keywords: ...

Text: **{TEXT}**

This prompt asks the LLM to analyze the text with respect to the given TASK (*e.g.*, topic, intent) from the perspective of STANDARD (*e.g.*, detailed subject area, banking customer purpose). Requirements such as listing the reasoning process encourage the LLM to generate rich content reflecting its comprehension of the text, while the subsequent summarization helps highlight key points. As a result, we can obtain:

$$x_i^+ = G(x_i; T_{\text{DESC}}), \tag{1}$$

where $G(\cdot)$ denotes the LLM, and x_i^+ is the output interpretation generated by the LLM for the original text x_i. Here we present a concrete example from the posts in dataset StackEx as follows, where the original text x_i is "06 Saturn Ion 2 2.2l P0101" and the task description is DESC="topic: post topic":

An Example Input-Output

Input x_i:
06 Saturn Ion 2 2.2l P0101

Output x_i^+:
Reasoning: The text appears to be referring to a specific car model, the Saturn Ion, with details about its engine and a diagnostic code. The number "2" likely indicates the second generation or a specific variant, while "P0101" is a common diagnostic trouble code related to the air flow meter.
Keywords: Saturn, Ion, engine, diagnostic, code, air flow, trouble

3.2 Training LLM with Preference Feedback

Since the LLM relies on designed prompts to interpret texts one by one, it cannot grapse the overall data distribution and thus may not align well with the needs of the embedding model. To address this problem, we propose to introduce the distributional information from the embedding space into the LLM by selecting text interpretations that are preferred by the embedding model and train the LLM with preference feedback.

Specifically, we first encode the deterministic interpretation x_i^+ as:

$$e_i^+ = E(x_i^+), \tag{2}$$

where $E(\cdot)$ is the embedding model, and apply KMeans on $\{e_i^+\}_{i=1}^N$ to obtain the cluster centers to represent the distributional information:

$$\{\mu_j\}_{j=1}^K = \text{KMeans}(\{e_i^+\}_{i=1}^N, K). \tag{3}$$

Here, K is the number of clusters, and KMeans$(\cdot)$ partitions embeddings into K clusters and outputs the cluster centers. Then, we further employ the LLM with different sampling temperatures to generate additional candidate interpretations and obtain their corresponding embeddings:

$$x_i^{(m)} = G_{\tau_m}(x_i; T_{\text{DESC}}), \tag{4}$$

$$e_i^{(m)} = E(x_i^{(m)}). \tag{5}$$

Here, $G_{\tau_m}(\cdot)$ denotes the LLM $G(\cdot)$ generating text interpretations with sampling temperature τ_m, and $x_i^{(m)}$ is the corresponding sampled candidate, where $m = 1, 2, \ldots, M$. By assuming that samples locating at high-density region are more reliable, we believe that these samples can reflect some meaningful semantics in the embedding space to some extent. Therefore, for a given text x_i, we define the one among $\{e_i^+\} \cup \{e_i^{(m)}\}_{m=1}^M$ that lies in a high-density region as the preferred interpretation by the embedding model—*i.e.*, a good or helpful text interpretation in terms of the embedding model. Letting $e_i^{(0)} \triangleq e_i^+$, we compute the distances between all candidates for x_i and the cluster centers as follows:

$$d_m = \min_k ||e_i^{(m)} - \mu_k||_2. \tag{6}$$

In practice, we repeat KMeans and distance calculation multiple times and average d_m. We choose the one with minimum distance as the preferred interpretation, while the others as the opposite:

$$w = \arg\min_m d_m, \tag{7}$$

$$l \in \{0, 1, ..., M\} \backslash \{w\}. \tag{8}$$

Here, the index w means winning and l means losing. In this way, the distributional information can be indirectly represented as the preference data:

$$D_{\text{preference}} = \{(x_i, x_i^{(w)}, x_i^{(l)}) | d_w < d_l\}_{i=1}^N. \tag{9}$$

To leverage the above preference feedback, we follow the popular preference optimization paradigm, and use direct preference optimization (DPO) (Rafailov et al., 2023) to fine-tune the LLM, so that the probability of generating the preferred response can be raised while that of generating the not preferred one can be suppressed. Specifically, for text x_i with the preferred interpretation $x_i^{(w)}$ and the one not preferred $x_i^{(l)}$, we compute:

$$\ell_{il}^{\text{DPO}} = -\log \sigma \left(\beta \log \frac{G(x_i^{(w)}|x_i; T_{\text{DESC}})}{G_{\text{ref}}(x_i^{(w)}|x_i; T_{\text{DESC}})} - \beta \log \frac{G(x_i^{(l)}|x_i; T_{\text{DESC}})}{G_{\text{ref}}(x_i^{(l)}|x_i; T_{\text{DESC}})} \right), \tag{10}$$

where $G(y|x; T)$ is the probability of generating response y when given input x in the LLM, and $G_{\text{ref}}(\cdot)$ is the frozen LLM used as the reference model, and β is used to control the difference between $G(\cdot)$ and $G_{\text{ref}}(\cdot)$. Thus, by minimize the DPO loss for the LLM $G(\cdot)$:

$$\mathcal{L}_G = \mathcal{L}_{\text{DPO}} = \frac{1}{MN} \sum_{i=1}^N \sum_{l \neq w} \ell_i^{\text{DPO}}, \tag{11}$$

we can effectively transfer the distributional information and the preference of the embedding model to the LLM for better text interpretation.

