Improving Clustering with Positive Pairs Generated from LLM-Driven Labels

XiaoTong Zhang and **Ying Li**[*]
College of Computer Science and Technology, Zhejiang University, Hangzhou, China
22321341@zju.edu.cn, cnliying@zju.edu.cn

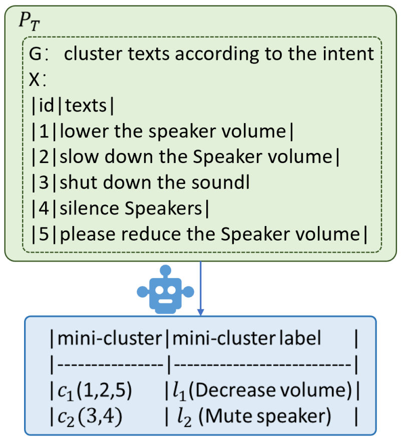
Abstract

Traditional unsupervised clustering methods, which often rely on contrastive training of embedders, suffer from a lack of label knowledge, resulting in suboptimal performance. Furthermore, the presence of potential false negatives can destabilize the training process. Hence, we propose to improve clustering with **P**ositive **P**airs generated from **L**LM-driven **L**abels (**PPLL**). In the proposed framework, LLM is initially employed to cluster the data and generate corresponding mini-cluster labels. Subsequently, positive pairs are constructed based on these labels, and an embedder is trained using BYOL to obviate the need for negative pairs. Following training, the acquired label knowledge is integrated into K-means clustering. This framework enables the integration of label information throughout the training and inference processes, while mitigating the reliance on negative pairs. Additionally, it generates interpretable labels for improved understanding of clustering results. Empirical evaluations on a range of datasets demonstrate that our proposed framework consistently surpasses state-of-the-art baselines, achieving superior performance, robustness, and computational efficiency for diverse text clustering applications.[1]

1 Introduction

Unsupervised text clustering has numerous applications across various domains, such as anomaly detection (Chandola et al., 2009), topic modeling (Meng et al., 2022), and community detection (Su et al., 2024). The typical workflow begins by obtaining text embeddings using pre-trained embedder models (Su et al., 2023; Liu, 2019). These embeddings are then subjected to conventional clustering algorithms, such as K-means or agglomerative clustering (Day and Edelsbrunner, 1984), to perform the clustering task.

[*]Corresponding author
[1]https://github.com/thelittleTom/PPLL



P_T

G: cluster texts according to the intent
X:
id	texts
1	lower the speaker volume
2	slow down the Speaker volume
3	shut down the soundl
4	silence Speakers
5	please reduce the Speaker volume

mini-cluster	mini-cluster label
c_1(1,2,5)	l_1(Decrease volume)
c_2(3,4)	l_2 (Mute speaker)

Figure 1: The superior clustering and labeling ability exhibited by LLM.

Recent advancements in large language models (LLMs) (Achiam et al., 2023; Brown et al., 2020; Dubey et al., 2024) have spurred the development of new clustering methods that leverage LLMs to improve performance. A promising direction is using LLMs to provide the analysis insights from the texts (Zhang et al., 2023; De Raedt et al., 2023) to finetune embedder models to enhance clustering performance. However, this approach neglects the importance of labels during training and inference and the clustering outputs lack interpretability, requiring users to spend considerable time understanding the results. Moreover, these methods typically rely on contrastive learning (Gao et al., 2021) for fine-tuning the embedder, which introduces a dependency on negative samples, resulting in unstable performance (Grill et al., 2020). Another class of methods directly employs LLMs for clustering (Wang et al., 2023; Pham et al., 2024), where LLMs generate and refine labels through iterative self-reflection (Asai et al., 2023) and assign labels to individual texts, while this approach can generate more interpretable labels, it incurs significant cost.

To address the challenges mentioned above and harness the inherent clustering capabilities of LLMs, as shown in Figure 1, we introduce PPLL. This framework leverages LLMs for data clustering and label generation, which in turn guide a lightweight embedder to efficiently identify text clusters, as depicted in Figure 2.

Specifically, PPLL consists of three stages. In the first stage, we acquire LLM-generated labels and construct positive pairs. In particular, due to the input length limitations of LLMs, we adopt an entropy-based data batching strategy to divide the data into batches for input. Subsequently, we leverage the LLM to cluster each batch and generate mini-cluster labels. Positive pairs are then constructed based on these labels, enabling us to perform clustering from a macro-level label perspective and incorporate label knowledge into the embedder training process. The second stage is BYOL training and inference. An embedder model is trained using the BYOL method (Grill et al., 2020; Zhang et al., 2021c) with the positive pairs which obviates the need for negative pairs during fine-tuning. Subsequently, both mini-cluster labels and texts are subjected to K-means clustering using this trained model. The inclusion of mini-cluster labels is intended to provide anchor points for the K-means algorithm, thereby enhancing clustering quality. The third stage is category label generation. The mini-cluster labels are input into a large language model to produce human-readable and informative labels for the resulting clusters. Experimental results demonstrate that PPLL significantly outperforms baseline methods in terms of clustering quality. Ablation studies further corroborate the effectiveness of our data batching strategy, the robustness of our training method, and the benefit of using labels.

Our contributions are as follows:

- We propose PPLL, a novel framework that leverages LLMs to generate label-based positive pairs to guide embedder clustering, and produce category labels for each cluster.
- Unlike traditional clustering methods that rely on contrastive learning to train embedders, we explore the use of BYOL to achieve training without negative pairs, mitigating the issue of false negatives.
- Experimental results demonstrate that our approach significantly improves clustering performance, with an average cost of ~$0.28 per run on our datasets using GPT-3.5, while also generating human-interpretable labels.

2 Related Work

Sentence Representation. The ability to effectively represent sentences through embeddings is a key aspect of NLP (Mikolov et al., 2013). With the tremendous success of Pre-trained Language Models (PLMs), recent approaches have focused on generating sentence embeddings by leveraging the embedding of the [CLS] token or applying mean pooling on the final layer of BERT (Reimers and Gurevych, 2019).

Recently, several models (Neelakantan et al., 2022; Wang et al., 2022a; Chuang et al., 2022; Zhuo et al., 2023; Liu et al., 2023) have adopted a contrastive learning framework without the need for labeled data to learn sentence representations. This approach pulls semantically similar sentences closer together while pushing dissimilar ones apart. Cheng et al. (2023) leverages LLMs to get pairwise scores for sentence relationships to provide better supervision signals for supervised contrastive learning of sentence embeddings. Among these, embedding models like Instructor (Su et al., 2023) have demonstrated superior performance on popular benchmarks (Muennighoff et al., 2023). This paper aims to enhance Instructor by LLMs.

Due to the issue of false negatives in contrastive learning, BYOL (Grill et al., 2020; Zhang et al., 2021c) and SCD (Klein and Nabi, 2022) use a two-branch Siamese network to train the embedder without negative pairs. To mitigate the issue of unreliable negative pairs generated by LLMs, we adopt BYOL (Zhang et al., 2021c) as our training method.

Clustering with LLM. Clustering has been extensively studied in both text (Li et al., 2020; Zhong et al., 2021) and image domains (Chang et al., 2017; Yang et al., 2022). Recent advancements in LLMs have spurred the development of numerous clustering methods that leverage LLMs. These methods can be broadly categorized into three groups.

The first group (An et al., 2023; Liang et al., 2024b; Yang et al., 2024; Feng et al., 2024) utilizes LLMs to provide sentence relationships. For instance, in unsupervised clustering, ClusterLLM (Zhang et al., 2023) prompts LLMs with a triplet task to infer sentence relationships, guiding the clustering process.

