LLM-MemCluster: Empowering Large Language Models with Dynamic Memory for Text Clustering

Yuanjie Zhu[1] **Liangwei Yang**[1*] **Ke Xu**[1] **Weizhi Zhang**[1]
Zihe Song[1] **Jindong Wang**[2] **Philip S. Yu**[1]
[1]University of Illinois Chicago
[2]William & Mary
{yzhu224, lyang84, kxu25, wzhan42, zsong29, psyu}@uic.edu
jwang80@wm.edu

Abstract

Large Language Models (LLMs) are reshaping unsupervised learning by offering an unprecedented ability to perform text clustering based on their deep semantic understanding. However, their direct application is fundamentally limited by a lack of stateful memory for iterative refinement and the difficulty of managing cluster granularity. As a result, existing methods often rely on complex pipelines with external modules, sacrificing a truly end-to-end approach. We introduce **LLM-MemCluster**, a novel framework that reconceptualizes clustering as a fully LLM-native task. It leverages a **Dynamic Memory** to instill state awareness and a **Dual-Prompt Strategy** to enable the model to reason about and determine the number of clusters. Evaluated on several benchmark datasets, our tuning-free framework significantly and consistently outperforms strong baselines. LLM-MemCluster presents an effective, interpretable, and truly end-to-end paradigm for LLM-based text clustering.

1 Introduction

Text clustering, a cornerstone task in Natural Language Processing (NLP), aims to automatically organize a collection of documents into meaningful groups based on content similarity. This unsupervised learning technique is pivotal for large-scale knowledge discovery and information organization, with its utility demonstrated in applications ranging from structuring massive document archives to analyzing the collective voice of online communities (Zhou et al., 2024; Hadifar et al., 2019). Traditional clustering methods, such as K-Means (Jin and Han, 2017; Sinaga and Yang, 2020) or hierarchical clustering (Sahoo et al., 2006; Ran et al., 2023), typically operate on vector-space representations like TF-IDF (Bafna et al., 2016) or, more recently, pre-trained text embeddings from benchmarks like MTEB (Muennighoff et al., 2023). While these approaches are effective, a notable limitation (Ezugwu et al., 2022) is their reliance on either handcrafted features or domain-specific fine-tuning to achieve optimal performance.

The advent of Large Language Models (LLMs) with powerful semantic understanding and reasoning capabilities, such as GPT-4, Gemini, and DeepSeek (Achiam et al., 2023; Team et al., 2023; Liu et al., 2024), has introduced a new paradigm for text clustering. Current research, however, has largely focused on hybrid frameworks that employ LLMs in auxiliary roles to enhance traditional embedding-based pipelines. These applications include enriching text representations (Wang et al., 2024), refining cluster assignments (Feng et al., 2024), and supervising the fine-tuning of external embedding models (Zhang et al., 2023). While innovative, these methods' reliance on external components precludes fully LLM-native clustering.

However, using LLMs as standalone clustering agents reveals two fundamental architectural challenges. The first is a direct conflict between operational requirements and model design: the limited context window necessitates processing large datasets in batches, yet the models' inherent statelessness prevents memory retention across these batches. This contradiction is a primary hurdle for achieving coherent and stable cluster assignments. This problem is further compounded by a second critical challenge: controlling clustering granularity. Without an explicit mechanism for guiding the partitioning process, LLMs tend to produce arbitrary and unstable topic partitions, as they lack an intrinsic method to determine a suitable degree of specificity. These limitations highlight the need for a framework that can impose statefulness while actively steering the clustering process.

To address these challenges, we introduce a novel framework for text clustering named **LLM-MemCluster**. This approach leverages large lan-

*Corresponding author.

guage models, requires no model fine-tuning or integration with traditional algorithms, and is driven by two key innovations—each specifically designed to address the aforementioned limitations.

1. **Dynamic Memory Mechanism:** We introduce a memory mechanism that maintains a dynamic set of cluster labels within the prompt. This evolving memory state transforms the LLM into a state-aware clustering agent that can iteratively assign documents to existing clusters, create new ones for distinct topics, and merge and refine the cluster labels to ensure global consistency.

2. **Granularity Control Mechanism:** To actively guide the LLM in determining a suitable number of clusters, we employ two distinct prompting modes. A strict prompt encourages the consolidation of the existing cluster memory into broader categories, whereas a relaxed prompt fosters the discovery of more fine-grained topics. This dual-mode strategy enables the framework to explore different levels of granularity, ultimately achieving a stable and well-justified cluster count.

Our comprehensive experiments on several public benchmark datasets demonstrate that LLM-MemCluster significantly outperforms both traditional embedding-based methods and existing LLM-enhanced baselines across multiple standard evaluation metrics. These findings validate our framework as an effective solution for text clustering, harnessing the potential of end-to-end LLMs.

In summary, our contributions are threefold:

- Dynamic Memory Mechanism overcoming inherent LLM statelessness and facilitating the iterative refinement of cluster quality.
- Granularity Control Mechanism employing a novel dual-prompt strategy to ensure precise, user-guided control of cluster granularity.
- State-of-the-art performance on multiple standard clustering benchmarks, demonstrating robust, fine-tuning-free generalization across a diverse spectrum of both proprietary and open-source large language models.