3.3 Training Embedding Model via Contrastive Learning

Since text interpretations from the LLM contain not only useful contents but also lots of redundancy, we further leverage contrastive learning (Chen et al., 2020; Gao et al., 2021) to extract helpful semantics. A simple way is to regard the interpretation x_i^+ as positive of x_i, that is:

$$\ell_i^{\text{CL}} = -\log \frac{\Delta(e_i, e_i^+)}{\Delta(e_i, e_i^+) + \sum_{k \neq i} \Delta(e_i, e_k)}, \tag{12}$$

where $e_i = E(x_i)$ and $e_i^+ = E(x_i^+)$ are the embeddings of the original text and its corresponding interpretation, and $\Delta(e_i, e_i^+) = \exp(\cos(e_i, e_i^+)/\tau)$ is the similarity function, and τ is the temperature parameter. We also introduce text augmentation for interpretation using dropout to enhance the training process by forwarding the embedding model twice with different dropout rates to obtain a different but related virtual augmentation $e_i^\dagger = E_\xi(x_i^+)$. We further view it as positive of e_i^+ and construct the following loss:

$$\ell_i^{\text{CL}^\dagger} = -\log \frac{\Delta(e_i^+, e_i^\dagger)}{\Delta(e_i^+, e_i^\dagger) + \sum_{k \neq i} \Delta(e_i^+, e_k^+)}. \tag{13}$$

Therefore, by optimizeing the contrastive loss:

$$\mathcal{L}_{\text{CL}} = \frac{1}{2N} \sum_{i=1}^N (\ell_i^{\text{CL}} + \ell_i^{\text{CL}^\dagger}), \tag{14}$$

the interpretation can not only be pulled to the original text, but also to its virtual augmentation, which can preserve rich semantics from the LLM and not excessively deviate from the original text.

To encourage the embedding space to form cluster structures, we use the DEC loss (Xie et al., 2016) as regularization. We use the Student's t-distribution (van der Maaten and Hinton, 2008) to compute the probability that e_i^+ belongs to cluster ν_j as $q_{ij} = \frac{(1+||e_i^+ - \nu_j||_2^2)^{-(\alpha+1)/2}}{\sum_{k=1}^{K}(1+||e_i^+ - \nu_k||_2^2)^{-(\alpha+1)/2}}$, where $\{\nu_j\}_{j=1}^K$ are trainable parameters initialized by KMeans, and α represents the degree of freedom and is set to 1. Then the auxiliary target distribution $p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{k=1}^{K} q_{ik}^2/f_k}$ is introduced, where $f_j = \sum_{i=1}^{N} q_{ij}$. Thus, the cluster-level regularization DEC is computed as:

$$\mathcal{L}_{\text{DEC}} = \frac{1}{N}\sum_{i=1}^{N} D_{\text{KL}}(p_i||q_i) = \sum_{i=1}^{N}\sum_{j=1}^{K} p_{ij}\log\frac{p_{ij}}{q_{ij}}. \tag{15}$$

Finally, after training the embedding model with the following loss:

$$\mathcal{L}_E = \mathcal{L}_{\text{CL}} + \lambda\mathcal{L}_{\text{DEC}}, \tag{16}$$

where λ is a weighting parameter, we can encode the text interpretations $\{x_i^+\}_{i=1}^N$ to obtain $\{e_i^+\}_{i=1}^N$, and achieve clustering by applying KMeans on top of them to get the final clustering result $\{c_i \in \{1, 2, ..., K\}\}_{i=1}^N$. In practice, we can optimizing (16) and (11) iteratively to achieve progressive improvement.

4 Experiment

4.1 Experimental Setup

Datasets We evaluate our method on 14 datasets involving 5 tasks following (Zhang et al., 2023). The statistics of the datasets are shown in Table 6 in Appendix A, and the details are as follows: (i) **Topic Mining**: Three datasets, including **ArxivS2S**, **Reddit** and **StackEx** from MTEB (Muennighoff et al., 2023), are used for evaluating capabilities of methods in topic mining. These datasets contain various fine-grained topic categories. (ii) **Emotion Detection**: **GoEmo** (Demszky et al., 2020) is a dataset with fine-grained emotions. The one we use here is processed by (Zhang et al., 2023), where multi-label or neutral instances are removed. (iii) **Type Discovery**: Three datasets, including **FewEvent**, **FewNerd** and **FewRel** originated from information extraction, focus on tasks about event, entity and relation type discovery (Li et al., 2022). (iv) **Domain Discovery**: Three datasets, including **CLINC(D)**, **Massive(D)** and **MTOP(D)**, aim at mining the potential scenarios where the user utterances may occur. (v) **Intent Discovery**: Unlike domain discovery, four datasets, including **CLINC(I)**, **Massive(I)**, **MTOP(I)** and **Bank77**, are trying to discover unknown intents in user utterances (Zhang et al., 2021b).