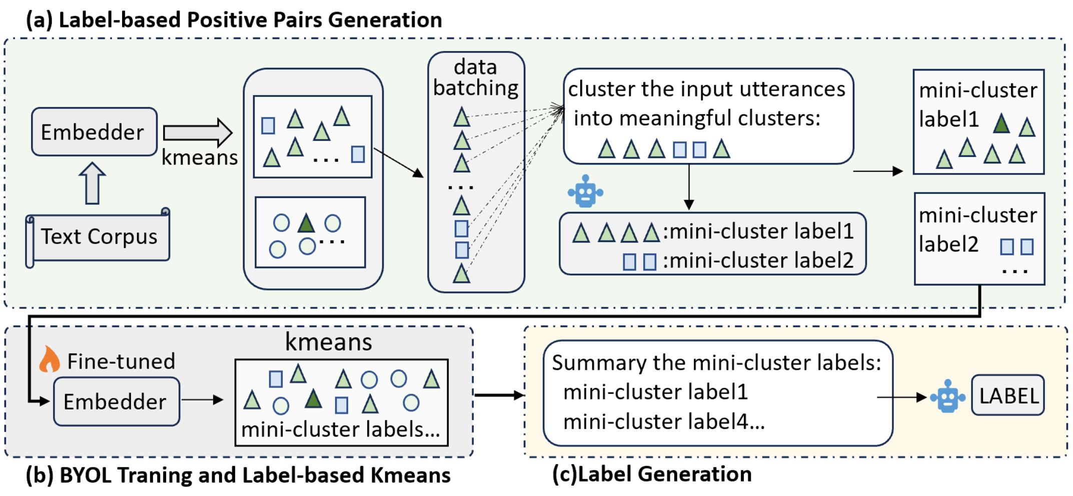


Figure 2: The overview of our framework PPLL. PPLL comprises three stages: (1) Label-based Positive Pair Generation, (2) BYOL Training with Label-based K-means, and (3) Label Generation.

The second group (De Raedt et al., 2023) leverages LLMs to extract implicit information from text. For instance, SynCID (Liang et al., 2024a) employs LLMs to generate descriptions for utterances and designs multiple loss functions to align the representations of descriptions and utterances, thereby enhancing clustering performance. Similarly, Viswanathan et al. (2023) utilizes LLMs to extract keywords from text, improving the expressiveness of embeddings.

The third group (Wan et al., 2024; Lam et al., 2024; Huang and He, 2025) directly employs LLMs for clustering without the need for training. For instance, Pham et al. (2024) utilizes LLMs to generate topics for documents, subsequently deduplicating and assigning these topics. Similarly, Wang et al. (2023) first employs LLMs to generate labels for sampled inputs, then queries the LLM to determine the relationship between the labels and the texts, and finally assigns labels to texts using an integer linear programming.

3 Method

3.1 Preliminary

The input space for text clustering comprises a set of texts X (the corpus), a string G (the goal description, e.g. "according to the intent"), and an integer K representing the desired number of clusters.

The output space includes K subsets of X, each representing a cluster, denoted as C_k where $k \in [K]$. Optionally, the output may also include K strings L_k, where L_k is a label or description of cluster C_k, providing valuable insights and facilitating interpretation of the clustering results.

3.2 Framework Overview

Figure 2 illustrates the three-stage PPLL framework. In **Stage 1** (Section 3.3), we input data batches to an LLM to generate labels for each mini-cluster, and then collect positive pairs based on these mini-cluster labels. In **Stage 2** (Section 3.4), we employ a BYOL approach for training the embedder without negative pairs, incorporating mini-cluster labels into the clustering process. Finally, in **Stage 3** (Section 3.5), interpretable labels are generated for the final clusters.

3.3 Label-based Positive-Pairs Generation

This stage aims to leverage LLM to cluster texts and generate labels, forming positive pairs for subsequent training. The entire process is detailed in Algorithm 1.

3.3.1 Entropy-based data batching

If we obtain all texts within each cluster, we can generate positive pairs by pairing every two texts within the same cluster. However, the K-means clustering results obtained using embeddings are susceptible to inaccuracies, so we subsequently employ an LLM to refine the clustering results by re-clustering the texts within each K-means cluster. Given the limited input length of LLMs, we propose a data batching strategy to group texts into appropriately sized mini-batches for input.

The data batching algorithm contains two steps. **Step 1** (see Algorithm 1 line 1~2): we perform K-means clustering on the texts and calculate the entropy for each text. **Step 2** (see Algorithm 1 line

3~9): within each cluster, texts are sorted by their entropies. Subsequently, the top and bottom γ texts are grouped into a mini-batch X_b, resulting in a batch size of 2γ. This process is repeated until all texts in the cluster are processed. This strategy ensures that both easily- and difficult-to-classify samples are included in each mini-batch, aiding the LLM in accurate clustering and label generation.

In particular, in step 1, after K-means clustering, we use the Student's t-distribution to compute the probability of assigning the sample x_i to each cluster k (Xie et al., 2016; Zhang et al., 2023):

$$p_{ik} = \frac{\left(1 + \left\|z_i - \mu_k\right\|^2 / \alpha\right)^{-\frac{\alpha+1}{2}}}{\sum_{k'} \left(1 + \left\|z_i - \mu_{k'}\right\|^2 / \alpha\right)^{-\frac{\alpha+1}{2}}} \quad (1)$$

where α represents the degrees of freedom in the Student's t-distribution, z_i is the embedding of i-th text, μ_k is the mean vector of all texts in the k-th cluster as the center of the cluster. After obtaining the predictive probabilities, we use the entropy (Lewis, 1995) to measure the uncertainty for each sample x_i:

$$e_i = - \sum_{k=1}^{K_{\text{closest}}} p'_{ik} \log\left(p'_{ik}\right) \quad (2)$$

where $p'_{ik} = \frac{p_{ik}}{\sum_{k'=1}^{K_{\text{closest}}} p_{ik'}}$, K_{closest} is a hyperparameter that is less than or equal to the total number of clusters K. Here, a higher e_i can indicate a higher likelihood of the model incorrectly assigning x_i to the wrong cluster.

3.3.2 LLM-based Positive-Pairs Generation

Now that we have a batch of input texts X_b for LLM to cluster, we prompt the LLM to cluster and label these texts using a prompt P_T (see Appendix Table 13). To accommodate user-specified clustering perspectives, a goal G is incorporated into P_T. The LLM generates cluster assignments and labels for the texts as follows:

$$\phi = P_T(G, X_b) \quad (3)$$

where $\phi = \{(c_{b,j}, l_{b,j})\}_{j=1}^{M_b}$, $c_{b,j}$ denotes the j-th mini-cluster within X_b, $l_{b,j}$ represents the corresponding mini-cluster label (see Figure 1 and Algorithm 1 line 10~14). Each text is thus assigned a mini-cluster label, which is extracted and refined through clustering by LLM. To mitigate the constraint of limited LLM input length and bridge the gap between batches, we treat texts with the same mini-cluster label as belonging to the same mini-cluster, regardless of the batch in which they were clustered. So, all possible unique pairs of texts within each mini-cluster are formed as positive training examples (see Algorithm 1 line 15~18).

Algorithm 1: Label-based Positive-Pairs Generation

```
Input: Texts $X = \{x_i\}_{i=1}^N$, embeddings
       $Z = \{z_i = f(x_i)\}_{i=1}^N$, the half of
       batch size $\gamma$, prompt $P_T$, the cluster
       goal description $G$
Output: Positive Pairs $P$
1  $K$ clusters ← Clustering($Z$);
2  Compute entropy $E$ with Eq 1, Eq 2;
3  sort the texts in each cluster by entropy $E$
4  $Batches$ ← []
5  for c in $K$ sorted clusters do
6      while length(c) ≥ 2*$\gamma$ do
7          $c[0:\gamma]$ + $c[-\gamma:]$ to $Batches$
8          $c \leftarrow c[\gamma:-\gamma]$
9      Append $c$ to $Batches$
10 mini-clusters $C$ ← {}
11 for $X_b$ in $Batches$ do
12     $\phi = P_T(G, X_b)$
13     for $(c_{b,j}, l_{b,j})$ in $\phi$ do
14         Append the texts in $c_{b,j}$ to $C[l_{b,j}]$
15 Positive pairs $P$ ←[]
16 for mini-cluster in $C$ do
17     for (i, j) in combinations(mini-cluster, 2) do
18         Append [i, j] to $P$
```

3.4 BYOL Training and Label-based Kmeans

3.4.1 BYOL for finetuning embedder

We observed that LLM-based clustering often suffers from false negatives, where texts within the same class are erroneously assigned to different clusters. To address this limitation, we adopted BYOL training (Grill et al., 2020; Zhang et al., 2021c), which eliminates the need for explicit negative examples and is more robust to noisy labels.

