TWIST: Training-free and Label-free Short Text Clustering through Iterative Vector Updating with LLMs

Anonymous ACL submission

Abstract

In this paper, we propose a training-free and label-free method for short text clustering that can be used on top of any existing embedder. In the context of customer-facing chatbots, companies are dealing with large amounts of user utterances that need to be clustered according to their intent. In these commercial settings, no labeled data is typically available, and the number of clusters is not known. Our method is based on iterative vector updating: it constructs sparse vectors based on representative texts, and then iteratively refines them through LLM guidance. Our method achieves comparable or superior results to state-of-the-art methods that use contrastive learning, but without assuming prior knowledge of clusters or labels. Experiments on diverse datasets and smaller LLMs show that our method is model agnostic and can be applied to any embedder, with relatively small LLMs, and different clustering methods. We also show that our method scales to large datasets, reducing the computational cost of the LLM. These low-resource, adaptable settings and the scalability of our method make it more aligned with real-world scenarios than existing clustering methods. Our source code is available here: `https://anonymous.4open.science/r/sparse_vector-F4C9/README.md`

1 Introduction

Short text clustering is an NLP task that automatically groups unlabeled data into clusters and is widely used in practical applications for text classification and new category detection. For example, in conversational understanding, user intents in real-world scenarios are dynamic and continuously evolving (Liang et al., 2024c), and relying on expert labeling can be costly. The short-text clustering task enables companies to efficiently organize tens of thousands of client requests in real time. Other downstream tasks, such as mining topics and opinions from online platforms, also rely on effective clustering (Wu et al., 2024).

The clustering task is often approached as a **Generalized Category Discovery (GCD) scenario** (Vaze et al., 2022), where limited labeled data and substantial unlabeled data are used to cluster all unlabeled samples, which include both known and unknown classes. Recently, other studies have explored an **emerging data scenario**, where no labels are provided at all (De Raedt et al., 2023; Viswanathan et al., 2024b; Zhang et al., 2023; Lin et al., 2025). This scenario reduces reliance on human annotation and shows greater flexibility.

A common approach for the emerging data scenarios is to train an embedder with contrastive learning to improve text representations (Zhang et al., 2021). With the advancement of LLMs, training-free approaches have also been proposed (De Raedt et al., 2023). The emerging data scenario has attracted increasing attention in recent research (Zhang et al., 2023; Feng et al., 2024; Lin et al., 2025). Despite the progress of prior work, these label-free approaches still face certain limitations and leave room for further improvement. First, methods involving LLM feedback often assume **known number of clusters** during method development (Zhang et al., 2023; Feng et al., 2024), which is difficult to estimate in real-world settings. Second, training-free methods have mainly been explored with large, proprietary models (Viswanathan et al., 2024b; De Raedt et al., 2023), which can be costly. Third, the **scalability** of these methods remains a concern. Motivated by these challenges, we propose a training-free approach that requires only a smaller LLM, alleviates the known-cluster assumption, and is scalable. The core idea of our method is to construct new representations (referred to as sparse vectors in this paper) such that texts belonging to the same cluster become more similar. Our method consists of two stages: we first construct sparse representations based on representative texts, and then iteratively refine them

by combining LLM guidance with pre-trained embeddings.

The contributions of our method are as follows:

- **Inclusive and easy to optimize:** Builds on existing embedders and requires only some hyperparameter tuning.
- **Flexible:** Does not assume a known number of clusters, nor require labeled data, contrastive learning on the task specific datasets, or fine-tuning, enabling easier adaptation to new domains.
- **Low-resource:** Even with small LLMs and embedders, the results are already competitive with methods that rely on labeled data or fine-tuning.
- **Scalable:** Does not require LLM feedback for every text, unlike many previous studies.

2 Related Work

Short text clustering Generalized Category Discovery (GCD) is often studied for this task (Liang et al., 2024b; Liang and Liao, 2023; Zou et al., 2025b). These works use limited labeled text and substantial unlabeled text to learn clustering representations. However, these studies use train–test splits within a single dataset, which implicitly assumes that new categories are similar to the known ones. In practice, emerging data may show topic drift. Recently, with the advancement of LLMs, many studies have explored the emerging data scenario (Zhang et al., 2023; Viswanathan et al., 2024b; Lin et al., 2025), where no labels are available. We follow this unsupervised scenario.

Boosting Unsupervised Short Text Clustering with LLM-Guided Contrastive Learning Contrastive learning is commonly used in unsupervised short text clustering (Zhang et al., 2021, 2023; Liang et al., 2024a). Zhang et al. (2023) use LLMs to construct hard triplets to derive training pairs, which are then used to fine-tune a small embedder. Liang et al. (2024a) use LLMs to refine texts by generating descriptors, then using small language models for contrastive learning to align the embeddings of texts and their descriptors. However, these methods either rely on knowing the number of clusters, proprietary models, or fine-tuning. Our method does not have these requirements.

Training-Free Methods for Unsupervised Short Text Clustering Training-free approaches use a frozen pre-trained embedder: De Raedt et al. (2023) select prototypical utterances and then use an LLM to generate a set of descriptive utterance labels for these prototypes. The labels and texts are then encoded to form better representations. Viswanathan et al. (2024a) use LLMs to improve texts by generating key phrases for each text, which are then separately encoded into embeddings and concatenated for clustering. Lin et al. (2025) use LLMs for selection rather than text enrichment. They first encode all of the texts, and then use the LLM to select similar pairs for each text. Each text embedding is then obtained by average pooling the embeddings of the text and its similar pairs. These methods have in common that they refine original embeddings to improve clustering. In contrast, we construct a new vector using LLM feedback. Feng et al. (2024) use LLMs to refine edge points. However, the assumption of a known number of clusters can still be a core problem in real-world applications.

3 Theoretic formulation and assumptions

3.1 Problem definition

In our work, we consider two variations of the short text clustering task: (i) the emerging data scenario and (ii) the GCD scenario. In the emerging data scenario (Lin et al., 2025), the only available resource is a collection of unlabeled texts $D^{test} \triangleq \{x_i\}_{i=1}^{N}$, where each $x_i \in D^{test}$ corresponds to a short text and N is the total number of samples. The dataset is used as a whole for clustering and evaluation, without any train-test split. In contrast, the GCD scenario has access to additional data, typically including some labeled samples (Zou et al., 2025b). In this work, we adapt this setting by considering only an additional unlabeled dataset $D^u \triangleq \{x_i\}_{i=1}^{N_u}$, in line with the goal of low-resource. The GCD scenario is included for the purpose of testing whether extra data improves clustering and generalizability of our method.