Baselines Following (Zhang et al., 2023), we compare our method with the baselines below, including traditional methods and LLM-assisted methods:

- **E5-KMeans** is a simple baseline that performs KMeans on embeddings from the embedding model *e5-large* (Wang et al., 2022).
- **SCCL** (Zhang et al., 2021a) bases on embedding models only and applies contrastive learning and DEC loss (Xie et al., 2016) to achieve clustering.
- **Label-Gen** (Huang and He, 2024) bases on LLMs only and assumes that partial labels are known. They query LLMs with unlabeled texts to obtain candidate label set, and then use LLMs for zero-shot classification via QA to achieve clustering.
- **ClusterLLM** (Zhang et al., 2023) utilizes both LLMs and embedding models for clustering. They mine boundary sample in embedding space together with candidate samples from its two closest clusters, and then query the LLM for their similarity relationships, which are used for training the embedding model with contrastive learning. The clustering result are obtained by KMeans.
- **Keyphrase** (Viswanathan et al., 2024) generates keyphrases for texts with the LLM and encode them into embeddings with the embedding model. The embeddings of texts and keyphrases are then concatenated together and clustered using KMeans.

Evaluation Metrics We use metrics accuracy (ACC) and normalized mutual information (NMI) to evaluate the performance of our method. Following previous works, we repeat KMeans 5 times with different seeds and report the mean values. We left training details in Appendix A.1.

Method	Topic						Emotion		Type	
	ArxivS2S		Reddit		StackEx		GoEmo		FewEvent	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
E5-KMeans	35.68	56.96	43.52	51.87	43.79	62.65	21.75	21.73	37.34	59.29
SCCL	33.57	55.39	40.89	50.31	41.39	61.96	17.51	14.20	31.62	52.52
Label-Gen	26.21	52.50	29.84	45.43	33.85	57.47	16.75	**29.71**	34.35	60.03
ClusterLLM	37.89	59.17	48.96	56.22	46.43	65.44	19.69	18.49	**38.76**	**62.89**
Keyphrase	38.77	59.59	49.14	56.73	46.63	65.57	**29.77**	28.65	35.80	58.79
Ours (iter-0)	38.81	59.69	49.11	56.00	46.42	65.02	19.72	15.59	35.91	57.43
Ours (iter-1)	40.45	60.76	49.74	56.46	46.97	65.72	23.87	19.80	37.01	58.45
Ours (iter-2)	**40.94**	**61.09**	**50.59**	**56.82**	**47.74**	**65.79**	28.20	25.00	38.67	59.70

Method	Type				Domain					
	FewNerd		FewRel		CLINC(D)		Massive(D)		MTOP(D)	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
E5-KMeans	27.49	42.35	41.98	56.79	59.17	55.15	61.57	64.99	87.23	83.00
SCCL	30.21	44.68	39.22	54.33	60.53	57.56	60.09	62.26	84.86	83.43
Label-Gen	29.14	**56.36**	27.10	52.45	24.64	52.65	33.86	58.49	49.98	66.51
ClusterLLM	28.81	44.13	46.09	60.46	49.40	51.32	60.36	64.05	84.32	82.33
Keyphrase	29.16	44.30	46.07	60.18	61.68	58.11	62.96	66.54	**93.06**	**87.78**
Ours (iter-0)	32.19	47.38	48.37	62.08	60.68	58.20	61.13	65.03	89.92	83.56
Ours (iter-1)	33.73	49.08	51.34	64.81	63.40	59.95	64.75	66.60	89.28	83.33
Ours (iter-2)	**36.85**	52.84	**52.73**	**66.51**	**64.07**	**62.08**	**65.77**	**66.76**	89.69	84.06

Method	Intent								Avg	
	CLINC(I)		Massive(I)		MTOP(I)		Bank77			
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
E5-KMeans	79.90	91.72	54.81	72.15	33.37	71.43	62.90	77.90	49.32	62.00
SCCL	84.29	91.07	36.34	56.36	27.22	64.68	59.73	73.35	46.25	58.72
Label-Gen	39.47	65.15	44.62	62.89	**47.56**	57.95	36.30	60.86	33.83	55.60
ClusterLLM	83.83	93.10	56.48	74.59	35.17	72.56	**72.81**	83.53	50.64	63.45
Keyphrase	83.31	93.18	57.68	75.05	35.33	72.68	68.28	81.41	52.69	64.90
Ours (iter-0)	85.86	92.98	58.08	74.98	35.02	71.43	71.53	83.00	52.34	63.74
Ours (iter-1)	86.70	93.76	60.05	75.14	36.30	73.00	70.27	83.36	53.85	65.02
Ours (iter-2)	**87.67**	**94.35**	**60.80**	**75.65**	37.75	**74.01**	70.54	**83.64**	**55.14**	**66.31**