Here is the BYOL training process (see Figure 3): Two texts x_1 and x_2 from a positive pair are encoded by the online network f_θ and the target network f_ξ, respectively. Both networks are initialized from the same embedder, while ξ is maintained as an exponential moving average (EMA) of θ during training. P_θ denotes a predictor, a multi-layer per-

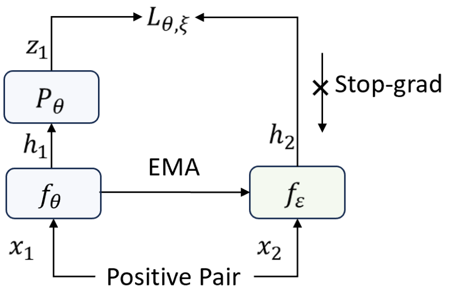


Figure 3: The framework of BYOL.

ceptron applied exclusively to the online network. A stop-gradient operation is applied to the target network. The loss function, $L_{\theta,\xi}$:

$$L_{\theta,\xi}(z_1, h_2) = -< \frac{z_1}{\|z_1\|}, \frac{h_2}{\|h_2\|} > \quad (4)$$

where $z_1 = p_\theta(f_\theta(x_1))$,$h_2 = f_\xi(x_2)$,and $\|\cdot\|$ denotes the $l2 - norm$ and $<,>$ denotes the dot product.

The calculation of loss involves parameters θ, ξ, however, and during training, we only update the parameter θ, as illustrated by the stop-gradient operation in Figure 3. The function f_ξ is decoupled from the optimization graph of $L_{\theta,\xi}$ and will be updated with a weighted moving average of f_θ. The updating dynamics becomes:

$$\begin{aligned} \theta_t &\leftarrow \theta_{t-1} + \nabla_\theta \mathcal{L}_{\theta,\xi}, \\ \xi_t &\leftarrow \delta\xi_{t-1} + (1-\delta)\theta_t. \end{aligned} \quad (5)$$

Here δ is the momentum. At the inference stage, we obtain the representation of a sentence with the online encoder f_θ.

3.4.2 Label-Based Kmeans

After fine-tuning the model using BYOL with positive pairs, we extract embeddings from the trained embedder and apply K-means clustering. To enhance the quality of clustering, we propose a label-based K-means approach that incorporates both the original texts and the LLM-generated mini-cluster labels. By jointly clustering these two representations, we expect the cluster centers to better capture the underlying semantic structure of the data. The final clustering results are obtained by discarding the mini-cluster labels and retaining only the clusters of the original texts.

3.5 Label Generation

After Label-Based K-means clustering, we obtain the set $label_k = \{l_{k,j}\}_j^{N_k}$ by aggregating the mini-cluster labels of texts for each class, where N_k represents the number of mini-cluster labels in class k. We then filter out mini-cluster labels with a frequency below a threshold of α, retaining only the top most frequent labels $labels_k^{'}$, denoted as:

$$labels_k^{'} = \{l_{k,j} \text{ if count}(l_{k,j}) > \alpha | j = 1, 2, ...N_k\} \quad (6)$$

We then prompt LLMs to generate an interpretable cluster label using a prompt P_G (See Appendix Table 14) with the filtered mini-cluster labels $labels_k^{'}$ within cluster k. The LLM then generates a concise and informative label L_k for cluster k:

$$L_k = P_G(labels_k^{'}) \quad (7)$$

4 Experiments

4.1 Datasets

To evaluate PPLL, we conduct experiments on a diverse set of datasets, including CLINC(I), MTOP(I), Massive(I) (FitzGerald et al., 2022), GoEmo (Demszky et al., 2020), CLINC-Domain, MTOP-Domain, and Massive-Scenario. Each dataset is provided in two sizes, differing in the number of data points while maintaining the same number of clusters. The detailed statistics are reported in Appendix A.1.

4.2 Experiment Details

Hyper-Parameters and Experimental Settings. For entropy-based sampling, we set $K_{\text{closest}} = 25$, $\gamma = 10$. Training is conducted with a single NVIDIA GeForce RTX 4090 (24GB). Besides, our clustering approach utilizes Instructor embeddings (Su et al., 2023), and for our experiments, we employ the ChatGPT (gpt-3.5-turbo-0301), Llama3 (Meta-Llama-3-8B-Instruct) as our LLMs.

To reduce cost, we run PPLL once for each dataset. We then run K-means on datasets for 5 seeds with ground truth K. For real-world applications where the ground truth K is unknown, please refer to Appendix C. More experimental details are provided in the Appendix A.2.

Evaluation We employ two evaluation metrics. Firstly, we utilize Accuracy (ACC), calculated after applying the Hungarian algorithm (Kuhn, 1955) to optimally map predicted clusters to ground truth labels. Secondly, we employ Normalized Mutual Information (NMI), which measures the mutual information between the predicted and true cluster assignments, normalized by their individual entropies. ACC is considered our primary evaluation metric,

Method	CLINC(I)		Massive(I)		MTOP(I)		CLINC(D)		Massive(D)		MTOP(D)		GoEmo		Avg	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
small-scale																
instructor	79.29	92.60	54.08	73.42	33.35	70.63	52.50	56.87	61.81	67.31	90.56	87.30	25.19	21.54	56.68	67.10
sccl	80.85	92.94	54.10	73.90	34.28	73.52	54.22	51.08	61.34	68.69	89.08	84.77	34.33	30.54	58.31	67.92
self-supervise	80.82	93.88	55.07	72.88	34.06	72.50	58.58	60.84	53.97	**71.53**	92.12	88.49	24.11	22.05	56.96	68.89
clusterLLM	82.77	93.88	59.89	76.96	35.84	73.52	52.39	54.98	61.06	68.62	**93.53**	**89.36**	27.49	24.78	59.00	68.87
clusterLLM-Iter	83.80	94.00	60.69	**77.64**	35.04	73.83	51.82	54.81	60.85	68.67	92.13	89.23	26.75	23.89	58.73	68.87
PPLL-llama3	**84.14**	94.37	60.99	76.15	**44.31**	**75.22**	57.96	60.70	57.92	65.13	92.41	88.82	35.36	31.17	61.87	70.22
PPLL-gpt3.5	84.13	**94.53**	**61.21**	75.72	40.10	75.14	**59.04**	**61.36**	**62.37**	68.39	91.74	88.67	**39.82**	**35.00**	**62.63**	**71.26**
large-scale																
instructor	79.52	92.65	54.72	72.29	35.53	70.78	51.74	55.28	56.11	61.86	85.01	83.96	24.02	20.15	55.24	65.28
self-supervise	81.87	93.55	58.30	73.73	35.27	71.88	50.93	53.01	58.14	64.49	89.54	86.69	24.34	21.17	56.91	66.36
clusterLLM	82.29	93.91	57.70	74.24	36.80	73.16	50.12	53.46	58.14	65.50	84.08	84.57	25.23	22.19	56.34	66.72
PPLL-llama3	84.35	94.20	56.82	73.65	39.52	74.00	54.92	60.72	59.64	**66.19**	89.51	88.26	34.38	29.92	59.88	69.56
PPLL-gpt3.5	**86.22**	**94.81**	**59.86**	**75.18**	**42.50**	**75.09**	**57.20**	**61.44**	**60.21**	65.50	**89.94**	**88.87**	**35.85**	**30.13**	**61.68**	**70.15**

Table 1: Results (in %) on multiple datasets. Average over all datasets are shown in the last two columns. Highlights(Underlines) indicate top (second) scores per column.

with higher values indicating superior clustering performance.