For both scenarios, the task is to partition D^{test} into clusters $\{\hat{S}_l\}_{l=1}^{\hat{K}}$, where $\hat{K}$ is the number of predicted clusters. Note that $\sum_{l=1}^{\hat{K}} |\hat{S}_l| = N$. The goal is to ensure that the clustering results $\{\hat{S}_l\}_{l=1}^{\hat{K}}$ are as close as possible to the true partition $\{S_l\}_{l=1}^{K}$, where K is the number of unique clusters in the ground truth. Importantly, unlike many existing studies (De Raedt et al., 2023; Feng et al., 2024; Zou et al., 2025b), which assume that K is known in advance, we make no such assumption.

For notation convenience, we define the full dataset as $D \triangleq D^{test} \cup D^u$ to simplify the explanation of our proposed method. Note that $D = D^{test}$ in the emerging data scenario.

3.2 Constructing the Sparse Vector

Basic Assumption Considering the fact that texts belonging to the same cluster are similar to each other, we represent each text as a sparse bag-of-texts vector that directly encodes its similarity to other texts. Formally, for any $x_i, x_j \in D$ with $i \neq j$:

- x_i and x_j are similar if $x_i, x_j \in S_k$ for some cluster S_k.
- x_i and x_j are not similar if $x_i \in S_k$ and $x_j \in S_l$ with $k \neq l$.

Construction of Bag-of-Texts Since similarity is determined by clusters, it is not necessary to use all texts in D to represent sparse vectors; instead, a representative subset of texts can constitute our bag of texts, denoted as $D^r \subseteq D$, where $|D^r| \triangleq d$. The set of representative texts D^r should contain at least one element from each cluster. That is, for any cluster $S \in \{S_l\}_{l=1}^K$, there exists at least one text $x \in D^r$ such that $x \in S$.

Therefore, to cover all possible clusters, the minimum size of d is $K - 1$. Since K is unknown, the most practical approach is to choose a relatively large d. For each $x \in D$, we define a d-diemnsional vector $\mathbf{z}$ assigned to the representative texts, where the j-th coordinate takes the value 1 if x is similar to the j-th representative text, and 0 otherwise.

Proposition 1. *Given the assumption, for any $x_i, x_j \in D$ with $i \neq j$ that belong to the same cluster S_k, the corresponding sparse vectors $\mathbf{z}_i$ and $\mathbf{z}_j$ will be identical.*

The proposition is straightforward, so we omit the proof.

Adaptation to real-world practice Although the proposition holds in the idealized setting, in practice similarity is typically determined using LLMs, which may introduce errors, and scanning the entire dataset for each x is computationally expensive. To address these, we construct a sparse vector $\mathbf{z}$ in a way that is adaptable to real-world practice: for each $x \in D$, instead of assigning fixed value in $\{0, 1\}$ based on representative texts, we propose an updating procedure that accounts for uncertainty. In addition, to address limited computational resources, we use an existing embedder to identify a relatively small set of m similar candidates that are likely to belong to the same category, as empirically shown in previous research (Lin et al., 2025; Zou et al., 2025a). The details of the method will be discussed in the next section.

4 Computational Method

Our method aims to construct sparse vectors that capture the similarity of texts within the same cluster. It consists of two stages. In the initial stage, we select representative texts, which are then used to construct a sparse vector for each text. In the second stage, we iteratively update each text's sparse vector by considering its similar texts until convergence, illustrated in Figure 1.

Unlike many prior studies that assume the number of clusters K is known, we consider a more practical setting in which K is unknown, and thus approximate the true number of clusters by the number of representative texts, d. This is less challenging than estimating K, as it only requires ensuring that d is large enough to cover K clusters.

4.1 Initial stage: construction of sparse representation

To build sparse vector representations, we first need to create a set of representative samples D^r. To this end, we first encode each text using an existing embedder F to obtain pre-trained embeddings:

$$\mathbf{x}_i = F(x_i).$$

Next, we apply a clustering algorithm to partition these embeddings into d clusters.

We use agglomerative clustering (Murtagh and Legendre, 2014) as it produces deterministic results, which facilitates reproducibility. For each initial cluster , we derive the medoid. We refer to the set of medoids as the representative text, denoted as:

$$D^r \triangleq \{x_{c_1}, \ldots, x_{c_d}\}.$$

and let $\mathbf{X}^r$ be D^r corresponding pre-trained embeddings, where $\mathbf{X}^r = \{\mathbf{x}_i : x_i \in D^r\}$. We then transform the representative texts into one-hot vectors, which serve as the basis vectors for the sparse representation.

For the remaining texts $x_j \in D \setminus D^r$, we construct their sparse vectors using this basis. For each x_j, we compute the pairwise cosine distances between its embedding $\mathbf{x}_j$ and the medoid embeddings in $\mathbf{X}^r$. By sorting these distances, we select