Table 1: The clustering performance of the baselines and our method. The best results are bolded and the second ones are underlined. In our method, the LLM is *Qwen2.5-7B-Instruct* and the embedding model is *e5-large*. The baselines are reproduced with the same set of models.

4.2 Experimental Result

4.2.1 Performance Comparison

The clustering performance of the baselines and our method is shown in Table 1. It can be seen that our method achieves promising performance on most datasets. We can also find some interesting phenomena by further analyzing the resutls.

Traditional methods such as E5-KMeans and SCCL do not perform well on most datasets. This can be attributed to the fact that embedding models are struggling to understand texts from diverse tasks and complex scenarios. With LLMs assisting text clustering, obvious improvement can be observed in ClusterLLM and Keyphrase. While Label-Gen can perform well on some datasets like MTOP(I), the difficulty of generating a label set as the ground-truth one makes it perform badly on most datasets. By contrast, ClusterLLM and Keyphrase query LLMs for similarities or expansions. These additional information helps embedding models present better performance.

However, existing methods simply ignore possible adjustment or feedback for the LLM. In contrast, by using the LLM to provide text interpretations for training the embedding model unidirectionally, our method (iter-0) has surpassed most baselines. Furthermore, by reversely using the trained embedding model to provide preference

	ArxivS2S	Reddit	StackEx	GoEmo	FewEvent	FewNerd	FewRel
$\tau_m = 0.0$	38.81	49.11	46.42	19.72	35.91	32.19	48.37
$\tau_m = 0.5$	38.38	48.98	46.51	20.26	35.08	32.11	48.79
$\tau_m = 1.0$	38.27	49.24	46.10	19.64	36.11	32.58	48.39
$\tau_m = 1.5$	38.33	49.13	45.97	19.99	35.72	32.74	48.75
Random (iter-1)	38.99	48.46	46.14	19.85	36.41	31.99	47.86
Ours (iter-1)	**40.45**	**49.74**	**46.97**	**23.87**	**37.01**	**33.73**	**51.34**
	CLINC(D)	Massive(D)	MTOP(D)	CLINC(I)	Massive(I)	MTOP(I)	Bank77
$\tau_m = 0.0$	60.68	61.13	89.92	85.86	58.08	35.02	**71.53**
$\tau_m = 0.5$	61.58	61.57	**90.07**	85.55	58.16	34.72	71.34
$\tau_m = 1.0$	60.20	62.63	89.86	86.02	57.42	35.06	70.69
$\tau_m = 1.5$	61.20	61.62	89.49	84.77	57.92	34.17	69.06
Random (iter-1)	60.85	60.84	89.12	85.40	58.27	34.65	70.65
Ours (iter-1)	**63.40**	**64.75**	89.28	**86.70**	**60.05**	**36.30**	70.27

Table 2: Ablation study for the construction strategy of the preference data (ACC).

	ArxivS2S	Reddit	StackEx	GoEmo	FewEvent	FewNerd	FewRel
Ours (iter-1)	**40.45**	**49.74**	**46.97**	23.87	**37.01**	**33.73**	51.34
w/o DEC	39.64	49.23	46.63	24.44	36.40	33.54	**51.55**
w/o Dropout	38.62	49.30	46.47	**25.06**	36.63	33.36	51.39
	CLINC(D)	Massive(D)	MTOP(D)	CLINC(I)	Massive(I)	MTOP(I)	Bank77
Ours (iter-1)	**63.40**	**64.75**	89.28	**86.70**	60.05	36.30	70.27
w/o DEC	63.27	64.70	89.43	86.52	60.03	36.16	**71.82**
w/o Dropout	63.17	64.65	**90.19**	86.31	**61.18**	**37.03**	68.84

Table 3: Ablation study for the training objective in the embedding model (ACC).

feedback for training the LLM, our method (iter-1) shows better results. Finally, by performing one more iteration on this bidirectional training process, our method (iter-2) can further improve the performance, validating that with text interpretations and preference feedback as bridges, our method can achieve the mutual improvement between the LLM and the embedding model.

4.2.2 Ablation Study

We conduct some experiments to study whether the construction strategy of the preference data really works. We first probe candidate interpretations sampled from different temperatures respectively. In Table 2, we show the ACC results when using only the candidate interpreations $\{x_i^{(m)}\}_{i=1}^N$ with temperature τ_m for training the embedding model. We can see that all cadidates show generally similar results, which means that the improvement of our method does not come from any particular candidate interpretations but rather preference filtering and LLM fine-tuning. Further, we introduce Random that randomly chooses preferred candidate to fine-tune the LLM. Compared with Random, we can observe that our method is still better, which demonstrates the effectiveness of our construction strategy of the preference data.