4.3 Compared Methods

We compare PPLL with several baseline approaches: Instructor, which directly applies K-means to embeddings extracted from the instructor-large model; ClusterLLM, which prompts LLM (gpt3.5) with a triplet task and then utilizes the labeled data to train the embedder; ClusterLLM-Iter, which runs the ClusterLLM twice; self-supervised clusterLLM, a variant of ClusterLLM that leverages an embedder for pseudo-label assignment; and SCCL (Zhang et al., 2021a), a deep text clustering algorithm equipped with Instructor.

4.4 Main Results

We show main results on multiple datasets in Table 1 with several variants of PPLL: PPLL-gpt3.5 adopts gpt3.5 and PPLL-llama3 adopts llama3 as LLM (see more results in Appendix D). We make the following observation: (1) PPLL-gpt3.5 consistently improves upon Instructor. For example, PPLL-gpt3.5 improves accuracy by 14.63% and NMI by 13.46% on GoEmo(small-scale), and improves accuracy by 11.83% and NMI by 9.98% on GoEmo(large-scale). In contrast, clusterLLM (Zhang et al., 2023), which also leverages GPT-3.5, does not exhibit a significant improvement over Instructor on Clinc(D) (small-scale), Massive(D) (small-scale), and Mtop(D) (large-scale) datasets. (2) Our experimental results demonstrate that the proposed method on gpt3.5 consistently surpasses all baseline methods. Specifically, it outperforms all baselines on a majority of small-scale datasets and on all large-scale datasets. Furthermore, PPLL achieves the best overall performance across all datasets, as evidenced by the highest average scores. (3) PPLL on open-sourced LLM Llama3 also demonstrates promising results. This indicates that PPLL does not purely rely on the powerful text understanding capabilities of close-sourced LLM GPT3.5, highlighting its effectiveness across different LLMs.

4.5 Ablation Study

Data Batching Strategy. To assess the impact of entropy-based sampling, we conduct an ablation study as summarized in Table 2. In the *w/o entropy* configuration, we eliminate the entropy-based data batching component and solely rely on K-means clustering, i.e., we randomly select batches from within each cluster to form the batches for prompting. In the *w/o entropy&kmeans* configuration, we further remove the intra-cluster batching step, resulting in random sampling from the entire dataset.

The ablation study presented in Table 2 reveals a substantial performance degradation when *w/o entropy* or when *w/o entropy&kmeans*, particularly for the primary metric ACC. Furthermore, the pair accuracy in Table 2 demonstrates that our proposed method of combining easy and hard samples, along with the incorporation of prior knowledge (i.e., pre-clustering with K-means), effectively improves the accuracy of LLM-generated labels. Despite achieving a lower pair-acc, the *w/o entropy&kmeans* method demonstrates comparable or even superior overall performance compared to clusterLLM baselines in Table 1, highlighting the robustness of PPLL. However, on the MTOP(I) and Massive(D) datasets, we observe that the *w/o entropy&kmeans* method outperforms *PPLL-gpt3.5* method. This can be attributed to the fact that our method restricts LLM inputs to instances that

Method	CLINC(I)			Massive(I)			MTOP(I)			CLINC(D)		
	ACC	NMI	Pair-Acc	ACC	NMI	Pair-Acc	ACC	NMI	Pair-Acc	ACC	NMI	Pair-Acc
PPLL-gpt3.5	**84.13**	94.53	**92.06**	**61.21**	75.72	**77.26**	40.10	75.14	83.33	**59.04**	**61.36**	**80.75**
- w/o entropy	83.87	**94.62**	90.24	60.20	**76.16**	76.95	40.50	74.61	**84.06**	57.24	60.92	75.24
- w/o entropy&kmeans	80.19	93.37	10.71	58.10	75.77	18.67	**43.44**	**75.26**	46.32	53.35	58.12	45.76

Method	Massive(D)			MTOP(D)			GoEmo			Avg		
	ACC	NMI	Pair-Acc	ACC	NMI	Pair-Acc	ACC	NMI	Pair-Acc	ACC	NMI	Pair-Acc
PPLL-gpt3.5	62.37	68.39	**79.92**	**91.74**	**88.67**	94.27	**39.82**	**35.00**	**59.86**	**62.45**	**71.17**	**81.06**
- w/o entropy	58.41	67.87	79.36	89.80	87.69	**94.76**	37.51	34.38	48.15	61.08	70.90	78.39
- w/o entropy&kmeans	**65.00**	**70.44**	49.09	89.67	87.52	64.97	25.00	22.17	9.79	59.25	68.95	35.04

Table 2: Ablation results on data batching strategy on small scale datasets. Pair-Acc refers to the accuracy of positive pairs, indicating the proportion of positive pairs that actually belong to the same ground class.

Method	CLINC(I)		Massive(I)		MTOP(I)		CLINC(D)		Massive(D)		MTOP(D)		GoEmo		Avg	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
PPLL-self-γ=1	80.33	93.40	58.84	74.98	38.50	74.60	57.70	59.92	57.79	66.93	88.19	87.86	37.09	33.92	59.78	70.23
PPLL-llama3-γ=1	81.09	93.75	**62.54**	76.17	**48.89**	76.29	55.54	58.81	**62.00**	65.93	88.95	86.43	34.32	30.29	61.90	69.67
PPLL-llama3-γ=2	83.88	94.37	60.75	76.09	46.30	**76.79**	55.04	59.35	59.14	66.10	86.60	86.92	**40.59**	**34.28**	61.76	70.56
PPLL-llama3-γ=5	**84.21**	**94.52**	61.29	**76.59**	45.94	75.77	**60.14**	**61.15**	61.64	**67.12**	86.73	86.95	37.67	32.10	**62.52**	**70.60**
PPLL-llama3-γ=10	84.14	94.37	60.99	76.15	44.31	75.22	57.96	60.70	57.92	65.13	**92.41**	**88.82**	35.36	31.17	61.87	70.22

Table 3: Ablation on Input size to LLM on small scale datasets.

Gold Label	Generated Label
make call	Phone call
greeting	Greetings
how old are you	Age-related queries
who do you work for	Workplace Inquiries
what are your hobbies	Interest and leisure activities
expiration date	Credit card expiration date

Table 4: Examples of generated labels on CLINC(I) (small scale).

are considered to be within the same cluster by the embedding model. In contrast, the *w/o entropy&kmeans* method, lacking this prior knowledge, is more likely to group together instances that should belong to the same class but are incorrectly classified as separate classes due to the embedder's limitations.

Impact of Input size to LLM. This section presents an ablation study on the input size to the LLM, as shown in Table 3. Specifically, we employ Llama3 as the LLM for prediction and vary the input size by adjusting the parameter γ. There are several observations from the results.

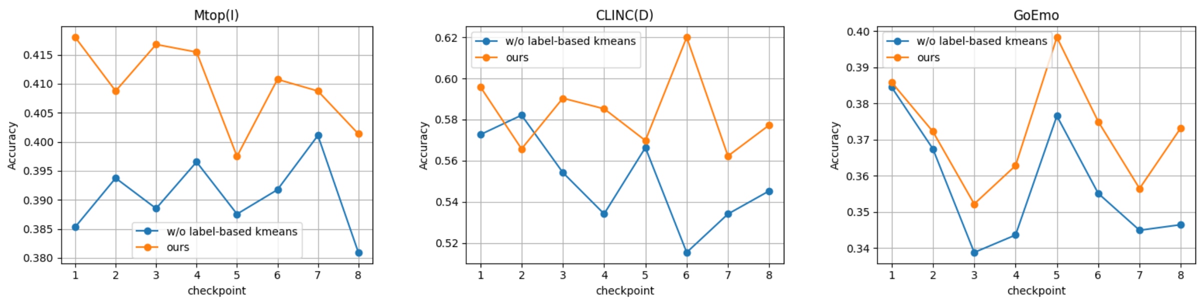
Firstly, PPLL demonstrates consistent performance across various settings of γ, highlighting its robustness to different input sizes.

