LLMs Enable Bag-of-Texts Representations for Short-Text Clustering

I-Fan Lin
Leiden University
Leiden, The Netherlands
i.lin@liacs.leidenuniv.nl

Faegheh Hasibi
Radboud University
Nijmegen, The Netherlands
faegheh.hasibi@ru.nl

Suzan Verberne
Leiden University
Leiden, The Netherlands
s.verberne@liacs.leidenuniv.nl

Abstract

In this paper, we propose LeBoT (LLM-enabled Bag-of-Texts), a training-free representation refinement method for unsupervised short text clustering that relies less on careful selection of embedders than other methods. In customer-facing chatbots, companies are dealing with large amounts of user utterances that need to be clustered according to their intent. In these settings, no labeled data is typically available, and the number of clusters is not known. Recent approaches to short-text clustering in label-free settings incorporate LLM output to refine existing embeddings. While LLMs can identify similar texts effectively, the resulting similarities may not be directly represented by distances in the dense vector space, as they depend on the original embedding. We therefore propose a method for transforming LLM judgments directly into a bag-of-texts representation in which texts are initialized to be equidistant, without assuming any prior distance relationships. Our method achieves comparable or superior results to state-of-the-art methods, but without embeddings optimization or 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 how our method scales to large datasets, reducing the computational cost of the LLM use. The flexibility and scalability of our method make it more aligned with real-world training-free scenarios than existing clustering methods. Our source code is available here: https://github.com/tom192180/BoT_vector

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

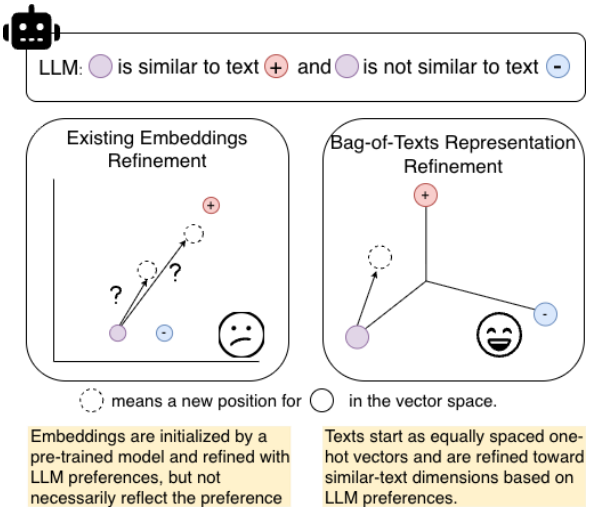


Figure 1: illustration of our proposed method. Left: traditional approach using the LLM output to refine the embeddings space. Right: our approach using the LLM output to directly construct a bag-of-texts space.

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 short-text 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.

Regardless the setting, the classic approach is to train an embedder with contrastive learning (Zhang

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 6432–6447

et al., 2021). In recent years, most methods have been designed to use LLMs to guide the optimization (Zhang et al., 2023; Zou et al., 2025b). Many LLM-guided training-free approaches have been proposed as well to enable easier domain adaptation and reduce the need for fine-tuning (De Raedt et al., 2023; Viswanathan et al., 2024b; Lin et al., 2025). Although these training-free approaches lead to substantial improvement, they only partially capture LLM-derived similarities and remain heavily dependent on the original embeddings. The resulting representation may still not reflect LLM preference, despite this being a strong signal. Hence our objective: to construct text representations that directly incorporate LLM similarity estimations into the vector space.

The core idea of our method (Figure 1) is to construct new representations (referred to as bag-of-texts vectors) such that texts belonging to the same cluster get more similar representations. Our method consists of two stages: we first construct bag-of-texts (BoT) representations based on representative texts, and then iteratively refine them by LLM guidance. In summary, the contributions of our method are as follows:

- **A Strategy for Encoding LLM Preferences** Unlike prior work that refines existing embeddings, we introduce a Bag-of-Texts (BoT) representation for encoding LLM outputs.
- **Flexible:** Our method can be added to any embedder and does not require fine-tuning, enabling easier adaptation to new domains.
- **Label-free and lightweight LLM-Based Approach:** Our method reduces the need for human labeling, can be implemented using relatively small, zero-shot models, and achieves results competitive with labeled or fine-tuned approaches.

2 Related Work

Short text clustering Generalized Category Discovery (GCD) is often studied for this task (Liang and Liao, 2023; Liang et al., 2024b; 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; Feng et al., 2024; Lin et al., 2025), where no labels are available. We follow this unsupervised scenario to support real world applications.

LLM-Guided Contrastive Learning for Short Text Clustering With the advancement of LLM, recent work has begun to incorporate LLM feedback into the optimization process. Zhang et al. (2023) use LLMs to construct hard triplets to derive training pairs. Liang et al. (2024b) identify samples in ambiguous regions and use LLMs to reassign them to new clusters, which are then used as training data. Li et al. (2025) use LLMs to enrich text representation and apply bidirectional refinement between LLMs and embedding models. Zou et al. (2025b) proposed a unified framework that actively learns from diverse and quality-enhanced LLM feedback, including instance-level, category description, and uncertain-instance alignment. However, these methods either rely on knowing the number of clusters, proprietary models, or fine-tuning. Our method does not have these requirements.

LLM-Guided Training-Free Approaches for Short Text Clustering Existing training-free approaches include direct cluster reassignment and embedding refinement. For example, Feng et al. (2024) propose LLMEdgeRefine; they perform edge-points reassignment on the clusters with LLM as an assessor; however, the method is built on an initially obtained clustering result and focuses on post-clustering refinement. In contrast, embedding refinement methods aim to improve representations for clustering. We categorize them into two types: The first category is input text enrichment: De Raedt et al. (2023) select prototypical utterances and then use an LLM to generate a set of descriptive utterance labels for these prototypes. Viswanathan et al. (2024a) use LLMs for few-shot in-context learning to improve texts by generating key phrases for each text. The second category is embedding post-processing: 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 its similar pairs. These methods have in common that they refine original embeddings. In contrast, we construct a new vector using LLM feedback.

3 Bag-of-Texts Theoretical Justification

3.1 Problem Definition

The task is to partition D^{test} into K clusters. K is treated as prior knowledge in most existing work (De Raedt et al., 2023; Zhang et al., 2023; Feng et al., 2024; Zou et al., 2025b), and many approaches incorporate K when refining embeddings. Our proposed method does not make any assumption on the knowledge of K thus can be implemented without knowing the number of clusters. For evaluation, however, we incorporate K for fair comparison to previous methods.

In line with prior work, we address two scenarios: (i) the emerging data scenario and (ii) the Generalized Category Discovery (GCD) scenario. In the emerging data scenario (Zhang et al., 2023; 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 size of the test dataset. The dataset is used as a whole for clustering and evaluation, without any train-test split. In contrast, the GCD scenario typically has access to additional data, typically including some *labeled* examples (Liang et al., 2024b; Zou et al., 2025b). As opposed to prior work, to align the GCD scenario with the objective of low-resource settings, we restrict the availability of additional data to only an *unlabeled* dataset $D^u \triangleq \{x_i\}_{i=1}^{N_u}$.