To understand the effect of the training objective in the embedding model, we first remove the DEC loss, and then further remove the contrastive loss for the dropout augmentation. We show the ACC results in Table 3. We can observe that the two losses are useful on most datasets, but some may be adversely affected instead. Thus, due to the diversity of tasks, different datasets may require different training objective for extracting the meaningful semantics. To avoid excessive tuning, we use the same objective for all datasets here.

4.2.3 Influence of Hyperparameters

Influence of #Iterations To find out whether more training iterations can lead to increasing clustering performance, we show the performance up to 4 iterations in Figure 2. On most of the datasets, we can see that the ACC and NMI metric are continuing to increase but the paces become gradually slower and slower, which is reasonable and predictable. Howerver, due to the cost of training the LLM, we generally choose 1 or 2 iterations in practice.

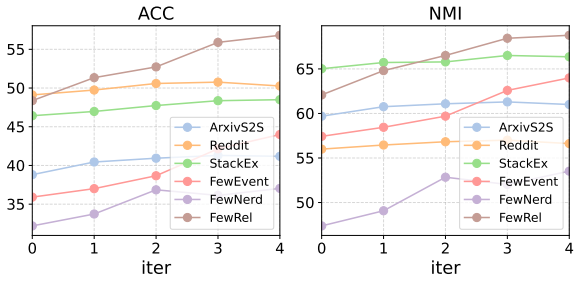

Figure 2: Influence of the number of the iterations

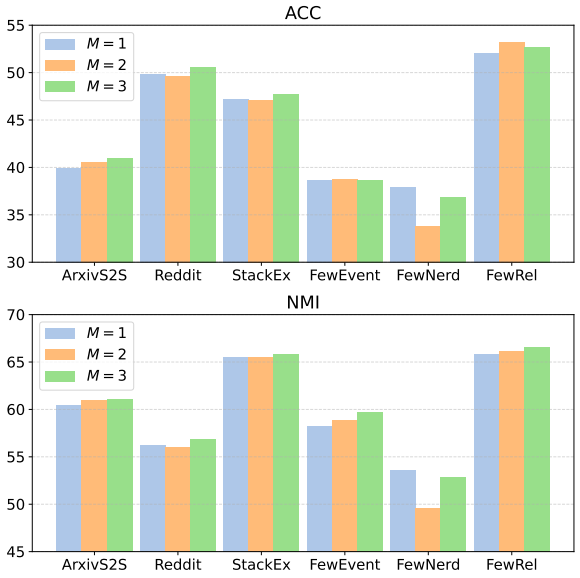

Figure 3: Influence of the number of the candidates

Influence of #Candidates To see whether different numbers of selectable candidates have imapcts on the clustering perfomance, we set $M = 1, 2, 3$ for comparison. The results are shown in Figure 3. We can observe that in most cases, the performance increases when the number of the candidates increases. This can be due to the fact that more sampled candidates allow larger space for exploring better interpretations. However, since more random sampling brings more diverse texts that may hinder the training of the LLM, the performance may drop on some datasets. Based on our experimental environment, we generally choose $M = 3$.

4.2.4 Case Study

We present a case about emotion detection to show the interpretation from the LLM for comprehensive understanding, and see how the feedback from the embedding model helps the LLM generate better results. As shown in Table 5, the generated interpretation is getting closer to the ground-truth label as the training proceeds. The example text from GoEmo is iterpreted as anger by the LLM without training, but it is later understood as amusement after training. Finally, the emotion is highlighted in the summarized keywords, demonstrating the effectiveness of our method.

	FewNerd	FewRel	CLINC(I)	Massive(I)
iter-0	30.10	45.58	84.52	56.81
iter-1	31.22	46.66	85.65	57.96
iter-2	32.34	46.94	86.61	59.11
task-aware	36.85	52.73	87.67	60.80

Table 4: The clustering performance of our method using the task-agnostic prompt (ACC).

4.2.5 Investigation of Task-Agnostic Prompt

To further investigate whether the performance improvement originates from the training framework or the task-aware prompts, we conduct an experiment using a task-agnostic prompt on several datasets. Specifically, we remove the task knowledge in the prompt template as follows:

Prompt Template T (Task-Agnostic)

You are very good at natural language understanding.

For a given text, analyze its underlying meaning to present a better understanding. Show your reasoning process and summarize by listing at most 8 keywords that reflect it. Do not be too lengthy and strictly follow the specified format:
Reasoning: ...
Keywords: ...