Secondly, the results show that Llama3 achieves better performance with γ=5 than with γ=10 (except for MTOP(D)), indicating that the model's performance may deteriorate when handling excessively large amounts of text. This observation underscores the significance of data batching.

Thirdly, a comparison between our proposed method and *PPLL-self-γ=1* (a self-supervised baseline where LLM are removed with γ=1) reveals a substantial improvement in terms of ACC and NMI across all datasets. This finding highlights the significant contribution of LLM-derived knowledge to our model's performance.

Fourth, when *PPLL-llama3-γ=1*, the small batch size causes LLM to assign all input pairs to a single mini-cluster. Nevertheless, by clustering texts based on mini-cluster labels obtaining more positive pairs, PPLL avoids degenerating to *PPLL-self-γ=1*. Comparisons between *PPLL-self-γ=1* and *PPLL-llama3-γ=1* demonstrate that incorporating LLM knowledge generally enhances performance by introducing valuable label knowledge. However, in some datasets, *PPLL-llama3-γ=1* underperforms *PPLL-self-γ=1*. This may be attributed to the limited input size when γ=1, resulting in insufficient samples for LLM to compare and consequently lower positive pair accuracy. See Appendix G for more analysis on γ.

Label-Based kmeans. Figure 4 presents an ablation study designed to assess the impact of label-



(a) the accuracy of Mtop(I). (b) the accuracy of Clinc(D). (c) the accuracy of GoEmo.

Figure 4: Ablation result of Label-Based Kmeans on small scale datasets.

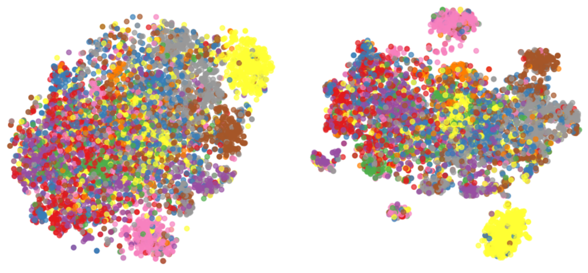
based K-means on clustering performance. We compare the accuracy of different checkpoints on three datasets when using GPT-3.5 as the LLM, with and without the inclusion of mini-cluster label information. Figure 4 and Table 9 in Appendix demonstrate that incorporating label information into K-means can significantly improve clustering accuracy. However, the performance can degrade in cases where the mini-cluster labels are of low quality. Nevertheless, our proposed evaluation metric in Appendix B allows us to select the most effective configuration, considering both label-based and label-free K-means.

4.6 Generated New Labels

To study the quality of labels produced by PPLL for the resulting clusters, we conduct a comparative analysis between the gold standard labels and PPLL generated labels (section 3.5) on the CLINC dataset. Table 4 compares label generation across different categories. For clusters with specific intents (e.g., *make call*, *greeting*), PPLL can accurately generate their corresponding intent labels. For general user questions, PPLL effectively condenses queries into high-level intent labels. For example, *Who do you work for* is transformed into *Workplace Inquiries*. Meanwhile, the generated labels effectively integrate additional cluster details, leading to more precise and informative intent labels. For example, the gold label *expiration date* is refined to *Credit card expiration date*. Our findings indicate that PPLL, which leverages the power of LLMs, is capable of uncovering latent semantic meanings within text, enabling the generation of high-quality, interpretable labels.

4.7 Visualization

We visualize the embeddings of Instructor and PPLL on the GoEmo with the t-SNE technique in Figure 5. The figure shows the embeddings generated by Instructor are unable to effectively differentiate between different clusters. However, PPLL enhances the cohesiveness of embeddings within the same cluster while simultaneously separating different clusters. This indicates that PPLL enables the model to extract goal attributes from each text, clustering content of the same cluster within the scope of labels generated by the LLM. See Appendix F for more visualizations.



(a) Instructor. (b) PPLL.

Figure 5: Scatter plots for t-SNE of embeddings on GoEmo (small-scale)

5 Conclusion

We introduce PPLL that leverages the clustering and label generation capabilities of LLMs to guide embedders in unsupervised clustering for various tasks. By prompting LLM to generate mini-cluster labels, we enable the construction of positive pairs based on these labels. These mini-cluster labels are then integrated into K-means to refine cluster centers in order to further incorporate label knowledge. PPLL also generates semantically rich and interpretable labels for each cluster. Furthermore, we train the embedder using the BYOL method, which obviates the need for negative pairs, enhancing its robustness. Extensive experiments demonstrate that PPLL not only enhances clustering quality but also generates appropriate labels.

Limitations

Despite the promising results obtained by our method, it is important to acknowledge several limitations. First, as discussed in Section 4.5, our method's reliance on LLM-generated mini-cluster labels for K-means introduces a potential drawback: suboptimal LLM-generated labels can degrade the overall performance. However, as detailed in Appendix B, we propose a strategy to determine whether to incorporate label-based K-means, mitigating the negative impact of potentially poor LLM-generated labels. Second, PPLL is primarily designed for text data, limiting its applicability to multimodal scenarios. Third, PPLL incurs some computational overhead due to its reliance on LLMs for dataset labeling and training. However, the analysis in the Appendix H demonstrates that the overall cost remains relatively low and within acceptable limits. Last, PPLL relies on feedback from the LLM, which may introduce potential risks of data leakage and other security concerns.

Acknowledgments

This work is supported by the Regional Innovation and Development Joint Fund of the National Natural Science Foundation of China (U22A6001), the National key R&D Plan Program of China (2022YFC3601604) and Zhejiang Province Elite Program Project (2023C03162).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Wenbin An, Wenkai Shi, Feng Tian, Haonan Lin, QianYing Wang, Yaqiang Wu, Mingxiang Cai, Luyan Wang, Yan Chen, Haiping Zhu, et al. 2023. Generalized category discovery with large language models in the loop. *arXiv preprint arXiv:2312.10897*.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3).

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.

Qinyuan Cheng, Xiaogui Yang, Tianxiang Sun, Linyang Li, and Xipeng Qiu. 2023. Improving contrastive learning of sentence embeddings from ai feedback. *arXiv preprint arXiv:2305.01918*.

Yung-Sung Chuang, Rumen Dangovski, Hongyin Luo, Yang Zhang, Shiyu Chang, Marin Soljačić, Shang-Wen Li, Wen-tau Yih, Yoon Kim, and James Glass. 2022. Diffcse: Difference-based contrastive learning for sentence embeddings. *arXiv preprint arXiv:2204.10298*.

William HE Day and Herbert Edelsbrunner. 1984. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1):7–24.

Maarten De Raedt, Fréderic Godin, Thomas Demeester, and Chris Develder. 2023. Idas: Intent discovery with abstractive summarization. *arXiv preprint arXiv:2305.19783*.

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, Online. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Zijin Feng, Luyang Lin, Lingzhi Wang, Hong Cheng, and Kam-Fai Wong. 2024. Llmedgerefine: Enhancing text clustering with llm-based boundary point refinement. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18455–18462.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron

Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, et al. 2022. Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. *arXiv preprint arXiv:2204.08582*.

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, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284.

Alexander Hoyle, Rupak Sarkar, Pranav Goel, and Philip Resnik. 2023. Natural language decompositions of implicit content enable better text representations. *arXiv preprint arXiv:2305.14583*.

Chen Huang and Guoxiu He. 2025. Text clustering as classification with llms. *Preprint*, arXiv:2410.00927.

Fan Huang, Haewoon Kwak, and Jisun An. 2023. Is chatgpt better than human annotators? potential and limitations of chatgpt in explaining implicit hate speech. In *Companion Proceedings of the ACM Web Conference 2023*, WWW '23, page 294–297. ACM.

Tassilo Klein and Moin Nabi. 2022. Scd: Self-contrastive decorrelation for sentence embeddings. *arXiv preprint arXiv:2203.07847*.