2 Method

2.1 Problem Formulation

Text clustering aims to automatically organize a collection of documents into meaningful groups based on content similarity. Formally, given an unlabeled text corpus, $\mathcal{D} = \{x_1, x_2, \ldots, x_N\}$, the objective is to derive a partition of the corpus, $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K\}$. This partition consists of K clusters, where each cluster $\mathcal{C}_k$ is a subset of the original corpus $\mathcal{D}$, formally defined as:

$$\mathcal{C}_k = \{x_j \in \mathcal{D} \mid l_j = k\}$$

Here, l_j represents the label assigned to instance x_j. The final partition covers all instances, with mutually exclusive clusters. The number of clusters, K, is dynamically determined, satisfying $1 \leq K \leq N$.

2.2 Framework Overview

We propose **LLM-MemCluster**, a novel framework that leverages API calls to a large language model (LLM), eliminating the need for model fine-tuning or integration with traditional algorithms. As illustrated in Figure 1, our framework is designed to directly address two principal challenges: the statelessness of LLMs and the inherent ambiguity in determining the number of clusters.

The architecture of LLM-MemCluster is centered on two synergistic innovations: a **Dynamic Memory** mechanism that endows the LLM with a functional state, and a **Dual-Prompt Strategy** for active control over clustering granularity. The framework processes each text instance from $\mathcal{D}$ sequentially. Throughout this process, it maintains a dynamic set of assignments $\mathcal{A} = \{(x_j, l_j)\}_{j=1}^{N}$, recording the label l_j for each instance x_j. This set $\mathcal{A}$ is crucial for the iterative refinement and is used to produce the final partition $\mathcal{C}$.

2.3 Stateful Clustering via Dynamic Memory

The inherent statelessness of contemporary LLMs, confining their operational memory to a single context window, presents a significant challenge for iterative tasks. In clustering, this leads to inconsistent assignments and redundant clusters. Our Dynamic Memory mechanism addresses this by providing the LLM with a persistent working memory of the evolving cluster landscape. Our framework operates in a single pass, efficiently completing the clustering of N instances in exactly N steps, unlike iterative methods like K-Means.

The memory module, denoted as $\mathcal{M}_{\text{mem}}$, maintains a dynamically updated set of descriptive labels representing the discovered clusters (e.g., "Arts", "Science"). At each step i (for $i = 1, \ldots, N$), the framework processes instance x_i.

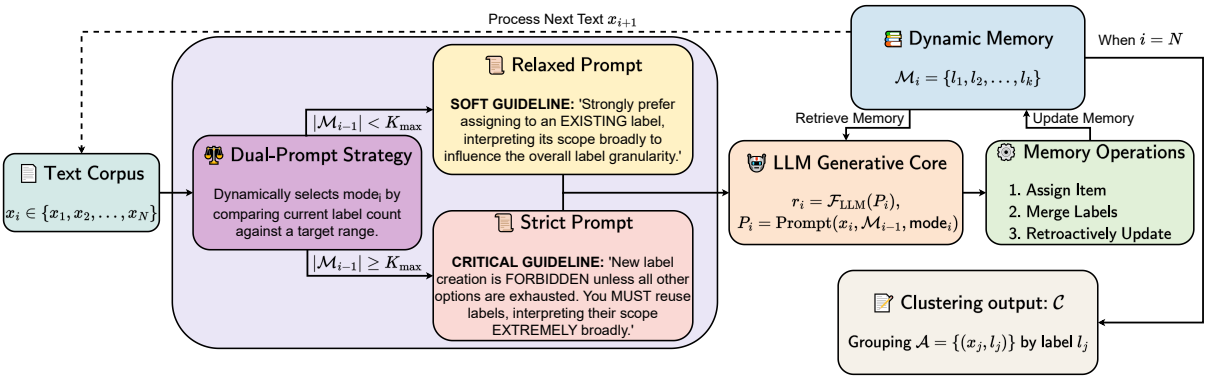


Figure 1: An overview of our proposed LLM-MemCluster framework. This figure illustrates the core iterative process, which is driven by a Dynamic Memory mechanism and the Dual-Prompt Strategy.

Let $\mathcal{M}_{i-1}$ be the memory state before this step. The core operation is to invoke the LLM, modeled as a function $\mathcal{F}_{\text{LLM}}$, with a prompt constructed from the current instance x_i, the memory state $\mathcal{M}_{i-1}$, and the active mode (detailed in Section 2.4). Conceptually, the LLM returns a structured tuple containing an assignment label l_i and an optional merge suggestion s_i. A merge suggestion has the form $(\mathcal{L}_{\text{old}}, l_{\text{new}})$; for example, $s_i = (\{\text{“ML”}, \text{“DL”}\}, \text{“AI”})$ proposes consolidating existing labels. Formally:

$$(l_i, s_i) = \mathcal{F}_{\text{LLM}}(x_i, \mathcal{M}_{i-1}, \text{mode}_i). \quad (1)$$

The framework uses the returned label l_i and merge suggestion s_i to update its state, cycling through three core operations:

- **Reuse or Create.** For each text instance x_i, our framework constructs a prompt containing the current set of labels from $\mathcal{M}_{i-1}$. The LLM's primary directive is to either reuse an existing label for x_i or create a new label if the text represents a fundamentally distinct topic, yielding the intermediate memory state $\mathcal{M}'_i$:

$$\mathcal{M}'_i = \begin{cases} \mathcal{M}_{i-1} \cup \{l_i\} & \text{if } l_i \text{ is a new label} \\ \mathcal{M}_{i-1} & \text{otherwise} \end{cases} \quad (2)$$