Text: **{TEXT}**

The corresponding results are shown in Table 4. It can be seen that our method can still work using the task-agnostic prompt, which demonstrates the effectiveness of the training framework. At the same time, using the task-aware prompts can further improve the performance, which is expected. In practice, we assume the task knowledge is available since it can be expressed as the clustering goal and obtained from users.

5 Conclusion

In this paper, we propose a training framework that leverages LLMs and embedding models for text clustering. We use the embedding model to select helpful interpretations, and optimize the LLM via preference optimization to generate better outputs.

ground-truth label	**amusement**
original text	I know you're joking, but there are people here either stupid or desperate enough to believe and perpetuate such idiocy.
iter-0	Reasoning: The text expresses skepticism and disappointment towards others who might believe in something considered foolish or false. The speaker acknowledges that some people might be easily deceived, which implies a sense of frustration and possibly even a touch of anger. \n\n Keywords: Skepticism, Disappointment, Frustration, Anger, Deception, Foolishness, Belief, Perpetuation
iter-1	Reasoning: The text expresses skepticism and disappointment towards certain individuals who believe in something deemed foolish or nonsensical. The speaker acknowledges that some people might take the joke seriously, indicating a sense of **amusement** mixed with frustration. This sentiment suggests a blend of humor at the expense of others' gullibility and a deeper concern for the impact of such beliefs. \n\n Keywords: Skepticism, Frustration, Humor, Gullibility, Disappointment, Joke, Nonsensical, Belief
iter-2	Reasoning: The text expresses skepticism and disdain towards those who are perceived as naive or desperate. The speaker is also slightly amused by the joke but predominantly feels annoyed or irritated by the topic being discussed. \n\n Keywords: Skepticism, Disdain, **Amusement**, Irritation, Naivety, Desperation, Joke, Idiocy

Table 5: A text interpretation case from the LLM.

Then improved interpretations are used to enhance the embedding model through contrastive learning and achieve clustering, forming a training circle between the two models. Experiments show the effectiveness of our method in text clustering by training both the LLM and the embedding model.

Limitations

As previous works, our method requires that the number of clusters is known. Furthermore, by incorporating the LLM in both the training and inference stage, our method can incur additional costs. However, we have restricted the training cost by limiting the number of the preference data samples (10000 samples in our experiment), thus it is actually controllable in training. What's more, thanks to the lightweight LLM, our method can also be accelerated in inference with open-source acceleration library, which helps reducing the inference cost in practice. Nevertheless, we will leave further improving the efficiency of the method as our future work.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 62276280), Guangzhou Science and Technology Planning Project (No. 2024A04J9967).

References

Wenbin An, Wenkai Shi, Feng Tian, Haonan Lin, QianYing Wang, Yaqiang Wu, Mingxiang Cai, Luyan Wang, Yan Chen, Haiping Zhu, and Ping Chen. 2024. Generalized category discovery with large language models in the loop. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 8653–8665.

Somnath Banerjee, Krishnan Ramanathan, and Ajay Gupta. 2007. Clustering short texts using wikipedia. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 787–788.

Christos Bouras and Vassilis Tsogkas. 2017. Improving news articles recommendations via user clustering. *International Journal of Machine Learning and Cybernetics*, 8(1):223–237.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 1597–1607.

Maarten De Raedt, Fréderic Godin, Thomas Demeester, and Chris Develder. 2023. IDAS: Intent discovery with abstractive summarization. In *Proceedings of the 5th Workshop on NLP for Conversational AI (NLP4ConvAI 2023)*, pages 71–88.

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. 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.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910.

Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. 2017. Improved deep embedded clustering with local structure preservation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1753–1759.

Amir Hadifar, Lucas Sterckx, Thomas Demeester, and Chris Develder. 2019. A self-training approach for short text clustering. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 194–199.

Xiaohua Hu, Xiaodan Zhang, Caimei Lu, E. K. Park, and Xiaohua Zhou. 2009. Exploiting wikipedia as external knowledge for document clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 389–396.

Chen Huang and Guoxiu He. 2024. Text clustering as classification with llms. *arXiv preprint arXiv:2410.00927*.

Shaohan Huang, Furu Wei, Lei Cui, Xingxing Zhang, and Ming Zhou. 2020. Unsupervised fine-tuning for text clustering. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5530–5534.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, page 611–626.

Sha Li, Heng Ji, and Jiawei Han. 2022. Open relation and event type discovery with type abstraction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6864–6877.

Zetong Li, Qinliang Su, Shijing Si, and Jianxing Yu. 2024. Leveraging BERT and TFIDF features for short text clustering via alignment-promoting cotraining. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14897–14913.

Jinggui Liang, Lizi Liao, Hao Fei, and Jing Jiang. 2024a. Synergizing large language models and pre-trained smaller models for conversational intent discovery. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14133–14147.

Jinggui Liang, Lizi Liao, Hao Fei, Bobo Li, and Jing Jiang. 2024b. Actively learn from LLMs with uncertainty propagation for generalized category discovery. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7845–7858.