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

Michelle S. Lam, Janice Teoh, James A. Landay, Jeffrey Heer, and Michael S. Bernstein. 2024. Concept induction: Analyzing unstructured text with high-level concepts using lloom. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, page 1–28. ACM.

David D Lewis. 1995. A sequential algorithm for training text classifiers: Corrigendum and additional data. In *Acm Sigir Forum*, volume 29, pages 13–19. ACM New York, NY, USA.

Junnan Li, Pan Zhou, Caiming Xiong, and Steven CH Hoi. 2020. Prototypical contrastive learning of unsupervised representations. *arXiv preprint arXiv:2005.04966*.

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 7838–7851.

Jiduan Liu, Jiahao Liu, Qifan Wang, Jingang Wang, Wei Wu, Yunsen Xian, Dongyan Zhao, Kai Chen, and Rui Yan. 2023. Rankcse: Unsupervised sentence representations learning via learning to rank. *arXiv preprint arXiv:2305.16726*.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364.

Yu Meng, Yunyi Zhang, Jiaxin Huang, Yu Zhang, and Jiawei Han. 2022. Topic discovery via latent space clustering of pretrained language model representations. In *Proceedings of the ACM web conference 2022*, pages 3143–3152.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. MTEB: Massive Text Embedding Benchmark. *Preprint*, arXiv:2210.07316.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. 2022. Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005*.

Soham Parikh, Quaizar Vohra, Prashil Tumbade, and Mitul Tiwari. 2023. Exploring zero and few-shot techniques for intent classification. *arXiv preprint arXiv:2305.07157*.

Letian Peng, Yuwei Zhang, and Jingbo Shang. 2024. Controllable data augmentation for few-shot text mining with chain-of-thought attribute manipulation. *Preprint*, arXiv:2307.07099.

Chau Minh Pham, Alexander Hoyle, Simeng Sun, Philip Resnik, and Mohit Iyyer. 2024. Topicgpt: A prompt-based topic modeling framework. *Preprint*, arXiv:2311.01449.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *Preprint*, arXiv:1908.10084.

Gaurav Sahu, Olga Vechtomova, Dzmitry Bahdanau, and Issam H. Laradji. 2023. Promptmix: A class boundary augmentation method for large language model distillation. *Preprint*, arXiv:2310.14192.

Dominik Stammbach, Vilém Zouhar, Alexander Hoyle, Mrinmaya Sachan, and Elliott Ash. 2023. Revisiting automated topic model evaluation with large language models. *Preprint*, arXiv:2305.12152.

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. *Preprint*, arXiv:2212.09741.

Xing Su, Shan Xue, Fanzhen Liu, Jia Wu, Jian Yang, Chuan Zhou, Wenbin Hu, Cecile Paris, Surya Nepal, Di Jin, Quan Z. Sheng, and Philip S. Yu. 2024. A comprehensive survey on community detection with deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):4682–4702.

Vijay Viswanathan, Kiril Gashteovski, Carolin Lawrence, Tongshuang Wu, and Graham Neubig. 2023. Large language models enable few-shot clustering. *Preprint*, arXiv:2307.00524.

Mengting Wan, Tara Safavi, Sujay Kumar Jauhar, Yujin Kim, Scott Counts, Jennifer Neville, Siddharth Suri, Chirag Shah, Ryen W White, Longqi Yang, Reid Andersen, Georg Buscher, Dhruv Joshi, and Nagu Rangan. 2024. Tnt-llm: Text mining at scale with large language models. *Preprint*, arXiv:2403.12173.

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

Yufei Wang, Can Xu, Qingfeng Sun, Huang Hu, Chongyang Tao, Xiubo Geng, and Daxin Jiang. 2022b. Promda: Prompt-based data augmentation for low-resource nlu tasks. *Preprint*, arXiv:2202.12499.

Zihan Wang, Jingbo Shang, and Ruiqi Zhong. 2023. Goal-driven explainable clustering via language descriptions. *Preprint*, arXiv:2305.13749.

Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487. PMLR.

Chen Yang, Bin Cao, and Jing Fan. 2024. Tec: A novel method for text clustering with large language models guidance and weakly-supervised contrastive learning. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 18, pages 1702–1712.

Mouxing Yang, Yunfan Li, Peng Hu, Jinfeng Bai, Jiancheng Lv, and Xi Peng. 2022. Robust multi-view clustering with incomplete information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):1055–1069.

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, Online. Association for Computational Linguistics.

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

Haode Zhang, Haowen Liang, Liming Zhan, Albert Y. S. Lam, and Xiao-Ming Wu. 2024a. Revisit few-shot intent classification with plms: Direct fine-tuning vs. continual pre-training. *Preprint*, arXiv:2306.05278.

Yan Zhang, Ruidan He, Zuozhu Liu, Lidong Bing, and Haizhou Li. 2021c. Bootstrapped unsupervised sentence representation learning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5168–5180.

Yuwei Zhang, Siffi Singh, Sailik Sengupta, Igor Shalyminov, Hang Su, Hwanjun Song, and Saab Mansour. 2024b. Can your model tell a negation from an implicature? unravelling challenges with intent encoders. *Preprint*, arXiv:2403.04314.

Yuwei Zhang, Zihan Wang, and Jingbo Shang. 2023. Clusterllm: Large language models as a guide for text clustering. *arXiv preprint arXiv:2305.14871*.

Huasong Zhong, Jianlong Wu, Chong Chen, Jianqiang Huang, Minghua Deng, Liqiang Nie, Zhouchen Lin, and Xian-Sheng Hua. 2021. Graph contrastive clustering. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9224–9233.

Ruiqi Zhong, Charlie Snell, Dan Klein, and Jacob Steinhardt. 2022. Describing differences between text distributions with natural language. *Preprint*, arXiv:2201.12323.

Ruiqi Zhong, Peter Zhang, Steve Li, Jinwoo Ahn, Dan Klein, and Jacob Steinhardt. 2023. Goal driven discovery of distributional differences via language descriptions. *Preprint*, arXiv:2302.14233.

Wenjie Zhuo, Yifan Sun, Xiaohan Wang, Linchao Zhu, and Yi Yang. 2023. Whitenedcse: Whitening-based contrastive learning of sentence embeddings. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12135–12148.

A Experimental Details

A.1 Dataset Details

The CLINC dataset, originally for out-of-scope detection, was adapted by focusing on in-domain utterances to evaluate intent clustering. A domain discovery dataset, CLINC(D), was also created using domains as labels. Similarly, Massive(I)/(D) and MTOP(I)/(D) from MTEB were adapted for clustering by removing low-instance intents and keeping only English data. The train and test splits of all datasets were used as large- and small-scale datasets, respectively. The GoEmotions dataset, for fine-grained emotion detection, was also used for evaluation.

Table 5 reports the detailed statistics for the CLINC(I), MTOP(I), Massive(I), GoEmo, CLINC-Domain, MTOP-Domain, and Massive-Scenario datasets. These datasets cover intent classification, emotional clustering, and domain-specific scenarios.

Task	Name	#Clusters	#data(small)	#data(large)
Intent	CLINC(I)	150	4,500	15,000
	MTOP(I)	102	4,386	15,638
	Massive(I)	59	2,974	11,510
Emotion	GoEmo	27	5,940	23,485
Domain	CLINC(D)	10	4,500	15,000
	MTOP(D)	11	4,386	15,667
	Massive(D)	18	2,974	11,514

Table 5: Dataset Statistics.

A.2 Implementation Details

To assess the performance under different data scales, we conduct experiments on both large and small datasets. For large datasets, to reduce computational cost, we employ a pre-clustering step using K-means to partition the data into Q clusters. Subsequently, we select the data point closest to the centroid from each cluster to form a smaller dataset for subsequent analysis. For instance, in the GoEmo dataset, we initially cluster 23,485 data points into 23485/4≈5871 clusters and then proceed with our proposed method using the resulting 5,871 data points. During the final clustering output phase, the class labels of the Q cluster centers are propagated to their corresponding neighboring nodes.