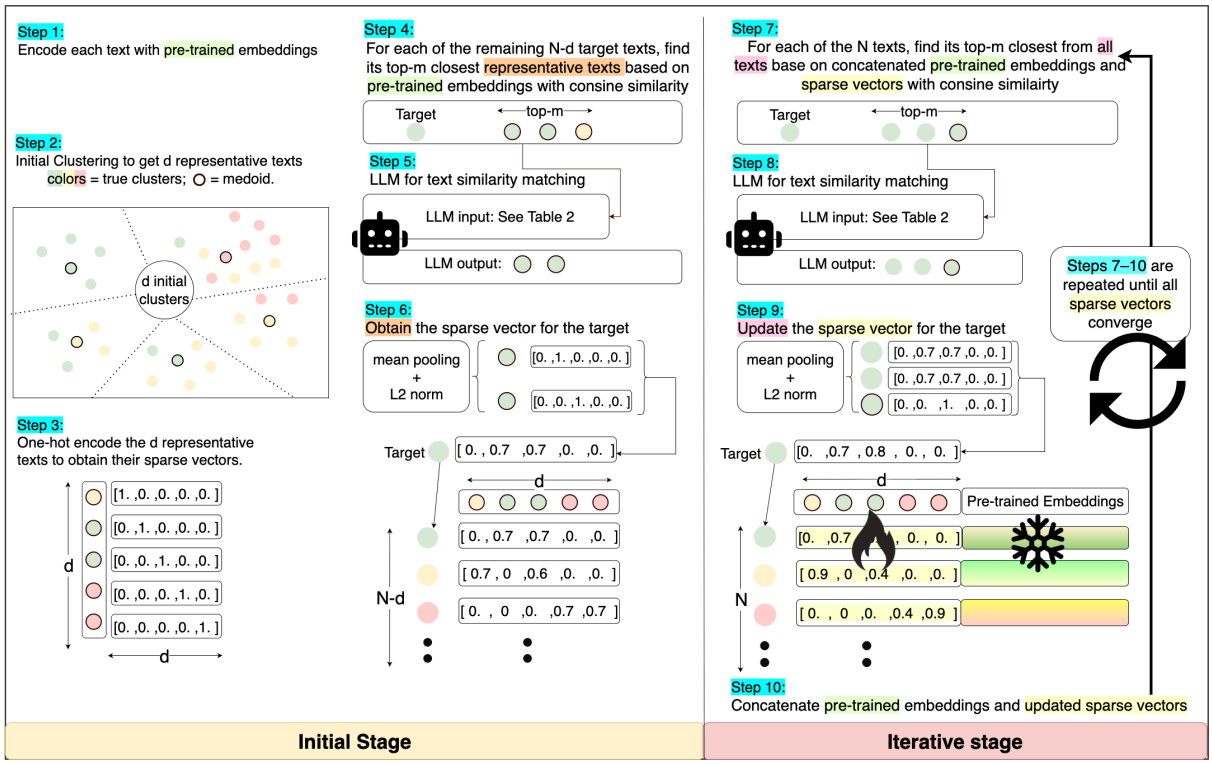

Figure 1: Construction of Sparse vectors. **Initial stage**: We first encode each text in D using a pretrained embedder and partition them into d clusters. We then select the medoid of each cluster as the representative text and construct one-hot vectors for these representatives. The remaining text vectors are obtained based on LLM selection. **Iterative stage**: We update all of the vectors iteratively until convergence.

the top m closest representative texts to x_j, denoted as $x_{j1}, \ldots, x_{jm}$. Then, we use an LLM to identify the similar text(s) to x_j among these representative texts, denoted as $M_j \triangleq LLM(x, x_{j1}, \ldots, x_{jm}) \in \mathcal{P}(\{x_{j1}, \ldots, x_{jm}\}) \cup \{\text{None}\}$, where $\mathcal{P}(\cdot)$ represents the power set, i.e., all possible subsets of the candidate texts, and 'None' indicates that no representative text is selected. If the LLM output is not None, the sparse vector of x_j is then derived by averaging the sparse vectors of the texts in M_j, denoted as $\mathbf{z}_j$. Instead of assigning a fixed value of 1 to the selected dimension, we average the vectors to capture how strongly each representative text contributes to the feature. We then apply L2 normalization to ensure a consistent scale for the iterative stage. If the LLM output is None, we make this text into a representative text itself and increase the dimensionality of all sparse vectors by one and construct its corresponding one-hot vectors.

Note: if we set $d = N$, all texts will be representative texts and one-hot encoded.

4.2 Iterative stage: sparse vectors refinements

In this stage, we update the sparse vectors of all texts using the dynamic subset of m candidates.

At each iteration step t (with the initial stage being $t = 0$), for each $x_j \in D$, we first concatenate its pre-trained embedding and sparse vector: $\mathbf{u}_j^t \triangleq \mathbf{x}_j \oplus \mathbf{z}_j^t$. We then compute the pairwise cosine distances between $\mathbf{u}_j^t$ and the concatenated representations of the all texts $\mathbf{U}^t$. By sorting these distances, we select the top m closest texts to x_j, denoted as $x_{j1}^t, \ldots, x_{jm}^t$. The concatenation ensures that the pre-trained embedding guides the sparse vectors. Also, the reason we do iterative updating instead of just once is to gain more stable sparse vectors.

Next, we use an LLM to identify the subset of the texts most similar to x_j, denoted as $M_j^t \triangleq LLM(x_j, x_{j1}^t, \ldots, x_{jm}^t) \in \mathcal{P}(\{x_{j1}^t, \ldots, x_{jm}^t\}) \cup \{\text{None}\}$. If the LLM output is None, we fix sparse vectors, i.e., $\mathbf{z}_j^{t+1} = \mathbf{z}_j^t$. Otherwise, we update the sparse vectors by averaging the sparse vector of the texts in M_j^t. We then apply L2 normalization, as explained previously.

For x_j, the update process is ended if the sparse vector at iteration $t + 1$ is almost identical to those at iteration t, as we define by a cosine similarity greater than 0.99. For efficiency, we set $m = 30$ and cap the maximum number of iterations at 10.

Emerging data	K	$\lvert D^{test}\rvert$	$\lvert D^{u}\rvert$
Bank77	77	3,080	-
Clinc150	150	4,500	-
Mtop	102	4,386	-
Massive	59	2,974	-
GoEmo	27	5,940	-
GCD	K	$\lvert D^{test}\rvert$	$\lvert D^{u}\rvert$
Bank77	77	3,080	2,700
Clinc150	150	2,250	5,400
Stackoverflow	20	1,000	5,400

Table 1: Dataset Statistics in two scenarios. **Emerging data scenario**: we use the same settings as Zhang et al. (2023); Lin et al. (2025). **GCD scenario**: for fair comparison, we evaluate on the same test sets as De Raedt et al. (2023); Liang et al. (2024a); Zou et al. (2025b), but we only use 30% of the original training set, which contains unlabeled data only. Further details are provided in the Appendix A.

CLINC150, Bank77, Mtop, and Massive
Task Instructions: Compare each Candidate Utterance to the Target Utterance. Select only Candidates with the same intent as the Target Utterance. Intent refers to the request or the purpose the user wants to achieve.
GoEmo
Task Instructions: Compare each Candidate Utterance to the Target Utterance. Select only Candidates with the same emotion as the Target Utterance.
Stackoverflow
Task Instructions: Compare each Candidate Question to the Target Question. Select only Candidates with the same main programming framework, language, tool, or concept as the Target Question.
Answer Format: Only provide the final selection of Candidate Utterances by listing their numbers if they match the Target Utterance intent or request. 1. If Candidates 3, 4, and 9 match, write: The Candidate utterance number(s): 3, 4, 9 2. If no Candidates match, write: The Candidate utterance number(s): none Note: Stick to the answer format and avoid providing extra explanations. **Task:** Target Utterance: {sentence 1} Candidate Utterances: 1. {sentence 1} ... m. {sentence m}

Table 2: Task Instructions Prompt. Note: The boldface used here is for readability; it is not used in the prompt. If the dataset is Stackoverflow, we use 'question' instead of 'utterance' in the Answer format.

5 Experimental Settings

5.1 Datasets and models

We use different topics of datasets for our experiments, including the intent datasets Bank77 (Casanueva et al., 2020), CLINC150 (Larson et al., 2019), Mtop (Li et al., 2020), Massive (Fitzgerald et al., 2023); the emotion dataset GoEmo (Demszky et al., 2020), and the community QA dataset Stackoverflow (Xu et al., 2015) for our experiments. Statistics are provided in Table 1.