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 Embedding Refinement via Bag-of-Texts

Existing LLM-guided short-text clustering approaches often refine embeddings either directly using contrastive learning (Zhang et al., 2023), indirectly by text enrichment (De Raedt et al., 2023; Viswanathan et al., 2024b), or through embedding post-processing (Lin et al., 2025). While contrastive optimization enforces inequality through gradient updates, it requires training on new samples, which makes it impractical for low-resource settings. Text enrichment approaches, on the other hand, can perform for low-resource scenarios but lack explicit guarantees for pulling similar texts closer and pushing dissimilar ones apart. Embedding post-processing approaches heavily depend on the embedder used.

Our novel bag-of-texts representation combines the strengths of these approaches and directly enforces inequality through text-based LLM outputs, without requiring training data or expensive gradient-based updates of embedders. Our approach mimics contrastive learning by reconstructing the vector space using a bag-of-texts representation and updating this representation using LLM guidance. Using bag-of-texts representation, all texts start with equivalent distances in the vector space, and LLM output about similarity of texts to each other is directly translated to the vector representation, moving each text toward the correct region of the space; see Figure 1 for illustration.

Formally, for a text x, let l_{x,x^+} and l_{x,x^-} denote similarities between x and a positive and negative example, and l'_{x,x^+} and l'_{x,x^-} be the corresponding similarities after embedding refinement. The positive and negative examples are identified by a function (e.g., an LLM or a human), which determines whether the text x belongs to the same cluster as example x^+ (positive) or to a different cluster from the example x^- (negative). Ideally, our refinement approach should satisfy $l'_{x,x^+} > l'_{x,x^-}$. Below, we formally explain how our bag-of-texts representation achieves this goal.

Assumption 1. *We assume that a function (e.g., an LLM or a human) makes a selection based on the following rules.*

- *Similarity: For any texts x_i and x_j with $i \neq j$, x_i and x_j are similar if they belong to the same cluster S_k, and not similar if they belong to different clusters S_k and S_h with $k \neq h$.*
- *Distinct Clusters: Each text is assigned to exactly one cluster $S \in \mathcal{S}$.*

Note that $\mathcal{S}$ in Assumption 1 refers to the clusters defined by the function; there is no knowledge of a ground truth. We assume distinct clusters as is conventional in prior work (Zhang et al., 2023; An et al., 2024; Lin et al., 2025).

Definition 1 (Bag-of-Texts)**.** *Let $g(x, D)$ be a function that takes a text x and a set of texts D, and outputs a bag-of-text representation $\mathbf{z} \in \mathbb{R}^{|D|}$. In this representation, the entry corresponding to each text in D that belongs to the same cluster as x is assigned a value of 1, while all other entries are 0.*

Proposition 1. *Given Assumption 1, for any $x_i, x_j \in D$ and their corresponding bag-of-texts representation $\mathbf{z}_i, \mathbf{z}_j$, the function $g(x, D)$ guarantees that $l'_{x,x^+} > l'_{x,x^-}$.*

The proof of the proposition builds on *Definition 1* that if two texts $x_i, x_j \in D$ belong to the

same cluster, their bag-of-texts representations $\mathbf{z}_i$ and $\mathbf{z}_j$ are identical. Considering cosine similarity as an example similarity function, $\text{cosine}(\mathbf{z}_i, \mathbf{z}_j) = 1$ if they are identical, and $\text{cosine}(\mathbf{z}_i, \mathbf{z}_j) = 0$ if they are dissimilar.

3.3 Adaptation to Real-world Practice

Although the proposition holds in the idealized setting, in practice the similarity between texts is typically determined using LLMs, which may introduce errors, and scanning the entire dataset for each x is computationally expensive. We outline the modifications needed for practical implementation. The detailed analysis is in appendix A.

Memory constraints. Large datasets lead to high-dimensional bag-of-text vectors. To address these issues, we construct bag-of-texts vectors using a representative subset $D^r \subseteq D$ with $|D^r| \triangleq d$, rather than all texts. This dimensionality reduction is possible because it relies on the assumption that texts within the same cluster are similar; therefore, it is not necessary to use all texts in D to represent bag-of-texts vectors. Instead, the bag-of-texts vectors of the remaining texts can be represented via the vectors of the selected representatives.

Computation constraints. Selecting similar texts requires scanning the entire dataset for each text, which is computationally expensive with an LLM. To reduce computation, we use a pre-trained embedder to retrieve a small set of m candidate texts that are likely to be similar, as supported by prior work (Lin et al., 2025; Zou et al., 2025a).

Selection uncertainty. During automatic selection (e.g., by LLM) inconsistent preferences can occur and violate Assumption 1. For example, (x_i, x_j) may be identified as belonging to the same cluster, (x_i, x_l) as different, yet (x_j, x_l) as the same. Also, in practice, each selection is made with some inherent confidence. To reduce these inconsistencies, we perform iterative update and assign the mean of the selected bag-of-texts vectors rather than a fixed value of 1.

4 Computational Method

Our method aims to construct Bag-of-Texts (BoT) vectors that capture the similarity of texts within the same cluster. It consists of two stages. First, we select representative texts, which are then used to construct a BoT vector for each text. Second, we iteratively update each text's BoT vector by considering its similar texts until convergence.

Algorithm 1 Select Representative Texts

Require: D, $\mathbf{X}$, representative texts size d
Ensure: Initial Representative Texts D^r, $\mathbf{Z}^r \in \mathbb{R}^{d \times d}$
$\mathcal{C} = \{C_1, \cdots, C_d\} \leftarrow \text{Agg}(\mathbf{X}, d), \quad C_j \subseteq \mathbf{X}$
$D^r \leftarrow \emptyset$
for $j = 1, \ldots, d$ **do**
  $\mathbf{x}_j^r = \arg\min_{\mathbf{x} \in C_j} \sum_{\mathbf{y} \in C_j} \|\mathbf{x} - \mathbf{y}\|_2$
  $D^r \leftarrow D^r \cup \{x_i \in D : \mathbf{x}_i = \mathbf{x}_j^r\}$
end for
Initialize $\mathbf{Z}^r \leftarrow I_d$ {one-hot vectors for D^r}
return D^r, $\mathbf{Z}^r$