- **Merge and Refine.** A defining feature of our framework is its capacity to direct the LLM to propose a `MERGE_SUGGESTION` at any step, enabling proactive consolidation of semantically similar or redundant labels. Crucially, this is an optional, concurrent action rather than a post-processing phase, allowing real-time optimization of the label space. The memory state $\mathcal{M}_i$ is updated based on the merge suggestion $s_i = (\mathcal{L}_{\text{old}}, l_{\text{new}})$:

$$\mathcal{M}_i = \begin{cases} (\mathcal{M}'_i \setminus \mathcal{L}_{\text{old}}) \cup \{l_{\text{new}}\} & \text{on merge} \\ \mathcal{M}'_i & \text{otherwise} \end{cases} \quad (3)$$

- **Retroactive Update.** Upon receiving a merge suggestion, the framework updates $\mathcal{M}_{\text{mem}}$ and retroactively re-maps historical assignments. This ensures global consistency (detailed in Appendix F). Specifically, any assignment $(x_j, l_j) \in \mathcal{A}$ is updated to (x_j, l'_j), where:

$$l'_j = \begin{cases} l_{\text{new}} & \text{if } l_j \in \mathcal{L}_{\text{old}} \\ l_j & \text{otherwise} \end{cases} \quad (4)$$

This integrated cycle transforms the stateless LLM into a state-aware clustering agent, ensuring both local accuracy and global consistency. The process is driven by a structured prompt (see Appendix A), which instructs the LLM to return a primary assignment—either reusing or creating a label—and, optionally, a merge suggestion.

2.4 Dual-Prompt Granularity Control

We introduce the Dual-Prompt Strategy to provide users with a means of actively guiding the final clustering granularity. This approach addresses the canonical challenge of steering the final number of clusters (K) to align with user-defined goals, and is implemented as a dedicated control layer that actively regulates the cluster count. By doing so, the strategy ensures the final partition conforms to user expectations or the data's intrinsic structure.

This strategy modulates the LLM's propensity for new label creation by dynamically switching between two prompting modes. The mechanism is

guided by a user-defined target range for the cluster count, $[K_{\min}, K_{\max}]$. While the entire range is provided to the LLM as a contextual guideline for its decision-making, the programmatic switch between modes is triggered by the upper bound, $K_{\max}$. Specifically, the prompt mode for instance x_i is determined by the current cluster count:

$$\text{mode}_i = \begin{cases} \text{Strict} & \text{if } |\mathcal{M}_{i-1}| \geq K_{\max} \\ \text{Relaxed} & \text{otherwise} \end{cases} \quad (5)$$

This strategy uses two distinct prompt templates:

1. **The Strict Prompt:** Activated when the current number of clusters meets or exceeds the desired maximum, this mode incorporates prescriptive constraints into the prompt, significantly curtailing new label creation and compelling the LLM to prioritize **Reuse** and **Merge**. This raises the threshold for creating new clusters, hindering their formation.

2. **The Relaxed Prompt:** As the default operational mode, this prompt is used when the cluster count is within the desired range. It grants the LLM greater latitude in label creation, allowing it to form new clusters for semantically distinct topics as needed, thereby facilitating cluster discovery.

By adjusting prompt constraints based on the real-time cluster count, this strategy provides explicit control over the final clustering granularity, preventing uncontrolled label growth or premature consolidation. The complete strict and relaxed prompt templates are detailed in Appendix A. Finally, we provide the detailed algorithmic pseudocode and a computational complexity analysis in Appendix F.

3 Experiments

In this section, we evaluate our proposed framework, **LLM-MemCluster**, through experiments addressing the following research questions:

- **RQ1:** How does LLM-MemCluster perform against a variety of strong clustering baselines that employ different algorithms and state-of-the-art text representations?

- **RQ2:** How do the Dynamic Memory and Dual-Prompt Strategy components individually contribute to the overall effectiveness of our proposed LLM-MemCluster?

- **RQ3:** How robust is the LLM-MemCluster framework to variations in experimental conditions, specifically the dual-prompt transition threshold and the dataset execution order?

- **RQ4:** What is the generalization capability of the LLM-MemCluster framework when its foundational component is substituted with different large language models?

3.1 Experimental Setup

3.1.1 Datasets

We evaluate our method on six public benchmark datasets (Zhang et al., 2023), selected to cover a wide range of text clustering challenges. As detailed in Appendix B, these datasets span numerous domains and feature a broad range of cluster counts (K from 18 to 102), providing a robust testbed to assess the generalization of our method.

3.1.2 Evaluation Metrics

We evaluate performance using three standard metrics, where higher values indicate better performance and 1 denotes a perfect score:

- **Accuracy (ACC):** Calculates the percentage of correctly assigned data points, based on the optimal one-to-one mapping between predicted clusters and ground-truth labels.

- **Normalized Mutual Information (NMI):** Measures the mutual information between predicted and true labels, normalized by their entropies. It quantifies the statistical information shared between the two assignments.

- **Adjusted Rand Index (ARI):** A chance-adjusted measure of similarity between two data clusterings. It is calculated based on the proportion of sample pairs that are correctly assigned to the same or different clusters.

3.1.3 Baselines

We assess effectiveness by benchmarking against baselines from three distinct paradigms:

- **Traditional Method:** K-Means on TF-IDF vectors, a classic baseline relying on sparse, high-dimensional lexical features.

- **Embedding-based Methods:** We evaluate algorithms representing three key approaches: the centroid-based K-Means (Lloyd, 1982), the density-based DBSCAN (Deng, 2020),

and the graph-based Spectral Clustering (Ng et al., 2001). We apply these methods to `instructor-large` embeddings (Su et al., 2023) alongside the established BERTopic pipeline (Grootendorst, 2022).