For the **emerging data scenario**, we use the embedders e5-large (Wang et al., 2022) and instructor-large (Su et al., 2022) to compare with previous studies (Zhang et al., 2023; Feng et al., 2024). We also experiment with bert-base-uncased (Devlin et al., 2018), all-MiniLM-L6-v2 (Reimers and Gurevych, 2019). For the **GCD** scenario, we use all-MiniLM-L6-v2 (22.7M paramters) as the embedder, since most prior studies (An et al., 2024; Zou et al., 2025b) fine-tune bert-base (110M paramters). Using all-MiniLM-L6-v2 allows us to minimize the impact of model size on our experiments.

For the LLM experiments, the main results are obtained using gemma-2-9b-it (Rivière et al., 2024). Llama3.1-8B-Instruct (AI@Meta, 2024) and Qwen/Qwen3-8B (Team, 2025) are used for hyperparameter search or further analysis. An A100 GPU or an H100 GPU were used throughout the experiments.

5.2 Prompts used

Prompt for LLM The prompts we used are shown in Table 2. It consists of three parts: task instruction, answer format, and task.

5.3 Hyperparameter setting

The main hyperparameter in our approach is the initial dimension (number of representatives) of the sparse vector d, which we set to 2,048 based on validation results on two external datasets: ArxivS2S and Reddit (Muennighoff et al., 2023). We use external datasets to prevent overfitting on a specific dataset. See details in Appendix B. Qwen3-8B (as LLM) and All-MiniLM-L6-v2 (as embedder) are used in the hyperparameter search.

5.4 Evaluation

The emerging and GCD scenarios are evaluated under two conditions: with HDBSCAN (McInnes et al., 2017) and K-means (Lloyd, 1982).

To reflect real-world settings with unknown K, we apply HDBSCAN (McInnes et al., 2017) to partition D^{test} into $\hat{K}$ clusters. We set `min_samples = 1` to reduce noise. The noise points are assigned to the nearest cluster centroid. We perform an incremental grid search over `min_cluster_size` values of $10, 15, \ldots, 50$, stopping early if the silhouette score does not improve (Rousseeuw, 1987; Gung et al., 2023).

K-means clustering is the dominant evaluation practice because many studies assume a known

cluster number K and incorporate it in their method pipelines (Zhang et al., 2023; De Raedt et al., 2023; Liang et al., 2024b; Feng et al., 2024; Zou et al., 2025b). Following their convention, the number of clusters for K-means is set to match the number of ground-truth cluster number K.

To reduce cost, we follow (Zhang et al., 2023) and perform a single pipeline run per dataset using a fixed seed to obtain a sparse vector for each text. Most steps are deterministic. The main source of variability comes from splitting candidate examples across multiple LLM calls (e.g., three calls with 10 examples instead of one with 30), done only to save time and GPU memory. LLM outputs are generated via greedy decoding, without sampling. For evaluation, K-means is run over 5 different seeds and averaged. HDBSCAN, which is near-deterministic, is run once. After we derive the clusters, we apply standard clustering metrics for evaluation. These metrics include normalized mutual information (NMI) and clustering accuracy (Acc) (Rand, 1971; Meilă, 2007; Huang et al., 2014; Gung et al., 2023).

5.5 Baselines

Emerging Data Scenario We compare against **SCCL** (Zhang et al., 2021), ClusterLLM, (Zhang et al., 2023), IDAS (De Raedt et al., 2023), LLMedgeRefine (Feng et al., 2024), and SPILL (Lin et al., 2025). Table 3 shows that our method makes the fewest assumptions and has the lowest requirements. SPILL is the only method under the same assumptions. However, we still include ClusterLLM, IDAS, and LLMedgeRefine, which uses additional information – the known K – in its development pipeline, to provide a more comprehensive comparison.

GCD Scenario We compare against IDAS (De Raedt et al., 2023), Loop (An et al., 2024), ALUP (Liang et al., 2024b), IntentGPT (Rodriguez et al., 2024) and Glean (Zou et al., 2025b). Note that IntentGPT and IDAS can also be implemented only using unlabeled data. Table 3 shows that our method again makes the fewest assumptions and has the lowest requirements.

6 Results

6.1 Main results

Emerging data scenario Table 4 shows the HDBSCAN results. Our approach substantially improves performance compared to directly using the embedder.[1] Moreover, the results from BERT (non-contrastive) also improve with our approach. For the K-means results, Table 5 shows that our approach generally outperforms IDAS and ClusterLLM on both metrics across different embedders, and achieves comparable results to LLMEdgeRefine. This indicates that, even without assuming prior knowledge of K during method development – as the compared methods do – our approach remains effective and better reflects real-world practice. While SPILL achieves slightly higher NMI, our method attains substantially higher ACC.

Lastly, we found that the HDBSCAN results overall are similar to those of K-means, indicating that our method remains robust even without knowing K.

Emerging data	FT	known K	Labeled data
TWIST (Ours)			-
SCCL (Zhang et al., 2021)	✓		-
ClusterLLM (Zhang et al., 2023)	✓	✓	-
LLMEdgeRefine (Feng et al., 2024)		✓	-
IDAS (De Raedt et al., 2023)		✓	-
SPILL (Lin et al., 2025)			-
GCD			
TWIST (Ours)			
Loop (An et al., 2024)	✓	✓	✓
ALUP (Liang et al., 2024b)	✓	✓	✓
IntentGPT (Rodriguez et al., 2024)			
Glean (Zou et al., 2025b)	✓	✓	✓

Table 3: Baselines settings. **FT**: Finetuning. **known** K: Assuming known K during the method pipeline development. **Labeled data**: Labeled data is required.

GCD scenario Table 6 shows the HDBSCAN results for GCD. Our method consistently improves the performance of the existing embedder. Table 7 shows the K-means results for GCD. Our approach is on average the best in the unlabeled setting, without any known categories. It even outperforms baselines that know 10% of the categories, and is comparable to other baselines that know 25% of the categories (or up to 75% in the case of IntentGPT). These strong results, together with the fewer assumptions required by our method, highlight its effectiveness and efficiency. Notably, BANK77 and CLINC150 perform similarly with both clustering algorithms, while Stackoveflow shows a noticeable drop.

[1]The estimated number of clusters, $\hat{K}$, is provided in Appendix D.