Algorithm 2 Construct Bag-of-Texts Vectors

Require: $D \setminus D^r$, $\mathbf{X}$, D^r, $\mathbf{Z}^r$, candidate size m, LLM
Ensure: Bag-of-texts vectors $\mathbf{Z} \in \mathbb{R}^{N \times (d + d^*)}$
Initialize $\mathbf{z}_j \leftarrow \mathbf{0}$ for all $x_j \in D \setminus D^r$
for each $x_j \in D \setminus D^r$ **do**
  $M_j \leftarrow \{x_{j1}, \ldots, x_{jm}\} \in D^r$ from $\mathbf{X}$ as the top-m by cosine similarity to x_j
  $O_j \leftarrow \text{LLM}(M_j) \in \mathcal{P}(M_j) \cup \{\emptyset\}$ {# $\mathcal{P}(.)$: power set}
  if $O_j \neq \emptyset$ **then**
    $\mathbf{z}_j \leftarrow \text{MeanPooling}\{\mathbf{z}_k : x_k \in O_j\}$
    $\mathbf{z}_j \leftarrow \frac{\mathbf{z}_j}{\|\mathbf{z}_j\|_2}$
  end if
end for
$D^{r^*} \leftarrow \{x_j : \mathbf{z}_j = \mathbf{0}\}$ {# new reps. $|D^{r^*}| = d^*$}
$D^r \leftarrow D^r \cup D^{r^*}$ {# $|D^r| = d + d^*$}
Let $\mathbf{P} \leftarrow \{\mathbf{z}_j : \mathbf{z}_j = \mathbf{0}\}$; one-hot encode each $\mathbf{z}_j \in \mathbf{P}$ in new dimensions, and pad the first d columns with zeros.
Let $\mathbf{Q} \leftarrow \{\mathbf{z}_j : \mathbf{z}_j \neq \mathbf{0}\}$; pad $\mathbf{Q}$ and $\mathbf{Z}^r$ with d^* zero columns
return $\mathbf{Z} \leftarrow \text{Stack}(\mathbf{Z}^r, \mathbf{P}, \mathbf{Q})$, rows ordered according to D

4.1 Initial stage: construction of BoT representation

We first encode all texts D with a pre-trained embedder to obtain embeddings $\mathbf{X}$. To select initial representative texts D^r, we perform agglomerative clustering (Murtagh and Legendre, 2014) to partition them into d clusters, d being a trivially large number (see Section 5.3). The medoid of each cluster is chosen as a representative and converted into a one-hot vector with initially equal spacing. See Algorithm 1 for specification.

The remaining texts are positioned based on LLM preference. Specifically, for each text $x_j \in D \setminus D^r$, we retrieve the top m representative texts from the pretrained embeddings $\mathbf{X}$ that are closest based on cosine similarity. An LLM is then used to select which representative(s) are similar. The bag-of-texts vector for each text is computed by averaging the vectors of its selected representatives. After processing all texts, those with no selected

representatives are promoted to new representatives and added to D^r (so that $|D^r| = d + d^*$, where d^* is the number of such new representatives), and the dimensionality of the BoT vectors is expanded to match. See Algorithm 2 for specification.

This procedure ensures that the initial BoT vectors are evenly spaced for the representative texts, while the vectors for the remaining texts encode LLM-informed relationships, avoiding the need to use the full dimensionality of all texts.

4.2 Iterative stage: bag-of-texts vectors refinements

After constructing the initial positions for all texts, we iteratively update the BoT vectors to better align their relative distances with LLM preferences. The update procedure follows the same approach as in Algorithm 2, with three key modifications. First, the update uses all texts as the pool for subset selection, since each text has a BoT vector. Second, for each text, we select a subset of m candidates based on the concatenation of its pre-trained embedding and BoT vector. This ensures that the pre-trained embedding initially guides the BoT vectors, which carry minimal information at the start. Third, the update is performed iteratively rather than just once, allowing the candidate set to vary dynamically as the BoT vectors are refined. The update for a text stops when its BoT vector changes negligibly, as measured by a cosine similarity greater than 0.99. Note that the pretrained embeddings are used only for updating the BoT vectors; only the final BoT vectors $\mathbf{Z}^T$ will be used for clustering. We set $m = 30$ and cap the maximum number of iterations at $T = 10$ as budget. See Algorithm 3 in Appendix B for specification.

5 Experimental Settings

5.1 Datasets

We use a variety of datasets to benchmark our method: the intent datasets Bank77 (Casanueva et al., 2020), CLINC150 (Larson et al., 2019), Mtop (Li et al., 2020), and Massive (Fitzgerald et al., 2023); the emotion dataset GoEmo (Demszky et al., 2020), and the community QA dataset Stackoverflow (Xu et al., 2015). Statistics are provided in Table 1. Among these, MTOP, Massive, and GoEmo have highly imbalanced clusters.

Emerging data scenario	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 scenario	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 unlabeled training set. Further details are provided in Appendix C.

5.2 Models

To see how sensitive our approach is to the choice of embedder, we design experiments that incorporate multiple models. We experiment in the **emerging data scenario** using TF–IDF, bert-base-uncased (Devlin et al., 2018), all-MiniLM-L6-v2 (Reimers and Gurevych, 2019), and e5-large (Wang et al., 2022). To see how far our method is from semi-supervised approaches and the effect with additional data, in the **GCD scenario**, we use all-MiniLM-L6-v2 as the embedder following prior work (Rodriguez et al., 2024). For the LLM experiments, the main results are obtained using gemma-2-9b-it (Rivière et al., 2024). An A100 GPU or an H100 GPU were used throughout the experiments.

Prompts used The prompts we used for the LLM are shown in Table 2. They include three components: task instruction, answer format, and task description. Following prior work (Zhang et al., 2023; De Raedt et al., 2023; Viswanathan et al., 2024a; Lin et al., 2025; Zou et al., 2025b), we keep the prompts simple to ensure a fair comparison and generalizability between models.

5.3 Hyperparameter setting

The main hyperparameter in our approach is the initial dimension d (number of initial representatives). Although we can simply set the number of d equal to the total number of examples N, this becomes memory-intensive when N is extremely large. Following the approach in (Feng et al., 2024; Lin et al., 2025), we determine hyperparameters using external datasets to avoid biasing evaluation on the test sets. The search space starts from 512, setting a sufficiently large minimum to ensure that d to in-

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.

clude at least one representative from each cluster. We fix the BoT vector dimension to 1,024, based on validation results from two external datasets, including ArxivS2S and Reddit (Muennighoff et al., 2023). Empirically, we find that performance remains very similar between different values of d (see Appendix D).

5.4 Evaluation

The Bag-of-Text vectors are evaluated under K-means. K-means clustering is a widely adopted evaluation practice in short text clustering (Zhang et al., 2023; De Raedt et al., 2023; Viswanathan et al., 2024b; Liang et al., 2024b; Lin et al., 2025; Zou et al., 2025b). Following convention, the number of clusters for K-means is set to match the number of ground-truth cluster number K during evaluation. K-means is run over 5 different seeds and averaged. 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

We compare our method with SOTA representation learning baselines for text clustering. Table 3 summarizes these methods. For the **emerging data scenario**, we compare against IDAS (De Raedt et al., 2023), ClusterLLM (Zhang et al., 2023), DG-CoE (Li et al., 2025), and SPILL (Lin et al., 2025).[1] For the **GCD Scenario** we compare against IntentGPT (Rodriguez et al., 2024). We also include comparisons with methods that require some labeled data, including LOOP (An et al., 2024), ALUP (Liang et al., 2024b) and Glean (Zou et al., 2025b) to measure how far our method is from the performance of these semi-supervised approaches.