Jiao Liu, Yanling Li, and Min Lin. 2019a. Review of intent detection methods in the human-machine dialogue system. *Journal of Physics: Conference Series*, 1267(1):012059.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Karen Spärck Jones. 2004. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 60(5):493–502.

Stefan Stieglitz, Milad Mirbabaie, Björn Ross, and Christoph Neuberger. 2018. Social media analytics – challenges in topic discovery, data collection, and data preparation. *International Journal of Information Management*, 39:156–168.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2023. One embedder, any task: Instruction-finetuned text embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1102–1121.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

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. 2023. Goal-driven explainable clustering via language descriptions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10626–10649.

Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 478–487.

Jiaming Xu, Bo Xu, Peng Wang, Suncong Zheng, Guanhua Tian, Jun Zhao, and Bo Xu. 2017. Self-taught convolutional neural networks for short text clustering. *Neural Networks*, 88:22–31.

Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. 2017. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 3861–3870.

Qing Yin, Zhihua Wang, Yunya Song, Yida Xu, Shuai Niu, Liang Bai, Yike Guo, and Xian Yang. 2022. Improving deep embedded clustering via learning cluster-level representations. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2226–2236.

Asano YM., Rupprecht C., and Vedaldi A. 2020. Self-labelling via simultaneous clustering and representation learning. In *International Conference on Learning Representations*.

Dejiao Zhang, Feng Nan, Xiaokai Wei, Shang-Wen Li, Henghui Zhu, Kathleen McKeown, Ramesh Nallapati, Andrew O. Arnold, and Bing Xiang. 2021a. Supporting clustering with contrastive learning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5419–5430.

Hanlei Zhang, Hua Xu, Ting-En Lin, and Rui Lyu. 2021b. Discovering new intents with deep aligned clustering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14365–14373.

Yuwei Zhang, Zihan Wang, and Jingbo Shang. 2023. 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.

Xiaolin Zheng, Mengling Hu, Weiming Liu, Chaochao Chen, and Xinting Liao. 2023. Robust representation learning with reliable pseudo-labels generation via self-adaptive optimal transport for short text clustering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10493–10507.

A Appendix

A.1 Training Details

We implement our method using PyTorch(Paszke et al., 2019). For the LLM, since the baseline methods rely on the powerful GPT models which cannot be fine-tuned, we adopt *Qwen2.5-7B-Instruct* for all reproductions to ensure fair comparisons. For the embedding model, considering the difficulty of capturing the semantics of the complex texts, we use *e5-large*, a variant of the BERT model refined by (Wang et al., 2022).

In terms of experimental settings, we use the Adam optimizer. For fine-tuning the embedding model, we set the batch size to 32 and the learning rate to 3e-7. The number of clusters K involved in DEC, used for regularization, is fixed at 256 to preserve fine-grained semantic clusters. For fine-tuning the LLM, we set the batch size to 4 and the learning rate to 1e-5. The parameter β in DPO is fixed at 0.25. Due to memory constraints, LoRA is applied for fine-tuning, with the rank set to 64. Additionally, the weighting parameter λ is set to 1 for most datasets but is set to 10 for ArxivS2S, Reddit and StackEx, and the temperature parameter τ is set to 0.1. What's more, we set the number of the candidates (including the deterministic one x_i^+ with sample temperature 0) to 4, and the corresponding sample temperatures are set to 0, 0.5, 1.0 and 1.5. We generally perform 2 iterations for training both the LLM and the embedding model.

In practice, since the method involves generating a large amount of candidate interpretations using the LLM, we employ vLLM (Kwon et al., 2023) as the inference engine to significantly accelerate the generation process. When constructing the preference data, we use KMeans to obtain cluster cen-

task	dataset	#texts	#clusters	DESC
topic	ArxivS2S	3674	93	topic: detailed subject area
	Reddit	3217	50	topic: post topic
	StackEx	4156	121	topic: post topic
emotion	GoEmo	5940	27	emotion: potential emotion
type	FewEvent	4742	34	event type: the event
	FewNerd	3789	58	entity type: the entity
	FewRel	4480	64	relation type: the relation
domain	CLINC(D)	4500	10	utterance scenario: life scenario
	Massive(D)	2974	18	utterance scenario: life scenario
	MTOP(D)	4386	11	utterance scenario: life scenario
intent	CLINC(I)	4500	150	intent: banking customer purpose
	Massive(I)	2974	59	intent: user intent
	MTOP(I)	4386	102	intent: user intent
	Bank77	3080	77	intent: user intent

Table 6: The statistics of the datasets.

ters and select high-confidence candidates based on their densities. Considering KMeans' sensitivity to initialization of cluster centers, we perform KMeans 100 times and compute the distances between candidates and cluster centers in each run. The final score is obtained by averaging these distances to reduce randomness. Given the computing resource of four RTX 3090 GPUs, we set the number of the candidates to 4. As a result, the preference data used in DPO is three times the size of the original dataset. We randomly sample 10000 training samples, which can control the LLM training time to about 40min. As for the embedding model with contrastive learning, it takes about 10min for each dataset on average.