	Bank 77		Clinc150		Mtop		Massive		GoEmo		Average	
	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc
Bert												
Plain	0.84	1.69	73.03	50.87	63.95	40.97	2.86	8.18	1.35	12.05	28.41	22.75
TWIST (Ours)	**76.33**	**60.23**	**93.23**	**85.91**	**73.86**	**47.47**	**72.02**	**62.32**	**21.01**	**14.19**	**67.29**	**54.02**
MiniLM												
Plain	78.78	60.39	86.79	67.71	67.30	**55.91**	62.55	48.15	11.31	18.62	61.35	50.16
TWIST (Ours)	**85.61**	**75.29**	**93.86**	**86.44**	**74.52**	50.36	**76.67**	**67.51**	**21.63**	**23.27**	**70.46**	**60.57**
E5												
Plain	78.68	62.18	87.57	65.62	63.45	**53.79**	1.15	7.74	15.43	17.70	49.26	41.41
TWIST (Ours)	**84.86**	**72.24**	**93.41**	**86.89**	**72.53**	44.32	**75.85**	**65.12**	**27.77**	**29.03**	**70.88**	**59.52**
Instructor												
Plain	82.50	67.34	92.29	79.27	67.63	**54.31**	67.99	53.54	0.25	11.48	62.13	53.19
TWIST (Ours)	**85.63**	**73.77**	**93.64**	**84.62**	**74.06**	46.81	**76.52**	**64.07**	**31.64**	**33.65**	**72.30**	**60.58**

Table 4: Five-benchmark results (**Emerging Data Scenario, HDBSCAN**). Gemma is used in TWIST. **Plain**: direct embedding clustering. Bold numbers: best within the embedder.

	Bank 77		Clinc150		Mtop		Massive		GoEmo		Average	
	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc
Bert												
Plain	50.72 (0.42)	29.07 (0.85)	72.90 (0.60)	49.48 (0.83)	62.24 (0.46)	26.78 (0.32)	49.12 (0.64)	30.49 (1.00)	10.77 (0.46)	12.90 (0.37)	49.15	29.75
TWIST (Ours)	**76.23** (0.14)	**60.49** (0.72)	**93.25** (0.07)	**86.43** (0.65)	**71.79** (0.20)	**36.41** (0.20)	**72.00** (0.13)	**60.56** (0.61)	**18.66** (0.10)	**18.76** (0.29)	**66.38**	**52.53**
MiniLM												
Plain	79.08 (0.80)	61.27 (1.36)	89.46 (0.19)	73.73 (0.95)	67.41 (0.53)	31.40 (1.62)	70.37 (0.62)	54.04 (1.28)	10.84 (0.16)	13.89 (0.63)	63.43	46.87
TWIST (Ours)	**85.57** (0.06)	**74.07** (0.71)	**93.65** (0.07)	**86.35** (0.34)	**71.92** (0.22)	**37.83** (0.72)	**76.18** (0.26)	**66.15** (0.58)	**20.69** (0.09)	**24.44** (0.37)	**69.60**	**57.77**
E5												
Plain	77.86 (0.35)	60.88 (0.89)	91.09 (0.17)	75.70 (0.36)	71.17 (0.28)	33.04 (0.74)	71.70 (0.84)	53.97 (2.00)	21.13 (0.44)	21.93 (0.60)	66.59	49.10
SCCL	77.34 (0.62)	63.60 (1.37)	91.89 (0.49)	77.96 (1.78)	70.42 ((0.34)	33.82 ((1.07)	71.57 (0.89)	54.48 (1.80)	20.05 (0.63)	22.03 (0.69)	66.25	50.38
ClusterLLM	84.16 (0.36)	70.13 (1.34)	92.92 (0.29)	80.48 (0.93)	**74.46** (0.11)	37.22 (1.18)	74.39 (0.21)	56.08 (1.01)	22.23 (0.17)	22.22 (1.15)	69.63	53.23
SPILL	83.56 (0.47)	70.25 (1.56)	92.93 (0.08)	83.18 (0.78)	71.77 (0.35)	36.83 (1.10)	75.40 (0.51)	60.28 (1.81)	25.14 (0.36)	24.82 (0.86)	69.76	55.07
TWIST (Ours)	**85.09** (0.19)	**74.23** (0.88)	**93.31** (0.04)	**85.30** (0.58)	71.57 (0.18)	**37.32** (1.67)	**75.72** (0.22)	**62.93** (1.28)	**26.58** (0.23)	**33.20** (0.54)	**70.45**	**58.60**
Instructor												
Plain	82.42 (0.68)	65.49 (2.41)	93.27 (0.32)	81.68 (1.31)	71.63 (0.39)	33.14 (0.52)	74.02 (0.54)	54.62 (2.06)	29.18 (0.76)	32.51 (1.51)	70.10	53.49
SCCL	81.77 (1.36)	65.48 (1.36)	92.94 (0.44)	80.85 (0.74)	73.52 (0.38)	34.28 (0.58)	73.90 ((0.36)	54.10 ((1.05)	30.54 (0.68)	34.33 (0.86)	70.53	53.81
ClusterLLM	85.15 (0.41)	71.20 (1.59)	94.00 (0.21)	83.80 (0.41)	**73.83** (0.79)	35.04 (0.97)	**77.64** (0.21)	60.69 (0.96)	23.89 (0.68)	26.75 (1.76)	70.90	55.50
LLMEdgeRefine	-	-	**94.86**	**86.77**	72.92	**46.00**	76.66	**63.42**	29.74	34.76	-	-
IDAS	-	-	92.35	81.36	72.31	37.30	75.74	63.01	25.57	30.61	-	-
SPILL	85.01 (0.29)	71.05 (0.83)	93.77 (0.35)	85.14 (1.04)	72.65 (0.32)	37.11 (1.62)	77.62 (0.46)	62.42 (2.06)	**32.78** (0.52)	33.21 (2.33)	**72.37**	57.79
TWIST (Ours)	**85.46** (0.18)	**74.66** (1.01)	93.92 (0.09)	85.62 (0.71)	72.34 (0.28)	38.40 (0.81)	76.43 (0.03)	63.17 (0.25)	30.62 (0.24)	**37.04** (1.55)	71.75	**59.78**

Table 5: Five-benchmark results (**Emerging Data Scenario, K-means**). Averages over 5 runs (±SD). Gemma is used in TWIST. **Plain**: direct embedding clustering. Bold numbers: best within the embedder. SCCL/ClusterLLM/LLMEdgeRefine cited directly from prior work. SCCL results are cited from ClusterLLM; IDAS results are cited from LLMedgeRefine.

		Bank 77		Clinc150		StackO		Average	
KCR	Method	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc
-	Plain	78.78	60.39	83.02	51.82	66.26	57.20	76.02	56.47
0%	TWIST	**87.43**	**75.16**	**95.19**	**89.07**	**78.54**	**71.70**	**87.05**	**78.64**

Table 6: Three-benchmark results (**GCD scenario, HDBSCAN**) Gemma is used in TWIST. Bold numbers: best. KCR ratio of known category. All-MiniLM-L6-v2 used as backbone embedder. **Plain**: direct embedding clustering. StackO: StackOverflow dataset.