	Unsup	LLM	FT	K
LeBoT (Ours)	✓	Gemma		
IDAS (De Raedt et al., 2023)	✓	Gemma	✓	✓*
ClusterLLM (Zhang et al., 2023)	✓	GPT-*	✓	✓
IntentGPT (Rodriguez et al., 2024)	✓*	GPT-*		
ALUP (Liang et al., 2024b)		GPT-*	✓	✓
LOOP (An et al., 2024)		GPT-*	✓	✓
SPILL (Lin et al., 2025)	✓	Gemma		
DG-CoE (Li et al., 2025)	✓	Qwen	✓*	✓
Glean (Zou et al., 2025b)		GPT-*	✓	✓

Table 3: LLM-based representation learning for short-text clustering. **Unsup**: No access to labels. IntentGPT can be semi-supervised as well. **LLM**: Model used in the main results (Gemma = Gemma2-9b-it; GPT-* = GPT variants with unknown parameters; Qwen = Qwen2.5-7B-Instruct). **FT**: Fine-tuning the embedder. Note that DG-CoE fine-tunes both the LLM and the embedder. **K**: Incorporates the number of clusters K into embedding refinement. IDAS can be applied either without knowing K or knowing K

6 Results

6.1 Main results

Emerging data scenario Table 4 reports the K-means results. Our approach consistently outperforms pretrained embeddings and on average performs better than all other baselines, including those that require fine-tuning (ClusterLLM and DG-CoE). Furthermore, our approach maintains consistent performance across backbones, unlike SPILL and IDAS, which degrade more when the embedder changes. We also observe that with the stronger E5 embedder, which has the highest zero-shot performance, the gap between methods narrows. This is likely because E5 provides richer semantic representations and better separation in the embedding space, making it easier for both text-enrichment approaches (IDAS) and embedding post-processing methods (SPILL) to perform well. See Appendix E for detailed embedder analysis.

[1] LLMEdgeRefine is a post-clustering refinement method and therefore not directly comparable with the other approaches considered in this work.

	Bank 77		Clinc150		Mtop		Massive		GoEmo		**Average**	
	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc
TF-IDF												
Plain	59.76 (0.31)	36.73 (0.99)	58.05 (0.43)	33.44 (1.11)	50.77 (0.84)	30.98 (1.19)	45.24 (1.55)	28.35 (1.61)	18.49 (1.65)	19.64 (1.25)	46.46	29.83
IDAS	72.62 (0.28)	51.96 (0.78)	76.37 (0.68)	54.44 (1.04)	63.95 (0.11)	39.47 (2.90)	58.07 (1.08)	40.82 (1.27)	19.49 (0.18)	21.04 (0.47)	58.10	41.55
IDAS†	72.68 (0.20)	52.11 (0.60)	85.12 (0.14)	69.72 (0.29)	65.58 (0.24)	38.33 (0.45)	65.14 (0.32)	46.77 (0.85)	23.39 (0.36)	22.48 (0.45)	62.38	45.88
SPILL	63.73 (0.67)	42.36 (1.14)	59.72 (1.05)	36.74 (0.92)	49.54 (1.41)	28.26 (0.94)	43.26 (0.85)	28.06 (1.28)	9.51 (0.44)	13.48 (0.20)	45.15	29.78
LeBot (Ours)	**80.26** (0.28)	**65.45** (1.43)	**88.97** (0.11)	**77.70** (1.03)	**70.15** (0.23)	**40.00** (1.30)	**69.60** (0.06)	**54.52** (0.69)	**27.89** (0.30)	**25.82** (0.34)	**67.37**	**52.70**
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
IDAS	70.30 (0.32)	49.85 (0.49)	87.97 (0.27)	71.57 (1.02)	68.90 (0.29)	33.08 (0.69)	65.24 (0.32)	49.06 (0.59)	14.10 (0.52)	15.22 (0.73)	61.30	43.76
IDAS†	72.73 (0.26)	53.45 (0.82)	88.70 (0.18)	74.99 (0.61)	69.52 (0.21)	33.19 (0.87)	66.41 (0.44)	50.04 (0.94)	14.89 (0.26)	16.50 (0.52)	62.45	45.63
SPILL	65.67 (0.49)	43.44 (1.02)	86.91 (0.28)	71.39 (1.28)	69.50 (0.28)	32.49 (0.40)	62.67 (0.29)	43.64 (0.81)	12.30 (0.32)	13.86 (0.44)	59.41	40.96
LeBoT (Ours)	**76.86** (0.24)	**60.95** (1.11)	**92.90** (0.05)	**84.47** (0.46)	**72.28** (0.24)	**38.69** (1.46)	**73.16** (0.27)	**61.02** (0.82)	**19.25** (0.19)	**18.77** (0.14)	**66.89**	**52.78**
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
IDAS	83.01 (0.19)	65.96 (1.19)	92.82 (0.22)	80.05 (0.21)	70.46 (0.46)	35.55 (1.33)	75.67 (0.47)	59.52 (2.12)	**22.27** (0.16)	**27.22** (0.37)	68.85	53.66
IDAS†	84.88 (0.19)	72.77 (0.59)	93.41 (0.06)	85.52 (0.33)	70.88 (0.22)	36.46 (1.46)	75.93 (0.17)	57.75 (0.57)	21.25 (0.25)	25.67 (0.45)	69.27	55.63
SPILL	81.92 (0.16)	66.22 (0.46)	90.78 (0.22)	77.76 (0.07)	70.48 (0.41)	37.95 (1.27)	72.59 (0.31)	58.05 (1.40)	15.54 (0.31)	17.22 (0.75)	66.26	51.44
LeBoT (Ours)	**85.44** (0.18)	**73.12** (1.10)	**94.45** (0.08)	**87.58** (0.70)	**72.18** (0.29)	**39.40** (0.83)	**76.74** (0.24)	**66.84** (0.67)	21.77 (0.12)	25.22 (0.74)	**70.12**	**58.43**
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
IDAS	83.91 (0.24)	68.78 (1.11)	93.53 (0.26)	82.20 (1.22)	73.58 (0.37)	38.87 (1.04)	76.70 (0.89)	59.61 (3.25)	26.30 (0.17)	29.56 (0.64)	70.80	55.80
IDAS†	83.37 (0.14)	70.01 (0.59)	94.07 (0.09)	85.38 (0.35)	73.09 (0.45)	39.84 (1.24)	**77.29** (0.15)	**63.64** (0.80)	25.99 (0.09)	28.72 (1.08)	70.76	57.52
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
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
DG-CoE†	83.64	70.54	**94.35**	**87.67**	74.01	37.75	75.65	60.80	25.00	28.20	70.53	56.99
LeBoT (Ours)	**85.34** (0.06)	**73.86** (0.76)	93.63 (0.06)	85.65 (0.46)	72.51 (0.32)	**40.53** (1.64)	76.16 (0.17)	62.75 (0.67)	**26.70** (0.04)	**33.70** (0.26)	**70.87**	**59.30**

Table 4: Five-benchmark results (**emerging data scenario**). Averages over 5 runs (±SD). Gemma is used in LeBoT, SPILL and IDAS. **Plain**: direct embedding clustering. Bold numbers: best within the embedder. ClusterLLM and DG-CoE cited directly from prior work. † is with access to information of known K during representation refinements.