We fine-tune the embedder on the dataset for a single epoch, adopting hyperparameters largely consistent with those used in Zhang et al. (2021c). The $K_{closest}$ hyperparameter was set to a fixed value of 25 for different datasets, which is an empirical parameter derived from our experimental results.

We evaluate the model's performance in terms of Overall Silhouette Score and pair-score (Appendix B) to determine the optimal learning rate (selected from 1e-4, 5e-5) and the need for label-based K-means. This evaluation is conducted without relying on ground truth labels. Moreover, to mitigate overfitting and reduce the impact of potential noise in the LLM-generated labels, we evaluate the model every quarter of an epoch in terms of Overall Silhouette Score and pair-score. This results in a total of 16 potential configurations for each dataset, from which the optimal one in terms of Overall Silhouette Score and pair-score is automatically selected as output.

B Overall Silhouette Score and Pair-Score

To automatically select the relatively better configuration in the absence of ground truth labels, we propose a novel evaluation method. This method involves two primary metrics. The first one is Overall Silhouette Score (ss), which reflects the cohesiveness of the clusters. The second evaluation metric is *pair-score*, which is designed to assess the model's accuracy. To calculate the pair-score, we first obtain embeddings using a pre-trained embedder. Subsequently, we perform K-means clustering on these embeddings to form extremely small clusters, each containing an average of only two texts. We hypothesize that these mini-clusters are the same true class, and thus, the pairs within these clusters are considered true positive pairs. For each clustering result of the configuration to be evaluated, we assess whether these positive pairs are assigned to the same cluster to obtain the *pair-score*:

$$pair-score = \frac{\sum_{i=1}^{N} \mathbb{I}(C[P_{i,0}] = C[P_{i,1}])}{N} \quad (8)$$

where $C[P_{i,0}]$ denotes the predicted cluster assignment of $P_{i,0}$. $P_{i,0}$ and $P_{i,1}$ form the i-th pair. $\mathbb{I}_{\{\cdot\}}$ is an indicator function.

Our overall evaluation process is as follows:

Step 1, we calculate the ss and pair-score for each configuration, and then compute the score as

$$score = \text{pair-score}*(1 + ss) \quad (9)$$

Step 2, for results obtained with different learning rates, we first select the top two based on the *score*. Then, we compare the *pair-score* of these top two

scale	small				large			
Method	CLINC(D)		Massive(D)		CLINC(I)		Massive(D)	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
Overall Silhouette Score&Pair-Score	**59.04**	**61.37**	**61.14**	**67.79**	**86.22**	**94.82**	**60.21**	**65.50**
Overall Silhouette Score	58.21	60.13	61.02	67.74	84.45	94.53	57.69	64.97

Table 6: The actual clustering performance of configurations selected by different evaluation methods without ground truth.

results (since the cohesion, reflected by ss, may vary due to different learning rates, it is unfair to consider ss in this comparison). So we select the result with the highest *pair-score*. As shown in Table 6, employing Overall Silhouette Score and Pair-Score for evaluation is more effective in identifying superior configurations among the 16 candidates compared to using Overall Silhouette Score alone, thus enabling more accurate unsupervised evaluation of checkpoint performance.

C Real-world Applications

The proposed method is predicated on the assumption of a known number of clusters, a condition that is not reflective of most real-world applications where the true number of clusters is often undetermined. To solve this issue, we utilize the filtering strategy (Zhang et al., 2021b) to estimate K.

Initially, we estimate an upper bound on the number of clusters, denoted as $K^{'}$. Subsequently, we utilize the pre-trained Instructor model to extract embeddings from the dataset. These embeddings are then clustered using the K-means algorithm. We subsequently filter the resulting clusters, retaining only those that are densely populated and exhibit well-defined boundaries. Smaller and less distinct clusters are discarded. Formally, the filter process is defined as follows:

$$K_I = \sum_{i=1}^{K^{'}} \mathbb{I}(|S_i| > \frac{N}{K^{'}}) \qquad (10)$$

where $|S_i|$ is the i-th grouped cluster size, N is the size of the dataset, $\mathbb{I}(\cdot)$ is the indicator function, which returns 1 if the condition within the parentheses is satisfied and 0 otherwise.

Having obtained an initial estimate of K_I, we proceed with data batching and prompt the LLM to generate mini-cluster labels. Subsequently, we reapply the filtering process using the mini-cluster labels to refine the estimated number of clusters, resulting in K_{llm}. We can further refine our estimation by applying the filtering process to the mini-cluster labels using the trained embedder, resulting in an additional estimate denoted as K_e. As shown in Table 7, our model provides the most accurate estimate of the number of clusters, with minimal error, outperforming the ClusterLLM model which relies on GPT-3.5 for estimation. This empirically validates the significance of incorporating label knowledge and the effectiveness of training the embedder using positive pairs, enabling the embedder to better capture the underlying data distribution and facilitate accurate clustering.

Moreover, we conducted experiments using the estimated K_e to explore the influence of the number of clusters. Results in Table 8 show that our model achieves competitive performance, even without prior knowledge of the true number of clusters, highlighting the robustness of our proposed method.

D More result of PPLL

Table 10 presents the experimental results of PPLL using Mistral-7B-Instruct-v0.2 and gpt4o. As demonstrated, PPLL achieves superior performance compared to other methods. Moreover, a comparison with gpt-3.5 reveals a positive correlation between the capability of the LLM used and the effectiveness of PPLL-based clustering. Importantly, PPLL still exhibits remarkable performance even when utilizing a less capable open-source 7B LLM.

E The variance of PPLL's results

Table 11 reveals that PPLL exhibits a similar level of stability to ClusterLLM-Iter, as evidenced by the comparable variance in their K-means clustering results over five random seeds (values are scaled by 100 in the table), while achieving higher ACC and NMI.

Method	CLINC(I)		MTOP(I)		MTOP(D)	
	K(Pred)	Error	K(Pred)	Error	K(Pred)	Error
clusterLLM	142	5.33	92	9.80	18	63.64
Instructor-K_I	169	12.67	118	15.69	14	27.27
ours-K_{llm}	161	7.33	105	2.94	13	14.29
ours-K_e	**157**	**4.67**	**104**	**1.96**	**11**	**0.0**

Table 7: The results of predicting K with an unknown number of clusters. We set $K^{'}$ as three times of the ground truth number of clusters during clustering and run K-means on datasets for 5 seeds to get more accurate K.

Cluster Num K	CLINC(I)		MTOP(I)		MTOP(D)	
	ACC	NMI	ACC	NMI	ACC	NMI
K=GT#clusters	84.13	94.53	40.10	75.14	91.74	88.67
K=K_e	83.80	94.40	40.33	74.82	90.88	88.45

Table 8: Experimental results of different cluster number K. "K=GT#clusters" means K is the ground truth number of clusters.

F More Results of Feature Visualization

We also visualize the learned embeddings of Instructor and our method on the Mtop(I) (small-scale) and Mtop(D) (small-scale) dataset with the t-SNE technique in Figure 6. We can still see that our method significantly improves the Instructor's ability to form compact and distinct clusters.

G More analysis on γ

In the table 12, we present the accuracy of the positive pairs generated by PPLL in Stage 1. Here, γ=1 corresponds to processing two texts per batch by LLM, thus limiting the comparison to pairwise interactions, akin to the approach in ClusterLLM (Zhang et al., 2023). Conversely, γ=5 corresponds to processing ten texts per batch. This comprehensive global comparison enables the LLM to develop a more nuanced understanding of even challenging samples, leading to improved clustering accuracy and, consequently, a higher-quality training dataset for the embedder. Table 12 shows that increasing the number of texts processed per batch enhances the accuracy of the positive pairs generated by the LLM. It is precisely this extensive comparison that allows PPLL to effectively enhance clustering performance, thus demonstrating the efficacy of our proposed method.