6.2 Further analysis

Effect of Different LLMs Our main results were obtained with gemma-2-9b-it. To investigate whether using a different LLM affects performance, Figure 2 shows the results obtained with all-MiniLM-L6-v2 as the embedder and three different LLMs (see Appendix E for details). Overall, the differences between Qwen3-8B and gemma-2-9b-it are similar but an obvious drop with Llama3.1-8B-Instruct. Because Qwen3-8B achieves comparable results, we use it along with all-MiniLM-L6-v2 for remaining analyses to reduce computational cost.

Effect of the Number of Closest Neighbors m and Iterative Updates We use $m = 30$ across all the experiment, but this raises the question of whether different values of m would affect performance. Figure 3a shows that using too few neighbors leads to lower results. To investigate the effect of iterative updates, Figure 3b shows that in subsequent iterations, the results substantially improve, especially after the first iteration. Since most sparse vectors converge within 5 iterations, there are only minimal changes afterward: We find that most datasets have 90% convergence ratio by the 5th iteration (see Appendix F).

A scalable implementation of our method Our representation relies on LLM guidance for each text, which can be computationally expensive when the dataset is extremely large. To assess scalabil-

KCR	Method	Bank 77		Clinc150		StackO		Average	
		NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc
-	Plain	79.08	61.27	89.46	73.73	72.49	72.68	80.34	69.08
0%									
	IDAS	82.84	67.43	93.82	85.48	81.26	83.82	85.97	78.91
	IntentGPT	81.42	64.22	94.35	83.20	-	-	-	-
	TWIST (Ours)	**87.67**	77.10	95.21	89.07	**81.69**	82.56	**88.19**	82.91
10%									
	Loop	79.14	64.97	93.52	84.89	75.98	80.50	82.88	76.79
	GLEAN	82.23	67.99	95.21	88.71	79.67	82.40	85.70	79.70
25%									
	ALUP	84.06	74.61	94.84	88.40	76.58	82.20	85.16	81.74
	Loop	83.37	71.40	94.38	86.58	79.10	82.20	85.62	80.06
	GLEAN	85.62	76.98	**96.27**	**91.51**	80.90	**84.10**	87.60	**84.20**
75%	IntentGPT	85.94	**77.21**	96.06	88.76	-	-	-	-

Table 7: Three-benchmark results (**GCD scenario, K-means**). Averages over 5 runs with Gemma. KCR: known categories ratio. All-MiniLM-L6-v2 used as backbone embedder in TWIST. Loop results are cited from GLEAN. Plain: direct embedding clustering.

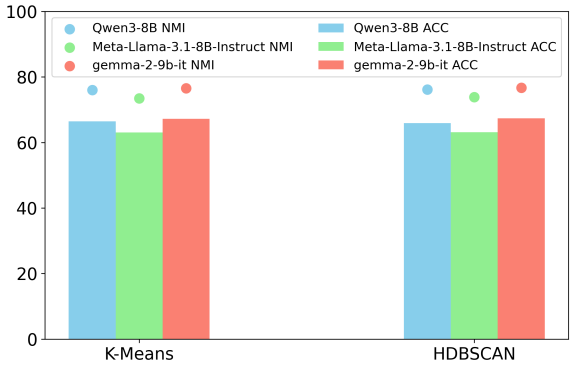

Figure 2: ACC and NMI across all test datasets (Emerging Data and GCD) for three LLMs. Results for K-means are averaged over 5 runs, HDBSCAN from a single run.

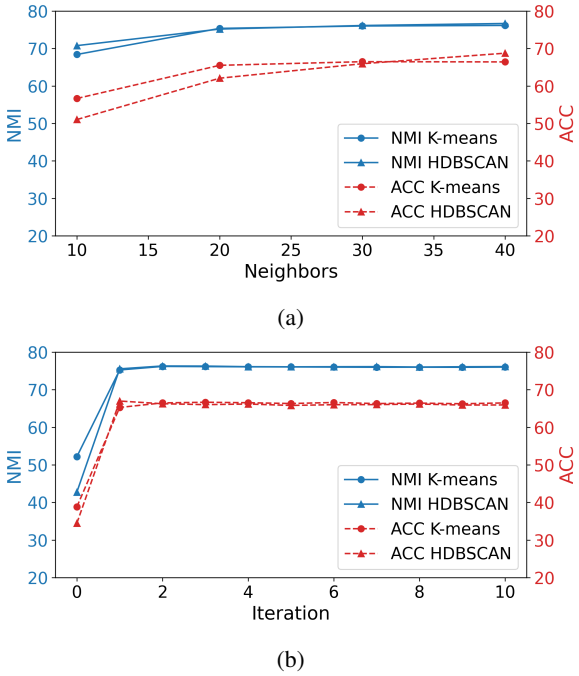

Figure 3: Comparison of (a) varying negihbor m and (b) iterative updates on ACC and NMI across all test datasets (Emerging Data and GCD).

Method	Bank 77		Clinc150		StackOverf		Average	
	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc
Kmeans								
Plain	76.31	59.41	88.99	73.99	68.42	65.93	77.91	66.44
TWIST (Ours)	**83.80**	**73.58**	**92.78**	**87.27**	**72.98**	**76.90**	**83.19**	**79.25**
HDBSCAN								
Plain	76.15	60.27	84.31	62.84	63.01	61.80	74.49	61.64
TWIST (Ours)	**83.39**	**71.79**	**92.55**	**86.04**	**71.08**	**69.58**	**82.34**	**75.80**

Table 8: Clustering results on the entire dataset using knowledge distillation. Averages over 5 runs for K-means. Plain: Directly embedding the test dataset with All-MiniLM-L6-v2.

ity, we additionally explore clustering over the full dataset. The goal is to implement our approach on a subset of the texts and then use a model to generalize to the remaining texts (a form of knowledge distillation), without requiring further LLM computation. Specifically, we train a lightweight MLP (∼10M parameters) to map pre-trained embeddings to sparse vectors. See Appendix G for experiment details. Note that we do not use any labeled data. Table 8 shows that our approach still improves the original embeddings by a large margin, which proves the scalability of our method, offering substantial savings in time and computational resources.

7 Conclusions

We propose TWIST (Short Text Clustering through Iterative Sparse Vector Updating with LLMs), an intuitive, domain-adaptive, label-free, and training-free method for short text clustering. Compared with other SOTA methods, TWIST is built under minimal assumptions and requirements. TWIST achieves better or similar results to state-of-the-art approaches without assuming prior knowledge of clusters or labels, making it more aligned with real-world scenarios. Experiments on diverse datasets and smaller LLMs show that our method is robust in low-resource settings, and also scalable to large datasets. In the future, we plan to expand our approach to clustering in specialized domains and explore ways to incorporate human feedback into our method pipeline.

Limitations

Language. Like most of the prior work, we only focus on English utterance datasets. This is relevant because most of benchmark datasets are in English. Our method is completely data-driven and not dependent on labeled data and it is therefore applicable to other languages than English.