GCD scenario Table 5 shows the results for GCD. The goal is to examine how increasing the amount of data affects performance and to see the gap between our approach and semi-supervised learning. Our method achieves the best performance on all benchmarks in the fully unlabeled setting. 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 suggest that our approach can substantially reduce the need for human labeling.

KCR	Method	Bank 77 NMI	Bank 77 Acc	Clinc150 NMI	Clinc150 Acc	StackO NMI	StackO Acc	Average NMI	Average Acc
0%	IntentGPT	81.42	64.22	94.35	83.20	-	-	-	-
	LeBoT (Ours)	**86.90**	74.93	95.32	88.71	**81.21**	82.54	**87.81**	82.06
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 5: Three-benchmark results (**GCD scenario**). Averages over 5 runs. KCR: Known Categories Ratio. All-MiniLM-L6-v2 used as in LeBoT. Bold: best. Underline, second best. StackO: StackOverflow dataset. IntentGPT, GLEAN and ALUP cited directly from prior work. LOOP cited directly from GLEAN. † is with access to information of known K during representation refinements.

6.2 Further analysis

Dimensionality of the bag-of-text representations We observe that the increase in dimensionality by d^* is negligible: for most runs, the values fall between 0 and 20, and the maximum increase across all runs, datasets, and backbone embedders is only 44. Since we start with a relatively large initial dimension of $d = 1,024$, this increase is very limited. This is expected, as each text is more likely to identify similar counterparts when the pool of candidate texts is large. We also tested other initial dimensions, and Figure 2 shows that the results remain robust across different initial settings, as long as the initial dimension is not too small.

Effect of the Number of subset candidates m and Iterative Updates We use $m = 30$ across all experiments, but this raises the question of whether different values of m would affect performance. Figure 3a shows that small size of $m = 10$ leads to lower results because the LLM has too few candidates to choose from. However, once $m \geq 20$, performance remains largely stable. To investigate the effect of iterative updates, Figure 3b shows that in subsequent iterations, the results substantially

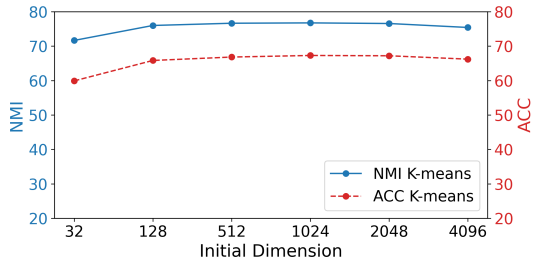
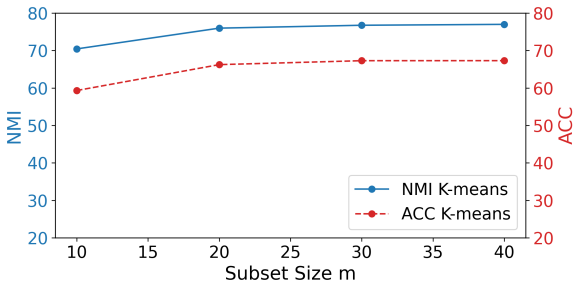
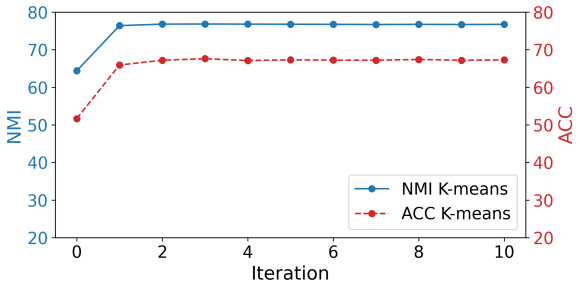


Figure 2: ACC and NMI across all test datasets (Emerging Data and GCD scenarios) for different d. $4,096$ means $\min\{4096, N\}$ since some datasets are smaller.



(a)



(b)

Figure 3: Comparison of (a) varying neighbor m and (b) iterative updates on ACC and NMI across all test datasets (Emerging Data and GCD). Backbone embedder: All-MiniLM-L6-v2.

improve, especially after the first iteration, but no longer after the second or third. This is due to the early convergence of most text representations. As shown in Figure 4, most datasets reach a 90% convergence ratio by the fifth iteration.

Effect of Different LLMs Our main results were obtained with gemma-2-9b-it. For completeness, we also experimented with alternative LLMs. Importantly, because our objective is to evaluate whether the vectors encode LLM-derived selections rather than to benchmark LLM performance, these analyses are presented as ancillary robustness checks. Figure 5 shows the results obtained with three different LLMs. The results between Qwen and Gemma are similar. There is a slight drop

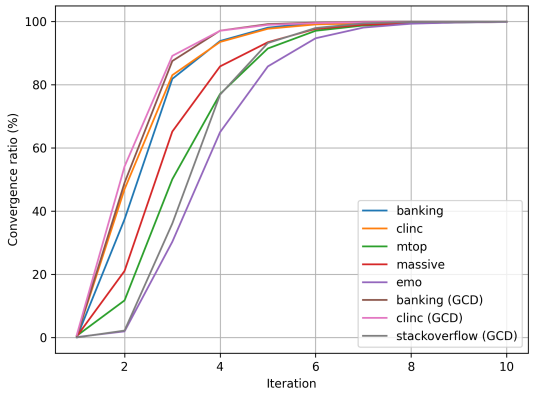


Figure 4: Ratio of texts converged at iteration t. All-MiniLM-L6-v2 (as embedder) and gemma-2-9b-it (as LLM) were used.

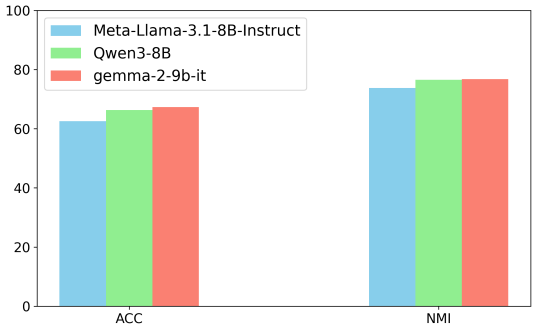


Figure 5: ACC and NMI across all test datasets (Emerging Data and GCD) for three LLMs. Backbone embedder: All-MiniLM-L6-v2

with Llama. This finding is consistent with prior work (Lin et al., 2025), as Llama produces more incorrect selections (see Appendix F for details).