- **LLM-based Method:** We compare against ClusterLLM (Zhang et al., 2023), which uses an LLM to generate pseudo-labels for training a smaller sentence encoder, enabling a highly scalable, multi-stage clustering approach. To ensure reproducibility, we set the temperature to 0 for all LLM-based experiments.

3.2 Main Results (RQ1)

As shown in Table 1, our framework, LLM-MemCluster, establishes a new state-of-the-art in unsupervised text clustering. On average, LLM-MemCluster surpasses the strongest baseline, ClusterLLM, by absolute margins of 9.5% in ACC, 4.4% in NMI, and 17.3% in ARI. The framework's advantages are particularly evident on high-cardinality datasets where conventional methods tend to falter. For instance, on MTOP-I (K=102), it achieves an ARI of 68.9—a 38.9-point improvement over ClusterLLM. A similar 42.4-point gain in ARI on FewNerd (K=58) further demonstrates its effectiveness for semantically complex tasks.

These results offer a crucial insight: superior clustering performance is not merely a function of powerful text representations, but rather a result of an architectural design that effectively leverages these representations. This architectural dependence highlights the fundamental limitations of baseline methods. Embedding-based approaches, such as Spectral Clustering, rely on static vectors that, despite their quality, lack contextual adaptability. Other LLM-based methods like ClusterLLM treat the LLM as an external guide for knowledge distillation, rather than as a dynamic agent within the clustering process. This claim is further substantiated by our comprehensive generalization experiments in RQ4 (Section 3.5).

In contrast, the success of LLM-MemCluster is rooted in its novel architecture, which engages the LLM as a direct and active agent within a stateful, iterative process. The dynamic memory mechanism enables the framework to build a coherent, evolving understanding of the cluster space. This, in turn, allows the LLM to make adaptive, context-aware decisions at each step. We argue this direct and dynamic orchestration of the LLM's decision-making is the key innovation, allowing our method to navigate nuanced semantic relationships for more robust and accurate clustering.

3.3 Ablation Study (RQ2)

We conduct a comprehensive study to validate the contributions of our framework's modules. The primary findings are summarized in Figure 2 and analyzed in detail in the subsequent subsections. Full numerical breakdowns for all experimental variants are provided in Appendix C.

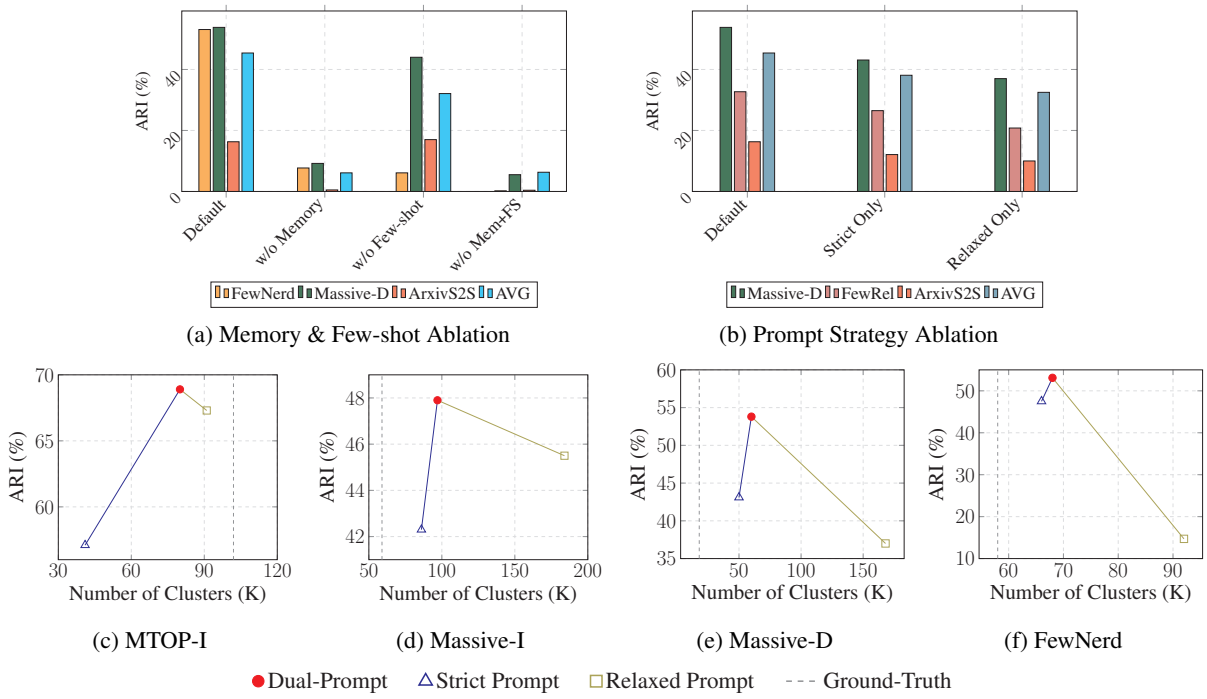
Memory and Grounding are Indispensable. Figure 2a starkly highlights the critical roles of memory and in-context examples. Deactivating the Dynamic Memory (*w/o Memory*) causes a catastrophic performance degradation across all datasets, validating that an external memory is essential for overcoming LLM statelessness. The importance of grounding the model with few-shot examples is also evident, though its impact varies. Removing them (*w/o Few-shot*) generally leads to a significant ARI drop, a trend mirrored in Massive-D (from 53.8 to 44.0). However, the effect is exceptionally pronounced on FewNerd, where the ARI collapses from 53.1 to a mere 6.1. The stark performance drop on FewNerd underscores that for semantically complex domains, few-shot grounding is a prerequisite for robust performance.