H Discussion on Computational Costs

Utilizing GPT-3.5 as the LLM in Stage 1 for Equation 3 with γ=10 incurs an average cost of \$0.0012 per batch, resulting in an average cost of \$0.25422 for a small-scale dataset (average length of 4,237 in our experiments). Subsequently, training the embedder for one epoch using BYOL takes 4 minutes with a single NVIDIA GeForce RTX 4090 (24GB), followed by an average cost of \$0.03 for Stage 3 using GPT-3.5. Thus, PPLL demonstrates a relatively low computational cost for small-scale datasets. For large-scale datasets, the approach detailed in the Appendix A.2 can be used to reduce them to a manageable size for annotation and training, thus maintaining a similar computational cost to small-scale datasets. Overall, PPLL demonstrates relatively low computational overhead, achieving both enhanced clustering quality and the generation of appropriate labels at a negligible cost.

I More Related Work

LLM as generator Recent works (Hoyle et al., 2023; Parikh et al., 2023) have leveraged the generative capabilities of large language models (LLMs) for a wide range of applications. Some studies, including Zhong et al. (2022) and Zhong et al. (2023), have employed LLMs to generate datasets. For instance, Zhang et al. (2024b) designed a specialized dataset using LLMs to improve embedders' understanding of two crucial real-world semantic concepts: negation and implicature. Some studies (Zhang et al., 2024a; Wang et al., 2022b), leverage LLMs for text augmentation. For instance, Peng et al. (2024) employs LLMs to extract attributes from text using LLMs, and modify these attributes

Method	Mtop(I)	CLINC(D)	GoEmo
w/o label-based kmeans	33.35	52.50	25.19
label-based kmeans	33.57	57.82	27.41

Table 9: Comparison of methods with and without label-based K-means clustering when the embedder is not finetuned by BYOL (Accuracy).

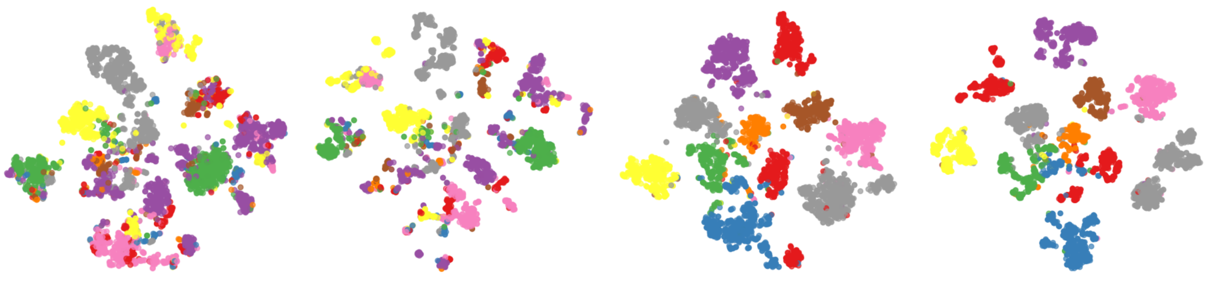
Method	CLINC(I)		Massive(I)		MTOP(I)		CLINC(D)		Massive(D)		MTOP(D)		GoEmo		Avg	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
PPLL-Mistral	84.22	94.30	60.03	76.00	41.46	74.85	58.40	58.66	58.02	68.21	92.71	88.29	35.45	31.71	61.47	70.29
PPLL-gpt4o	86.49	95.09	62.70	77.62	43.86	76.69	60.54	60.21	62.33	69.79	94.21	90.40	39.67	35.60	64.26	72.20

Table 10: Results (in %) on multiple small scale datasets. Average over all datasets are shown in the last two columns.

to generate diverse synthetic sentences, thereby enriching the training data. Similarly, Sahu et al. (2023) utilizes LLMs to generate challenging samples to assist embedders in identifying class boundaries. Additionally, LLMs have been employed for data annotation, as demonstrated in Huang et al. (2023), which highlights the powerful capabilities of ChatGPT in this task. Some studies, such as Stammbach et al. (2023), have employed LLMs for evaluation tasks. For instance, they designed tasks to assess the quality of generated topics and found that LLMs can provide evaluations that align well with human judgments, outperforming existing automated metrics. This paper concentrates on utilizing LLMs for data clustering and label generation.

J Prompt Templates

See Table 13, 14 for the prompt templates that we used in our framework.



(a) Mtop(I)-Instructor. (b) Mtop(I)-ours. (c) Mtop(D)-Instructor. (d) Mtop(D)-ours.

Figure 6: Scatter plots for t-SNE of embeddings on small scale datasets.

Method	CLINC(I)		Massive(I)		MTOP(I)		CLINC(D)		Massive(D)		MTOP(D)		GoEmo	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
clusterLLM-Iter	0.41	0.21	0.96	0.21	0.97	0.79	1.91	1.15	4.33	1.59	3.81	1.21	1.76	0.68
PPLL-llama3	0.79	0.13	1.44	0.23	1.17	0.31	2.84	1.05	1.25	0.76	2.73	1.10	2.31	0.40
PPLL-gpt3.5	0.48	0.10	2.03	0.51	1.43	0.61	2.51	1.60	1.25	0.76	3.17	1.10	1.09	0.52

Table 11: The variance of results, derived from five independent runs with different random seeds on the small-scale dataset.

Method	CLINC(I)	Massive(I)	MTOP(I)	CLINC(D)	Massive(D)	MTOP(D)	GoEmo	Avg
llama3-$\gamma = 1$	71.41	62.75	81.26	44.02	65.16	62.58	17.16	57.76
llama3-$\gamma = 5$	87.85	73.93	86.33	66.97	78.94	94.31	26.72	73.58

Table 12: Pair accuracy (Pair-acc) obtained in Stage 1 of PPLL under different settings of γ.

Prompt template P_T for generating mini-clusters and mini-cluster labels.

Instruction
##Context
- *Goal* Your goal is to cluster the input utterances into meaningful categories ****{according}****.
- *Data* The input data will be a markdown table with utterances including the following columns:
- **id** utterance index.
- **utterance** utterance.

##Requirements

### Format
- Output clusters as a **markdown table**with each row as a category with the following columns:
- **id**: all the utterance ids associated with this category
- **description**: the **{goal}** of the category that should be less than **4** words
Here is an example of your output:
"'markdown
|id|description|
|utterance ids|the **{goal}** of the category|
"'

###Quality
- **No** **overlap** or **inclusion** among the categories.
- **Do not include vague categories** such as "Other","General","Unclear","Miscellaneous" or "Undefined" in the cluster table.
- Provide your answers between the tags: <cluster table>your generated cluster table</cluster table>, <explanation>explanation of your reasoning process within **{n_exp}** words</explanation>.
- If the data points convey the **same** **{goal}**, you should output just one category.
- **Description** can **accurately** and **consistently** classify the Data **without ambiguity**. A data point must have and only belong to one category.

# Data
{data_table}

# Output

Table 13: A prompt template for clustering a batch of texts and generating mini-cluster labels, where the Chain-of-Thought (COT) method is used to improve the accuracy of the results. In practice, users need to modify the components highlighted in **Bold** (e.g., changing "according" to "according to the intent" and "goal" to "intent") to suit their specific dataset.

Prompt template P_G for generating labels.

```
Instruction
##Context
- *Goal* Your goal is to summarize the input data into a meaningful LABEL **{according}**.
- *Data* The input data will be a markdown table containing category descriptions and the corresponding number of utterances for each category, with the following columns:
- **category description** A description of the category, generated by clustering related utterances.
- **number** The number of utterances that belong to this category.

##Requirements
- Provide your answers between the tags: <summary>your generated summary LABEL with less than 8 words</summary>, <explanation>explanation of your reasoning process within {n_exp} words</explanation>.

# Data
{data_table}

# Output
```

Table 14: A prompt template for generating a new summary label for mini-cluster labels, where the Chain-of-Thought (COT) method is used to improve the accuracy of the results. In practice, users need to modify the components highlighted in **Bold** (e.g., changing "according" to "according to the intent" and "goal" to "intent") to suit their specific dataset.