Effect of Different Prompts Because all of the experiments are conducted on the same set of prompts, we further analyze the effects of prompt variation. We experiment with two additional prompts in the emerging data scenario (five datasets) using the backbone models Gemma-2-9b-it and MiniLM-L6-v2 (see Appendix G for the prompts). The average results across these datasets show that, for the three prompts, the NMI ranges from 69.38 to 70.12, and the ACC ranges from 57.52 to 58.43, indicating minimal differences between prompts. We hypothesize that this is due to the use of relatively general prompts, which reduces overfitting to the datasets and leads to lower variance.

Evaluation with different clustering algorithms Our approach does not require the number of clusters K when building the representations. We there-

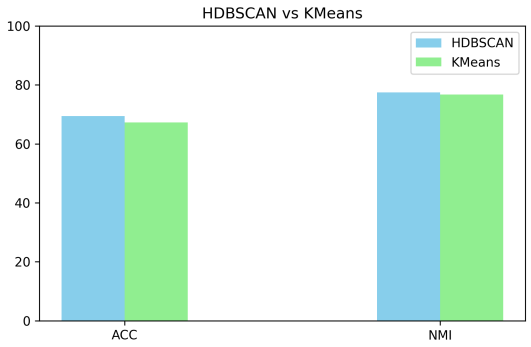


Figure 6: ACC and NMI across all test datasets (Emerging Data and GCD) for HDBSCAN and K-means. Backbone embedder: All-MiniLM-L6-v2

fore also evaluate it with HDBSCAN, which estimates the number of clusters automatically and is therefore closer to real-world use. Figure 6 shows that performance remains similar, indicating that our method is robust across different clustering algorithms. This confirms its practical applicability. Detailed implementation in Appendix H.

Complexity and Runtime Analysis The maximum total number of LLM calls (assuming no early convergence) for a dataset of size N is :

$$\frac{m}{c} \cdot \left[(N-d) + N \cdot T\right]$$

Here, $N-d$ corresponds to the initial stage, $N \cdot T$ corresponds to the iteration stage, d is number of the initial representative texts, T is the maximum number of iterations, m is the total number of candidates, and c is the number of candidates for similarity classification per call. Our approach scales linearly with the dataset size. We use $m = 30$, $c = 10$, $T = 10$, and $d = 1,024$ in our main results. We report results for the emerging data scenario using the embedder All-MiniLM-L6-v2 and the backbone LLM Gemma-2-9b-it, run on a single H100 GPU. All tasks can be completed within a few hours. **Banking77** (3,080 examples): 2,703s. **CLINC150** (4,500 examples): 3,167s. **Mtop** (4,386 examples): 3,438s. **Massive** (2,974 examples) : 1,998s. GoEmo (5,940 examples): 6,210s.

Increasing scalability with distillation Our representation requires on average 3 iterations to reach stable results, which can be computationally expensive when the dataset is extremely large. To assess scalability, we additionally explore clustering over the full CLINC (22,500 examples) and Banking (13,083 examples) datasets. The goal is to implement our approach on a subset of the texts (around 20%) and then use a model to generalize to the remaining texts (a form of knowledge distillation), without requiring further LLM selection. Specifically, we train a lightweight MLP (∼10M parameters) to map pre-trained embeddings to BoT vectors. See Appendix I for experiment details. Table 6 shows that our approach still improves the original embeddings by a large margin. During inference, our method with LLM selection took $0.7038s$ (CLINC) and $0.8778s$ (Banking) per text, while our distilled method with MLP only takes $0.0002s$ per text for both.

	Bank 77		Clinc150	
Method	NMI	Acc	NMI	Acc
K-means				
Plain	76.31	59.41	88.99	73.99
LeBoT (Ours)	81.44	68.99	91.19	83.62
HDBSCAN				
Plain	76.15	60.27	84.31	62.84
LeBoT (Ours)	81.35	68.01	91.50	85.28

Table 6: Clustering results on the entire large datasets using knowledge distillation. Averages over 5 runs for K-means. Plain: Directly embedding the test dataset with All-MiniLM-L6-v2.

7 Conclusions

We propose LeBoT (LLM-enabled Bag-of-Texts), an intuitive, domain-adaptive, label-free, and training-free method for short text clustering. Compared with other SOTA methods, LeBoT is less dependent on careful selection of the embedder and does not assume prior knowledge of clusters or labels, making it more aligned with real-world scenarios. LeBoT achieves better or similar results to state-of-the-art approaches, including methods that require labeled data. Experiments on diverse datasets shows our method is the most robust across different embedders, and experiments with smaller LLMs show that our method is robust in low-resource settings, while also scalable to large datasets. In the future, we plan to incorporate human feedback into our method pipeline.

Acknowledgments

This publication is part of the project LESSEN with project number NWA.1389.20.183 of the research program NWA ORC 2020/21 which is (partly) financed by the Dutch Research Council (NWO).

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 focuses on encoding preferences through Bag-of-Text vectors. Its effectiveness naturally depends on the LLM, and in highly specialized domains, some in-context learning may be required to improve the selection.

Embedder capability. Although we construct a Bag-of-Text vector that reduces reliance on the existing embedder, for computational efficiency we still retrieve subset of candidate texts using the embedder. This will affect the quality of the constructed vectors if the embedder is very poor. This can affect the quality of the constructed vectors when the embedder is very weak. However, our main results show that this influence is relatively small compared with other methods.

References

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*.

Zetong Li, Qinliang Su, Minhua Huang, and Yin Yang. 2025. Co-evolving llms and embedding models via density-guided preference optimization for text clustering. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 4796–4808.

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.

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.

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 Impact of our computational method on preference preservation

In this section, we analyze the impact of dimension reduction and subset selection on the preservation of preference, assuming that the stated Assumption 1 hold. We do not discuss pooling here, because pooling is intended to handle cases where Assumption 1 is violated, and to account for the estimation involved in subset selection.

Definition 2 (Preservation of Preference)**.** *We say that a function* preserves preferences *if* $l_{x,x^+} > l_{x,x^-}$*, where* l *denotes the cosine similarity,* (x, x^+) *is a similar pair, and* (x, x^-) *is a disimilar pair.*

We need a generalized version of g to consider the dimension reduction and subset selection.