The Dual-Prompt Strategy is Highly Effective. As shown in Figure 2b, the superiority of our dual-prompt approach is evident, as variants relying on a single prompt type consistently underperform the full model. This pattern is not only clear on average—where the full model achieves a 45.4 ARI, compared to 38.1 for the strict-only and 32.5 for the relaxed-only variants—but is also robustly replicated across individual datasets. For instance, on Massive-D the full model's 53.8 ARI significantly exceeds the alternatives (43.1 and 37.0); a similar trend is observed on FewRel (32.7 vs. 26.5 and 20.8). This consistent underperformance validates our core design principle: a dynamic transition from an exploratory to a consolidative phase is the most effective strategy for reliably achieving optimal clustering granularity.

To further analyze how optimal granularity is achieved, Figures 2c to 2f illustrate the framework's adaptive behavior on representative datasets. On Massive-I, the framework achieves a higher ARI score via **semantic splitting**, producing more clusters than the ground-truth by identifying fine-grained sub-topics. Conversely, on the high-

Method	ArxivS2S			Massive-I			MTOP-I			Massive-D			FewNerd			FewRel			AVG		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
K-Means-TF-IDF	12.1	31.7	1.2	31.1	49.8	8.4	31.7	55.3	16.0	43.9	44.2	10.5	11.1	37.7	1.0	23.1	36.1	4.2	25.5	42.5	6.9
DBSCAN	6.3	17.6	0.3	20.4	28.9	1.0	21.5	26.6	2.2	25.9	30.7	6.4	27.0	0.5	0.1	10.5	19.5	0.4	18.6	20.6	1.7
Spectral	25.1	48.0	9.2	**60.5**	72.9	38.7	39.2	68.8	27.8	54.1	64.7	33.0	34.0	42.0	9.3	35.4	51.5	15.7	41.4	58.0	22.3
BERTopic	17.9	39.2	1.8	52.5	70.0	32.5	35.8	64.1	15.9	52.6	56.2	29.3	34.9	40.3	11.7	31.1	50.7	9.7	37.5	53.4	16.8
K-Means-Inst	25.1	49.3	12.3	55.7	72.6	41.6	34.5	70.9	26.9	54.9	66.9	42.7	28.2	43.3	6.1	34.8	53.1	22.5	38.9	59.3	25.4
ClusterLLM	25.1	50.5	13.7	55.5	**74.6**	43.2	36.0	73.4	30.0	52.4	65.3	40.8	37.3	53.1	10.7	**43.8**	59.6	30.4	41.7	62.8	28.1
Our Method	**28.4**	**57.4**	**16.3**	54.8	73.5	**47.9**	**64.0**	**77.5**	**68.9**	**57.6**	**67.7**	**53.8**	**59.3**	**63.3**	**53.1**	43.2	**63.6**	**32.7**	**51.2**	**67.2**	**45.4**

Table 1: Comparison of LLM-MemCluster with baselines across six datasets using ACC, NMI, and ARI scores (%). The best and second-best results are highlighted in **bold** and underlined, respectively. Baselines utilize `instructor-large` embeddings (except K-Means-TF-IDF), while our method employs in-context learning and ClusterLLM uses it to provide guidance (both utilizing the GPT-4.1 mini model).



(a) Memory & Few-shot Ablation (b) Prompt Strategy Ablation

(c) MTOP-I (d) Massive-I (e) Massive-D (f) FewNerd

Figure 2: Comprehensive ablation study and adaptive clustering strategy comparison.

cardinality MTOP-I dataset, it performs **semantic consolidation**, merging overly similar categories to produce fewer clusters. Crucially, in both scenarios, the Dual-Prompt strategy yields the solution with the highest ARI score. This demonstrates that the framework does not rigidly pursue a specific K but rather optimizes for semantic coherence, adaptively deciding whether to split or merge, a determination strictly contingent on the intrinsic semantic properties of each dataset.

3.4 Robustness Analysis (RQ3)

To address RQ3, we evaluate the robustness of LLM-MemCluster against variations in experimental conditions, specifically focusing on the hyperparameter sensitivity and the stability under randomized dataset execution orders.

Hyperparameter Sensitivity We first analyze the sensitivity to the core hyperparameter: the transition threshold for the Dual-Prompt Strategy. This threshold determines when the model switches from the initial, exploratory relaxed prompt to the subsequent, consolidative strict prompt. We operationalize this threshold as an offset applied to the upper bound of the target range (K_{max}). A positive offset extends the exploratory phase, while a negative offset accelerates the consolidation process.