LLM capability. Our method relies on the abilities of LLMs. Although we empirically show that small LLMs can achieve good results, this is not guaranteed when the dataset comes from a rare domain.

Embedder capability. Although we empirically show that our approach generally improves performance compared to directly using the embedder for clustering, if the embeddings provide a poor initial result, the improvement might be limited.

References

AI@Meta. 2024. Llama 3 model card.

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

Iñigo Casanueva, Tadas Temcinas, Daniela Gerz, Matthew Henderson, and Ivan Vulic. 2020. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on NLP for ConvAI - ACL 2020*. Data available at https://github.com/PolyAI-LDN/task-specific-datasets.

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. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

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. 2023. Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4277–4302.

James Gung, Raphael Shu, Emily Moeng, Wesley Rose, Salvatore Romeo, Arshit Gupta, Yassine Benajiba, Saab Mansour, and Yi Zhang. 2023. Intent induction from conversations for task-oriented dialogue track at dstc 11. In *Proceedings of The Eleventh Dialog System Technology Challenge*, pages 242–259.

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

Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, et al. 2019. An evaluation dataset for intent classification and out-of-scope prediction. 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 1311–1316.

Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. 2020. Mtop: A comprehensive multilingual task-oriented semantic parsing benchmark. *arXiv preprint arXiv:2008.09335*.

Jinggui Liang and Lizi Liao. 2023. Clusterprompt: Cluster semantic enhanced prompt learning for new intent discovery. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10468–10481.

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.

Jinggui Liang, Yuxia Wu, Yuan Fang, Hao Fei, and Lizi Liao. 2024c. A survey of ontology expansion for conversational understanding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18111–18127.

I-Fan Lin, Faegheh Hasibi, and Suzan Verberne. 2025. SPILL: Domain-adaptive intent clustering based on selection and pooling with large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 15723–15737, Vienna, Austria. Association for Computational Linguistics.

S. Lloyd. 1982. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137.

Leland McInnes, John Healy, Steve Astels, et al. 2017. HDBScan: Hierarchical density based clustering.

Marina Meilă. 2007. Comparing clusterings—an information based distance. *Journal of multivariate analysis*, 98(5):873–895.

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.

Fionn Murtagh and Pierre Legendre. 2014. Ward's hierarchical agglomerative clustering method: which algorithms implement ward's criterion? *Journal of classification*, 31(3):274–295.

William M Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850.

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*. Association for Computational Linguistics.

Morgane Rivière, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, et al. 2024. Gemma 2: Improving open language models at a practical size. *CoRR*.

Juan A Rodriguez, Nicholas Botzer, David Vazquez, Christopher Pal, Marco Pedersoli, and Issam H Laradji. 2024. Intentgpt: Few-shot intent discovery with large language models. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

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

Qwen Team. 2025. Qwen3 technical report.

Sagar Vaze, Kai Han, Andrea Vedaldi, and Andrew Zisserman. 2022. Generalized category discovery. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7492–7501.

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

Vijay Viswanathan, Kiril Gashteovski, Carolin Lawrence, Tongshuang Wu, and Graham Neubig. 2024b. Large language models enable few-shot clustering. *Transactions of the Association for Computational Linguistics*, 11:321–333.

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*.

Xiaobao Wu, Thong Nguyen, and Anh Tuan Luu. 2024. A survey on neural topic models: methods, applications, and challenges. *Artificial Intelligence Review*, 57(2):18.

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

Dejiao Zhang, Feng Nan, Xiaokai Wei, Shang-Wen Li, Henghui Zhu, Kathleen McKeown, Ramesh Nallapati, Andrew O Arnold, and Bing Xiang. 2021. 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.

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.

Henry Peng Zou, Zhengyao Gu, Yue Zhou, Yankai Chen, Weizhi Zhang, Liancheng Fang, Yibo Wang, Yangning Li, Kay Liu, and Philip S Yu. 2025a. Testnuc: Enhancing test-time computing approaches and scaling through neighboring unlabeled data consistency. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30750–30762.

Henry Peng Zou, Siffi Singh, Yi Nian, Jianfeng He, Jason Cai, Saab Mansour, and Hang Su. 2025b. Glean: Generalized category discovery with diverse and quality-enhanced llm feedback. *CoRR*.

A Comparison of Data Splits in GCD scenario: Our Method vs. Prior Work

Table 9 shows the train–validation–test splits used in Loop (An et al., 2024), ALUP (Liang et al., 2024b) and GLEAN (Zou et al., 2025b) for the three benchmark datasets. In their setting, a specified ratio of all categories is randomly selected as known categories, referred to as the known category ratio (KCR). For each known category, 10% of the data is chosen to form the labeled dataset[2], while the remaining samples constitute the unlabeled dataset. Together, these two subsets form the training dataset. In our experiments, we do not use the validation set and use only 30% of the training dataset, restricted to the unlabeled setting with no access to labeled data. Evaluation is conducted on the same test split.

They train an embedder using the training split and optimizes it based on the labeled validation set, with the test set used solely for evaluation. In contrast, our method performs test-time embedding construction, so it requires to access test data, similar to IDAS De Raedt et al. (2023) and IntentGPT Rodriguez et al. (2024), using all available text data ($D = D^{test} \cup D^{u}$) to construct embeddings.

Since our clustering is unsupervised, no label information is used. In practice, clustering methods require access to the text to be clustered in order to construct representations, and our approach naturally uses all available texts.

	# clusters	# train	# valid	# test	# total
Bank77	77	9,003	1,000	3,080	13,083
Clinc150	150	18,000	2,250	2,250	22,500
StackO	20	18,000	1,000	1,000	20,500

Table 9: Train–Validation–Test splits from previous studies (An et al., 2024; Liang et al., 2024b; Zou et al., 2025b). 'StackO' is the StackOverflow dataset

B Selection of d

Table 10 shows the two datasets that we used for selecting the dimensionality of the sparse vector d. Figure 4 shows that 1024, 2048, and $\min\{4096, N\}$ give similar performance for both K-means and HDBSCAN. We selected value $d = 2048$ and used this value across all datasets and experiments.

[2]**IntentGPT** (Rodriguez et al., 2024) mention they use fewer than 10% for each known category, but they do not specify the exact percentage.

Emerging data	K	$\lvert D^{test} \rvert$
ArxivS2S	93	3,674
Reddit	50	3,217

Table 10: External datasets used for selecting the dimensionality hyperparameter d

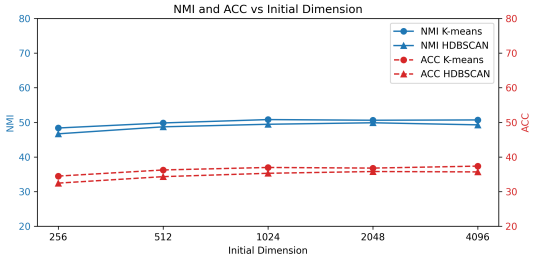

Figure 4: The average ACC and NMI across two datasets for different d. Results for K-means are averaged over 5 runs, while HDBSCAN is shown from a single run. 4096 here means $\min\{4096, N\}$ because the dataset size can be smaller.