Definition 3 (Bag-of-Texts: generalization version of g)**.** *Let* $g^*(x, \mathcal{T}, \mathcal{L}(x))$ *be a function that takes:*

- *a text* x*,*
- *a set of texts* D*,*
- *a subset of texts* $\mathcal{T} \subseteq D$*,*
- *a subset of texts* $\mathcal{L}(x) \subseteq \mathcal{T}$*, selected based on* x

and outputs a BoT representation $\mathbf{z} \in \mathbb{R}^{|\mathcal{T}|}$*. In this representation, the entry corresponding to each text in* $\mathcal{L}(x)$ *that is considered similar to* x *is assigned a value of 1, while all other entries are 0.*

The definition of the generalized function g^* is to consider the reduced dimension of BoT (memory constraints) and subset selection (computation constraints). Note $g^*(x, D, D, D) = g(x, D)$.

Proposition 2 (Preservation of Preference on generalization function g^*)**.** *Let* $D^r \subseteq D$ *be such that for every cluster* $S \in \mathcal{S}$*, there exists at least one text* $y \in D^r$ *with* $y \in S$*.*

Let $M^* = \mathcal{L}(x) \subseteq \mathcal{T}$ *such that, for every cluster* $S \in \mathcal{S}$*, there exists at least one text* $o_S \in S$ *that is included in* M^* *for* $x \in S$*.*

$g^*(x, D, D^r, M^*)$ *is a function that preserves the preference under Assumption 1.*

In the preposition 2, the dimensions constructed using D^r ensure that each cluster has at least one dimension in which a value can be assigned. The subset M^* guarantees that BoT vectors corresponding to texts within the same cluster share a 1 in at least one of these dimensions. As a result, the cosine similarity between vectors of texts from the same cluster is strictly positive due to the shared dimension(s). Furthermore, the more shared texts from the cluster that appear in M^* for each x, the greater the overlap of 1s, and thus the higher the cosine similarity.

This proposition 2 implies two key insights. First, instead of turning every text $x \in D$ into a dimension of the BoT vector, we may use only a subset, provided that the subset is sufficiently representative. Second, scanning the entire dataset for every x can be replaced with scanning only such a subset, as long as it contains at least one shared element for each cluster.

Remark 1: The proposition 2 is primarily intended for conceptual understanding. The shared text o_S and subset of texts M_j^* can also include the texts from remaining texts $D \setminus D^r$ not just those in D^r since these remaining texts are initially constructed with D^r. This increases the number of shared texts available across clusters, providing more opportunities for matching and potentially improving similarity estimates.

Remark 2: As mentioned in the section 3 and 4, in practice, we select a fairly large initial dimension d and use an LLM to include missing representative texts to form D^r. For subset selection, for each x_j, we use pre-trained embeddings to select the top m closest texts, denoted M_j, to approximate M_j^*.

B Algorithm for Updating the Bag-of-Texts Vectors

The BoT vector update largely involves a repetitive process similar to the original construction. See Algorithm 3 for details.

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

Table 7 shows the train–validation–test splits used in 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.

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

Algorithm 3 Update Bag-of-Texts Vectors

Require: D, $\mathbf{X}$, $\mathbf{Z}$, candidates size m, max iterations T, LLM
Ensure: Updated Bag-of-texts vectors $\mathbf{Z}^T \in \mathbb{R}^{N\times(d+d^*)}$

Initialize $\mathbf{Z}^0 \leftarrow \mathbf{Z}$
for $t = 0$ to $T-1$ **do**
  $\mathbf{U}^t \leftarrow \mathbf{X} \oplus \mathbf{Z}^t$
  for each $x_j \in D$ **do**
    if $t > 0$ **and** $\cos(\mathbf{z}_j^{t-1}, \mathbf{z}_j^t) > 0.99$ **then**
      $\mathbf{z}_j^{t+1} \leftarrow \mathbf{z}_j^t$ {# Skip updating}
    else
      $M_j^t \leftarrow \{x_{j1}^t, \ldots, x_{jm}^t\}$ from $\mathbf{U}^t$ as the top-m by cosine similarity to x_j
      $O_j^t \leftarrow \text{LLM}(M_j^t) \in \mathcal{P}(M_j) \cup \{\emptyset\}$ {# $\mathcal{P}(.)$: power set}
      if $O_j^t \neq \emptyset$ **then**
        $\mathbf{z}_j^{t+1} \leftarrow \text{MeanPooling}\{\mathbf{z}_k^t : x_k \in O_j^t\}$
        $\mathbf{z}_j^{t+1} \leftarrow \frac{\mathbf{z}_j^{t+1}}{\|\mathbf{z}_j^{t+1}\|_2}$
      else
        $\mathbf{z}_j^{t+1} \leftarrow \mathbf{z}_j^t$
      end if
    end if
  end for
  $\mathbf{Z}^{t+1} \leftarrow \{\mathbf{z}_j^{t+1}\}_{x_j \in D}$
end for
return $\mathbf{Z}^T$

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

They train an embedder using the training split and optimize 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.

D Selection of d

gemma-2-9b-it (as LLM) and All-MiniLM-L6-v2 (as embedder) are used in the hyperparameter search. Table 8 shows the two datasets that we used for selecting the dimensionality of the BoT vector d. Figure 7 shows that they all give similar performance. We selected value $d = 1024$ and used this value across all datasets and experiments.

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

Table 8: External datasets used for selecting the dimensionality hyperparameter d

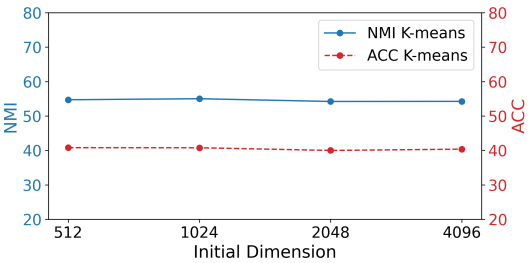


Figure 7: The average ACC and NMI across two datasets for different d. Results for K-means are averaged over 5 runs. 4096 here means $\min\{4096, N\}$ because the dataset size can be smaller.

E Effect of Different Embedders

Because we use a subset instead of scanning the entire dataset for each text, the quality of the texts selected by the embedder influences the construction of our BoT vector. We report two ratios of the first iteration, averaged across all texts.

Embedder Selection Ratio:

$$\frac{\#\{\text{True same-cluster in top } m \text{ as the text}\}}{m}$$

In our experiments, we set $m = 30$. Note that Embedder here does not refer to the pre-training embeddings alone; it denotes the concatenation of the pre-training embeddings and the BoT vectors. We focus on the first iteration, i.e., $\mathbf{U}^1$, since all BoT vectors have not yet converged and they are largely evenly spaced and uninformative in the initial state. As a result, the selection relies mostly on the pre-training embeddings.

LLM Selection Ratio (LLM):

$$\frac{\#\{\text{LLM-selected items truly in the same cluster as the text}\}}{\#\{\text{ items selected by LLM}\}}$$

Table 9 shows that E5 produces the best pool on average. We also observe that the LLM ratio remains relatively stable, even when the embedder ratio fluctuates substantially.