We evaluated a broad spectrum of offsets: -10, 0, +10, +50, +100, and +200. The results are visualized in Figure 3, which plots performance against different threshold offsets, with detailed results in Appendix D.1. While the average performance across all datasets (Figure 3c) shows rela-

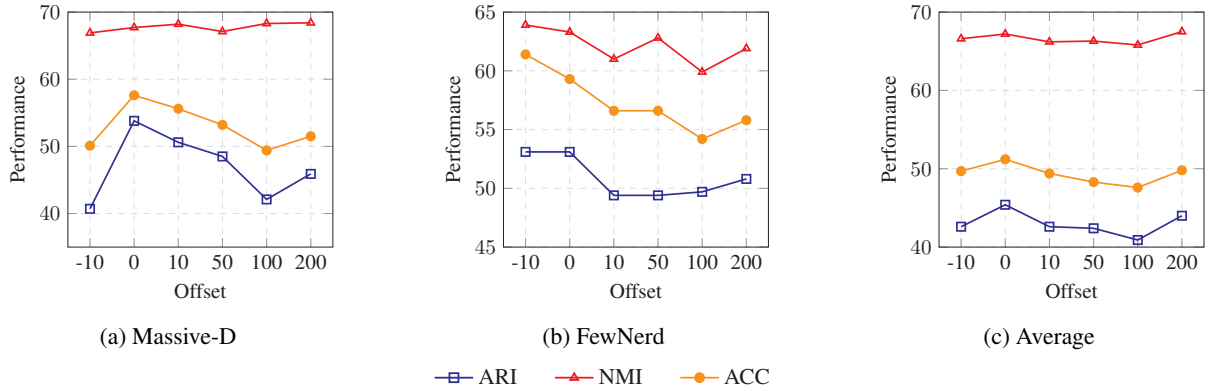


Figure 3: Hyperparameter sensitivity analysis of the prompt transition threshold, demonstrating robust and near-optimal performance across a wide range of values for representative datasets and on average.

tively flat curves, this stability is even more evident on individual datasets. For instance, on FewNerd (Figure 3b), the ARI score is exceptionally stable, fluctuating only minimally between a peak of 53.1 (default offset) and a low of 49.4. Even on Massive-D (Figure 3a), which exhibits more variance, performance peaks at an offset of 0 (53.8 ARI) and remains competitive across a wide range.

Notably, the framework performs well even at the extremes. An extended exploratory phase (offset +200) yields a strong ARI of 44.0 and the highest average NMI of 67.5. Conversely, an accelerated transition (offset -10) also maintains a robust 42.6 ARI. This resilience at the boundaries, mirrored across representative datasets, highlights the inherent robustness and self-correcting capacity of the dual-prompt mechanism, which adapts effectively to minor variations in the consolidation timing and control process.

Seed Robustness Beyond hyperparameter settings, we also examine the stability of our method under different dataset execution orders. This is a critical factor for stream clustering algorithms, where the processing sequence can inherently influence the resulting cluster integrity. We conducted experiments using 5 random seeds to simulate permuted input streams and report the mean and standard deviation in Appendix Table 7. The results demonstrate that LLM-MemCluster maintains consistent performance regardless of the input order. The standard deviation of the ARI scores remains low across all datasets, specifically falling within a tight 5% margin. For example, on Massive-I and MTOP-I, the standard deviations are merely 2.85% and 3.75%, respectively. This low variance statistically confirms that our dynamic memory and adaptive dual-prompt mechanisms work in concert to resiliently capture global semantic structures independent of the local input sequence.

Consequently, our framework provides a practical advantage by delivering near-optimal, reproducible results across diverse datasets without requiring dataset-specific tuning or seed selection.

3.5 Generalization to Different LLMs (RQ4)

In addressing RQ4, we assess our framework's generalization by evaluating it across a range of Large Language Models, including GPT-4.1-mini (default), GPT-3.5-turbo, GPT-4.1, Gemini-2.0-flash, Gemini-2.5-flash-preview-05-20, and DeepSeek-V3-0324, thereby confirming its portability.

The results, presented in Table 2, highlight the framework's strong portability and robustness. Performance remains exceptionally strong when other high-capability models are used. For instance, substituting our default GPT-4.1-mini (45.4 ARI) with the more powerful GPT-4.1 yields a nearly identical ARI of 45.3. Similarly, competitive performance is observed with Gemini-2.5-flash-preview-05-20 (44.5 ARI). The strong results from other models, including DeepSeek-V3-0324 (38.7 ARI), confirm our design's successful generalization across a diverse set of LLM backbones.

The advantages of our design are most apparent when paired with less capable models. For instance, our framework achieves a 34.6 ARI using Gemini-2.0-flash, significantly surpassing the 28.1 ARI of ClusterLLM, which uses the more capable GPT-4.1-mini (Tables 2 and 1). This comparison strongly indicates that our innovative design approach, rather than the underlying model's intrinsic capability, primarily accounts for the gains.

Base LLM	ArxivS2S			Massive-I			MTOP-I			Massive-D			FewNerd			FewRel			AVG		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
GPT-4.1-M	28.4	57.4	16.3	54.8	73.5	47.9	64.0	77.5	68.9	57.6	67.7	53.8	59.3	63.3	53.1	43.2	63.6	32.7	51.2	67.2	45.4
GPT-3.5-T	35.9	59.5	19.8	43.5	70.0	38.2	52.6	75.0	54.0	55.0	65.4	45.1	42.0	62.9	24.8	25.2	49.9	14.6	42.4	63.8	32.8
GPT-4.1	29.8	60.4	18.6	64.0	77.3	54.6	67.8	80.9	69.3	52.8	69.5	47.3	59.1	73.4	51.1	48.4	69.4	30.8	53.6	71.8	45.3
Gemini-2.0-F	33.8	60.2	21.3	29.1	51.7	11.1	63.2	74.7	63.4	50.0	63.2	35.9	46.3	57.4	52.3	37.1	59.1	23.9	43.3	61.1	34.6
DeepSeek-V3	23.9	54.1	14.9	48.1	69.1	39.3	60.9	73.7	63.2	53.7	59.7	39.7	53.6	62.0	62.3	20.9	46.5	12.7	43.5	60.9	38.7
Gemini-2.5-F	34.8	68.4	24.9	50.6	77.3	41.6	60.3	80.4	60.5	54.5	67.4	43.8	65.7	76.4	59.4	48.3	70.4	36.9	52.4	73.4	44.5

Table 2: Generalization of the framework across large language models in ACC, NMI, and ARI (%). For brevity, we abbreviate model names: GPT-4.1-M (GPT-4.1-mini), GPT-3.5-T (GPT-3.5-turbo), Gemini-2.0-F (Gemini-2.0-flash), DeepSeek-V3 (DeepSeek-V3-0324), and Gemini-2.5-F (Gemini-2.5-flash-preview-05-20).