Under the current settings, we acknowledge that the cost of training the LLM remains considerable. However, we believe that this concern will become less critical in the near future. Recent advances have shown that smaller LLMs (*e.g.*, 0.5B, 1.5B, 3B) are rapidly improving in capability. It's forseeable that these compact models will become viable alternatives to larger ones for many tasks including text clustering, making our framework more efficient without compromising performance.

A.2 Results of Different Models

To evaluate the influence of different models, we further conduct experiments using *Meta-Llama-3-8B-Instruct* to replace Qwen. The results are shown in Table 7. It can be observed that our method remains effective across multiple datasets as the number of the iterations increases. We also evaluate our method using *instructor-large* (Su et al., 2023) as the embedding model. The results are shown in Table 8. We can see that even when using a non-BERT architecture such as the T5-based encoder as the embedding model, our method can still maintain strong performance.

A.3 Analysis of Failure Case

We observe that performance gains are not always guaranteed across all datasets, such as the Bank77 dataset. To understand the possible reason, we probe and show a case in Table 9. It can be seen that two texts from the same ground-truth category may be interpreted very differently by the LLM. As a result, their corresponding embeddings may deviate and thus fail to capture the true semantics. In such a scenario, preference guidede by density can be low-confidence, which limits the performance improvement and may even lead to degradation. However, this issue can be attributed to the capability of the LLM, as *Meta-Llama-3-8B-Instruct* performs more consistently and better than *Qwen2.5-7B-Instruct* on the Bank77 dataset, as shown in Table 7.

	ArxivS2S	Reddit	StackEx	GoEmo	FewEvent	FewNerd	FewRel
KMeans	35.68	43.52	43.79	21.75	37.34	27.49	41.98
iter-0	37.32	49.26	46.58	21.03	35.63	29.98	45.33
iter-1	38.95	**49.59**	47.11	28.44	36.95	**30.75**	47.72
iter-2	**40.19**	48.73	**47.58**	**33.02**	**41.25**	28.31	**48.38**
	CLINC(D)	Massive(D)	MTOP(D)	CLINC(I)	Massive(I)	MTOP(I)	Bank77
KMeans	59.17	**61.57**	87.23	79.90	54.81	33.37	62.90
iter-0	60.25	57.55	87.45	83.61	53.38	33.73	69.78
iter-1	**62.44**	57.00	89.33	85.36	58.48	35.41	70.45
iter-2	56.99	56.79	**90.30**	**85.47**	**60.36**	**35.81**	**71.24**

Table 7: The ACC results when replacing *Qwen2.5-7B-Instruct* with *Meta-Llama-3-8B-Instruct*

	ArxivS2S	Reddit	StackEx	GoEmo	FewEvent	FewNerd	FewRel
KMeans	24.32	53.17	41.87	24.67	38.73	28.65	39.93
iter-0	31.09	**54.37**	43.15	24.97	41.22	33.34	47.91
iter-1	33.39	54.25	45.18	25.79	40.51	36.98	52.06
iter-2	**34.88**	53.60	**45.81**	**26.34**	**44.50**	**42.61**	**52.60**
	CLINC(D)	Massive(D)	MTOP(D)	CLINC(I)	Massive(I)	MTOP(I)	Bank77
KMeans	54.25	57.89	**91.41**	81.69	56.15	31.39	63.89
iter-0	62.44	64.55	89.51	86.52	58.67	36.85	**66.30**
iter-1	**63.36**	64.91	84.06	**87.16**	59.75	39.28	65.71
iter-2	60.98	**67.07**	81.75	86.84	**60.87**	**42.00**	65.08

Table 8: The ACC results when replacing *e5-large* with *instructor-large*

ground-truth label	**lost or stolen card**
original text A	card is lost, please help
interpretation	Reasoning: The text indicates a customer has **lost** their bank card and is seeking assistance from a service provider, likely a bank or financial institution. The customer's intent is to resolve the issue of a **lost** card, which could involve reporting the loss, blocking the card, and possibly arranging for a replacement. \n\n Keywords: **lost**, card, assistance, bank, report, block, replace, issue
original text B	How do I deal with a stolen card?
interpretation	Reasoning: The text is asking for guidance on handling a specific issue related to a bank card that has been **stolen**. The intent is to seek advice on the steps to take to secure one's financial information and prevent further unauthorized transactions. \n\n Keywords: **stolen**, card, deal, guidance, security, transactions, prevent, advice

Table 9: A failure case from the Bank77 dataset. Text A and B are from the same category but interpreted in different ways (keywords) by the LLM, which gradually leads to divergence in the embedding space.