	Bank 77		Clinc150		Mtop		Massive		GoEmo		Average	
	Emb	LLM	Emb	LLM	Emb	LLM	Emb	LLM	Emb	LLM	Emb	LLM
TF-IDF	43.26	68.02	49.71	82.60	62.93	78.55	45.28	70.21	**30.85**	**36.32**	46.41	67.14
Bert	38.08	63.41	56.12	86.49	69.72	82.48	50.48	73.61	20.02	25.90	46.88	66.38
MiniLM	**62.03**	**75.55**	**69.76**	**89.09**	70.88	81.27	61.58	76.14	22.33	27.79	57.32	69.97
E5	58.64	73.25	67.80	88.48	**72.00**	**81.53**	**61.66**	**76.33**	27.97	32.02	**57.61**	**70.32**

Table 9: Selection Accuracy By Embedders at 1st iteration (%)(**emerging data scenario)**. Bold numbers: best. LLM is Gemma.

F Results by Different LLMs

We report the detailed results of all datasets by different LLMs in Table 10. Qwen and Gemma consistently outperform LLaMA. We also report the LLM selection accuracy (see Appendix E for the definition). Table 11 shows that Gemma and Qwen have similar capabilities in identifying similar texts, outperforming LLaMA. This finding aligns with prior work (Lin et al., 2025).

G Varying the Prompts

We only adjust task instructions; the answer format is kept the same as the original:

CLINC150, **Bank77**, **Mtop**, and **Massive**:
Prompt variant 1: Compare the Target Utterance with each Candidate Utterance and identify which candidates share the same intent as the Target. Here, intent refers to the user's underlying request or goal.

Prompt variant 2: Examine each Candidate Utterance in relation to the Target Utterance and select those that express the same user intent. Intent means the purpose or objective the user aims to accomplish.

GoEmo:
Prompt variant 1:Compare the Target Utterance with each Candidate Utterance and identify which candidates express the same emotion as the Target.

Prompt variant 2: Examine each Candidate Utterance in relation to the Target Utterance and select those that convey the same emotion as the Target.

H HDBSCAN Clustering

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). Using the silhouette score for hyperparameter search follows prior work (Gung et al., 2023).

Tables 12 and 13 shows the detailed evaluation result with HDBSCAN clustering.

Table 14 and Table 15 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. Except for the Bank77 and GoEmo datasets.

I Details of the full dataset clustering

Datasets Used with LLM Guidance Table 16 shows the number of examples we use with the LLM. We re-use the representations derived in the emerging data setting, so the number of examples with LLM guidance remains the same as in Table 1, i.e., $|D^{test}|$, and there is no need to re-run all experiments. The examples account for 20.6% to 23.5% of datasets. Examples with LLM guidance are used to train the MLP, since their BoT vectors have been derived. The pre-trained embeddings from all-MiniLM-L6-v2 serve as input, and the LLM-generated BoT 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 BoT versions of these embeddings. These BoT vectors are used to cluster all examples.

	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
Gemma	85.44	73.12	94.45	87.58	72.18	39.40	76.74	66.84	21.77	25.22	86.90	74.93	95.32	88.71	81.21	82.54	76.75	67.29
Qwen	85.01	71.63	93.79	87.12	73.29	37.45	74.24	61.47	21.61	25.83	86.53	74.31	95.61	89.41	82.15	83.42	76.53	66.33
Llama	81.75	66.18	91.66	83.12	70.12	35.79	73.26	60.40	18.91	20.81	83.16	68.16	94.00	85.66	77.42	80.10	73.78	62.53

Table 10: Further analysis of results using different LLMs.

	Bank 77		Clinc150		Mtop		Massive		GoEmo		Average	
	Emb	LLM	Emb	LLM	Emb	LLM	Emb	LLM	Emb	LLM	Emb	LLM
Gemma	**62.03**	**75.55**	69.76	89.09	70.88	81.27	**61.58**	76.14	**22.33**	27.79	57.32	69.97
Qwen	61.08	75.32	**70.16**	**90.14**	**72.44**	**84.19**	60.88	**76.62**	22.27	**28.91**	**57.36**	**71.04**
Llama	56.15	64.57	64.05	78.95	67.99	75.16	58.17	67.74	19.96	23.43	53.26	61.97

Table 11: Selection Accuracy By LLMs at 1st iteration(%)(**emerging data scenario**). Bold numbers: best. All-MiniLM-L6-v2 used as backbone embedder.

Dataset	K	Total	LLM	No-LLM	LLM ratio
Bank77	77	13,083	3,080	10,003	23.5%
Clinc150	150	22,500	4,500	18,000	20%

Table 16: 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		Average	
	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc
TF-IDF												
Plain	53.06	33.34	56.55	32.64	39.79	22.83	41.26	26.09	11.56	16.10	40.44	26.20
LeBoT (Ours)	**79.92**	**65.23**	**89.29**	**79.58**	**71.14**	**46.62**	**69.81**	**56.46**	**30.93**	**24.47**	**68.22**	**54.47**
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
LeBoT (Ours)	**76.12**	**57.40**	**93.07**	**85.62**	**75.94**	**51.64**	**73.71**	**63.97**	**20.78**	**15.68**	**67.92**	**54.86**
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
LeBoT (Ours)	**85.13**	**71.85**	**94.52**	**87.58**	**76.83**	**59.51**	**76.88**	**67.78**	**23.18**	**21.92**	**71.31**	**61.73**
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
LeBoT (Ours)	**85.03**	**70.65**	**93.82**	**85.80**	**75.38**	53.17	**76.33**	**62.59**	**28.07**	**29.59**	**71.73**	**60.36**

Table 12: Five-benchmark results (**emerging data scenario, HDBSCAN**). Gemma is used in LeBoT. **Plain**: direct embedding clustering. Bold numbers: best within the embedder.

	Bank 77		Clinc150		StackO		Average	
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
LeBoT	**86.75**	**74.64**	**95.24**	**88.36**	**81.43**	**83.50**	**87.81**	**82.17**

Table 13: Three-benchmark results (**GCD scenario, HDBSCAN**) Gemma is used in LeBoT. Bold numbers: best. All-MiniLM-L6-v2 used as backbone embedder. **Plain**: direct embedding clustering.

	Bank 77	Clinc150	Mtop	Massive	GoEmo
Bert					
Plain	2	111	62	2	2
LeBoT	69	144	**77**	**61**	51
MiniLM					
Plain	**78**	139	17	27	11
LeBoT	66	145	67	56	53
E5					
Plain	73	134	9	2	**3**
LeBoT	68	147	64	63	60

Table 14: **Emerging** scenario: Estimated Number of Clusters $\hat{K}$ by HDBSCAN. **gemma-2-9b-it** used in LeBoT.

	Bank 77	Clinc150	Stackoverflow
Plain	**78**	82	13
LeBoT	80	**139**	**19**
True K	77	150	20

Table 15: **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 LeBoT.