Beyond performance, we provide a detailed analysis of the monetary cost in Appendix E. A key efficiency advantage of our design is that token consumption is predominantly driven by the input context, whereas the output completion tokens remain minimal. Given that completion tokens often incur higher rates, this characteristic ensures that monetary costs remain manageable and predictable, even when scaling to premium models.

4 Related Work

4.1 LLM-Augmented Clustering

A prominent approach employs a Large Language Model (LLM) as a high-level "oracle" to augment or refine clustering pipelines that rely on external models. These methods distill the LLM's semantic judgment to address specific, challenging parts of the clustering process. For instance, Cequel (Wang et al., 2025) generates pairwise constraints (Basu et al., 2004) to guide a downstream clustering algorithm. Other work focuses on refinement, where LLMEdgeRefine (Feng et al., 2024) re-assigns ambiguous "edge points" on the boundaries of initial clusters to enhance their integrity. A third approach, ClusterLLM (Zhang et al., 2023), leverages an LLM to generate supervisory signals from confusing document triplets to effectively fine-tune a smaller, more efficient sentence encoder. While pragmatic, these "LLM-as-oracle" frameworks are hybrid solutions and do not constitute an end-to-end generative clustering process.

4.2 End-to-End Generative Clustering

A more recent paradigm shift leverages LLMs as standalone clustering agents, bypassing traditional numerical algorithms. A representative approach within this paradigm reframes clustering as a classification problem. The T-CLC framework (Huang and He, 2025), for example, operates in two distinct stages where it first prompts an LLM to generate candidate labels. However, this approach relies on a subset of ground-truth labels to seed the generation process, effectively shifting the task paradigm towards semi-supervised learning. In contrast, our work addresses the more challenging, fully unsupervised setting where no prior knowledge of the label space is available. Other related works have focused on improving the prompting process. ZeroDL (Jo et al., 2025), for instance, first performs an open-ended inference step to learn the dataset's underlying distribution and then incorporates this meta-knowledge into a more data-aware prompt. However, this approach still treats clustering as a static inference task rather than a dynamic process with a continuously evolving state.

Our work, LLM-MemCluster, builds upon a generative paradigm but introduces a novel framework designed for efficient, iterative, single-pass clustering. In contrast to the multi-stage or static-inference approaches, it employs a **Dynamic Memory** to create a stateful process that continuously refines the cluster space (Xu et al., 2025; Liu et al., 2025). Furthermore, our **Dual-Prompt Strategy** provides an explicit mechanism for active granularity control throughout the process, addressing the challenge of dynamically determining the cluster count (Petnehazi and Aradi, 2025).

5 Conclusion

We introduce LLM-MemCluster, an end-to-end text clustering framework using a Dynamic Memory and Dual-Prompt Strategy to operate as a stateful, iterative agent, addressing the core challenges of LLM statelessness and cluster granularity. Its robust architecture achieves state-of-the-art performance, establishing an LLM-native paradigm that advances beyond hybrid approaches to effectively unlock LLM potential in unsupervised tasks.

Limitations

Our framework entails specific trade-offs. First, it relies on the instruction-following capabilities of the LLM; while our approach is effective on advanced models, performance may vary on smaller architectures unable to strictly adhere to complex dual-prompt constraints. Second, as is inherent to its single-pass streaming nature, it optimizes based on the evolving memory state rather than performing multi-pass global refinement over the entire dataset. Finally, although the user-defined target range offers flexible granularity control, it necessitates minimal domain knowledge regarding the expected cluster distribution.

Acknowledgments

This work is supported in part by NSF under grant III-2106758. This work used the Delta system at the National Center for Supercomputing Applications (award OAC 2005572) through allocation CIS251166 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Prafulla Bafna, Dhanya Pramod, and Anagha Vaidya. 2016. Document clustering: Tf-idf approach. In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pages 61–66.

Sugato Basu, Arindam Banerjee, and Raymond J Mooney. 2004. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the 2004 SIAM international conference on data mining*, pages 333–344. SIAM.

Dingsheng Deng. 2020. Dbscan clustering algorithm based on density. In *2020 7th international forum on electrical engineering and automation (IFEEA)*, pages 949–953. IEEE.

Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. 2021. Few-NERD: A few-shot named entity recognition dataset. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3198–3213, Online. Association for Computational Linguistics.

Absalom E Ezugwu, Abiodun M Ikotun, Olaide O Oyelade, Laith Abualigah, Jeffery O Agushaka, Christopher I Eke, and Andronicus A Akinyelu. 2022. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, 110:104743.

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, Miami, Florida, USA. Association for Computational Linguistics.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and Prem Natarajan. 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, Toronto, Canada. Association for Computational Linguistics.