C Prompts for Instructor embedder

Our method is built on top of existing embedders. For **instructor-large**, a prompt can be added to generate task-specific embeddings. We follow previous research (Su et al., 2022; Liang and Liao, 2023; Lin et al., 2025) and use mostly the same prompts. The prompts we used for the different datasets are listed in Table 11.

D Estimated Number of Clusters $\hat{K}$ by HDBSCAN

Table 12 and Table 13 show the estimated number of clusters, $\hat{K}$, for the two scenarios. The results indicate that the refined embeddings produced by our method are generally closer to the ground truth than the pre-trained (Plain) embeddings. An exception is the Bank77 dataset.

E Results by Different LLMs

We report the detailed results of all datasets by different LLMs in Table 14. Qwen and Gemma consistently outperform LLaMA. In addition, the average differences between K-means and HDBSCAN are minimal across both metrics for each LLM.

F Percentage of texts converged at iteration t

Figure 5 shows ratio of text convergence. It shows most datasets have 90% convergence ratio by the

Instructor:
Bank77: Represent the bank purpose for retrieval:
Mtop and Stackoverflow: Represent the sentence for retrieval:
Clinc150 and Massive: Represent the sentence for retrieving the purpose:
GoEmo: Represent an emotion sentence for retrieval:

Table 11: The prompts used in **Instructor**.

	Bank 77	Clinc150	Mtop	Massive	GoEmo
Bert					
Plain	2	111	62	2	2
TWIST	73	154	80	57	63
MiniLM					
Plain	**78**	139	17	27	11
TWIST	68	**150**	62	56	**44**
E5					
Plain	73	134	9	2	3
TWIST	70	142	82	57	58
Instructor					
Plain	71	144	18	29	2
TWIST	84	148	**99**	**58**	57
True K	77	150	102	59	27

Table 12: **Emerging** scenario: Estimated Number of Clusters $\hat{K}$ by HDBSCAN. **gemma-2-9b-it** used in TWIST.

5th iteration, and by the 7th iteration, all datasets achieve a 95% convergence ratio.

G Details of the full dataset clustering

Datasets Used with LLM Guidance Table 15 shows the number of examples we use with the LLM. The number of examples with LLM guidance is the same as in Table 1, i.e. $|D^{test}| + |D^{u}|$, so we do not need to re-run all experiments. The examples account for 29.6% to 44.2% of datasets. Examples with LLM guidance are used to train the MLP, since their sparse vectors have been derived. The pre-trained embeddings from all-MiniLM-L6-v2 serve as input, and the LLM-generated sparse vectors serve as target output for supervised learning with an MLP. 20% of the examples are used as a validation set to tune the hyperparameters.

MLP Architecture and Hyperparameter Search We build an MLP for each dataset. We only tune the hidden layer dimensionality using the validation set, with search over $\{512, 1024, 2048\}$.

Clustering of the entire dataset with MLP output We first encode all examples into pre-trained embeddings using all-MiniLM-L6-v2, then use the trained MLP to derive sparse versions of these embeddings. These sparse vectors are used to cluster all examples.

	Bank 77	Clinc150	Stackoverflow
Plain	**78**	82	**13**
TWIST	84	**142**	30
True K	77	150	20

Table 13: **GCD** scenario: Estimated Number of Clusters $\hat{K}$ by HDBSCAN. All-MiniLM-L6-v2 used as backbone embedder. **gemma-2-9b-it** was used in TWIST.

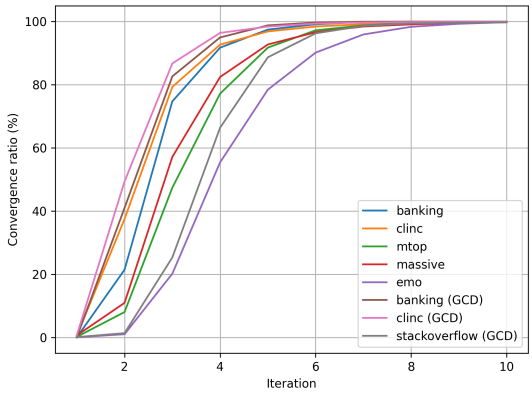

Figure 5: Ratio of texts converged at iteration t. All-MiniLM-L6-v2 (as embedder) and Qwen3-8B (as LLM) were used.

Dataset	K	Total	LLM	No-LLM	LLM ratio
Bank77	77	13,083	5,780	7,303	44.2%
Clinc150	150	22,500	7,650	14,850	29.6%
Stackoverflow	20	20,000	6,400	13,600	32.0%

Table 15: Dataset Statistics.

Evaluation For HDBSCAN, we set `min_samples = 1` to reduce noise. The noise points are assigned to the nearest cluster centroid. We perform an incremental grid search over `min_cluster_size` values of $50, 75, 100, 125, \ldots, 250$ – five times the original range – stopping early if the silhouette score does not improve. For K-means, we set the number of clusters equal to the ground-truth cluster count K.

	Bank 77		Clinc150		Mtop		Massive		GoEmo		Bank 77_{GCD}		$Clinc150_{GCD}$		$StackO_{GCD}$		Average	
	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc
Kmeans																		
Gemma	85.57	74.07	93.65	86.35	71.92	37.83	76.18	66.15	20.69	24.44	87.67	77.10	95.21	89.07	81.69	82.56	76.57	67.20
Qwen	85.13	73.18	93.59	86.96	72.23	39.03	74.55	63.64	20.12	22.80	87.04	75.78	94.80	88.21	80.69	82.32	76.02	66.49
Llama	82.46	69.81	91.03	82.21	70.08	36.21	72.47	60.47	17.60	19.67	84.34	70.91	93.83	86.27	75.69	78.68	73.44	63.03
HCBSCAN																		
Gemma	85.61	75.29	93.86	86.44	74.52	50.36	76.67	67.51	21.63	23.27	87.43	75.16	95.19	89.07	78.54	71.70	76.68	67.35
Qwen	85.04	73.90	93.56	86.42	74.42	50.11	74.39	62.36	22.10	19.81	86.81	75.29	94.78	88.18	78.07	71.50	76.15	65.95
Llama	82.45	68.70	91.00	78.64	72.80	51.12	73.19	63.84	18.53	16.84	84.16	69.35	93.84	84.18	74.78	72.20	73.84	63.11

Table 14: Further analysis of results using different LLMs.