Tianyu Gao, Xu Han, Hao Zhu, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2019. FewRel 2.0: Towards more challenging few-shot relation classification. 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 6250–6255, Hong Kong, China. Association for Computational Linguistics.

Maarten Grootendorst. 2022. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *arXiv preprint arXiv:2203.05794*.

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

Chen Huang and Guoxiu He. 2025. Text clustering as classification with llms. In *Proceedings of the 2025 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, SIGIR-AP 2025, page 374–384, New York, NY, USA. Association for Computing Machinery.

Xin Jin and Jiawei Han. 2017. K-means clustering. In *Encyclopedia of machine learning and data mining*, pages 695–697. Springer.

Hwiyeol Jo, Hyunwoo Lee, Kang Min Yoo, and Taiwoo Park. 2025. ZeroDL: Zero-shot distribution learning for text clustering via large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19597–19607, Vienna, Austria. Association for Computational Linguistics.

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

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Guangda Liu, Chengwei Li, Jieru Zhao, Chenqi Zhang, and Minyi Guo. 2025. Clusterkv: Manipulating llm kv cache in semantic space for recallable compression. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*, pages 1–7.

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

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 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, Dubrovnik, Croatia. Association for Computational Linguistics.

Andrew Ng, Michael Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14.

Gabor Petnehazi and Bernadett Aradi. 2025. Hercules: Hierarchical embedding-based recursive clustering using llms for efficient summarization. *arXiv preprint arXiv:2506.19992*.

Xingcheng Ran, Yue Xi, Yonggang Lu, Xiangwen Wang, and Zhenyu Lu. 2023. Comprehensive survey on hierarchical clustering algorithms and the recent developments. *Artificial Intelligence Review*, 56(8):8219–8264.

Nachiketa Sahoo, Jamie Callan, Ramayya Krishnan, George Duncan, and Rema Padman. 2006. Incremental hierarchical clustering of text documents. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 357–366.

Kristina P Sinaga and Miin-Shen Yang. 2020. Unsupervised k-means clustering algorithm. *IEEE access*, 8:80716–80727.

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

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Hongtao Wang, Taiyan Zhang, Renchi Yang, and Jianliang Xu. 2025. Cequel: Cost-effective querying of large language models for text clustering. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, CIKM '25, page 2998–3008, New York, NY, USA. Association for Computing Machinery.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916, Bangkok, Thailand. Association for Computational Linguistics.

Wujiang Xu, Kai Mei, Hang Gao, Juntao Tan, Zujie Liang, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*.

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, Singapore. Association for Computational Linguistics.

Sheng Zhou, Hongjia Xu, Zhuonan Zheng, Jiawei Chen, Zhao Li, Jiajun Bu, Jia Wu, Xin Wang, Wenwu Zhu, and Martin Ester. 2024. A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions. *ACM Comput. Surv.*, 57(3).

A Unified Prompt Template

This section details the unified prompt template at the core of our framework. As shown in Figure 4, the template integrates our **Dynamic Memory** by injecting the current `Known labels`, and our **Dual-Prompt Strategy** via placeholders for dynamic instructions. The specific content for the `[SYSTEM_GUIDELINE]` and `[USER_CONSTRAINT]` placeholders is provided in Figures 6 and 7.

```
--- SYSTEM PROMPT ---
You are an expert text analysis and clustering specialist.
Your primary goal is to determine the underlying theme, topic,
or relation type for each text input and assign it to an
appropriate category.

CORE PRINCIPLES:
- HIGHEST PRIORITY: Reuse existing labels whenever reasonably
  possible to ensure consistency.
- NEW LABELS: Create ONLY AS A LAST RESORT when an input is
  FUNDAMENTALLY NEW.
- MERGE: Suggest merging similar labels to improve conciseness.

[SYSTEM_GUIDELINE]
```

Figure 4: The unified prompt template (system prompt).

```
--- USER PROMPT ---
Known labels: ["label_1", "label_2", ...]

Examples:
Input: "Example text 1" -> Output: ASSIGNED_LABEL: "label_A"
Input: "Example text 2" -> Output: NEW_LABEL: "label_B"

Input to process: "text_to_cluster"

Instructions:
Your response must contain exactly one of the following primary lines:
- ASSIGNED_LABEL: <label_name>
- NEW_LABEL: <new_label_name> [USER_CONSTRAINT]

Optionally, you can also include the following line for consolidation:
- MERGE_SUGGESTION: MERGE: ["old_label"] INTO: ["new_label"]

RESPONSE FORMATTING:
- Exactly ONE 'ASSIGNED_LABEL:' OR 'NEW_LABEL:' line.
- Optionally, ONE 'MERGE_SUGGESTION:' line.
```

Figure 5: The unified prompt template (user prompt, with `Known labels` from dynamic memory).

```
[SYSTEM_GUIDELINE] Content
--------------------------
# Relaxed Mode (Default)
SOFT GUIDELINE: As an additional consideration, try to manage the overall list of
    known labels such that the total number of unique labels ideally stays {
    range_desc}. This is a soft guideline to influence label granularity; your
    primary decision-making process (prioritize reuse, create new only if essential,
     suggest useful merges) remains paramount.

# Strict Mode
CRITICAL GUIDELINE: The total number of unique labels MUST be managed towards {
    range_desc}. If approaching/exceeding the upper limit, new label creation is
    SEVERELY RESTRICTED. You MUST aggressively reuse existing labels (interpret
    their scope VERY broadly) and proactively seek merge opportunities.
```

Figure 6: Content for placeholder [SYSTEM_GUIDELINE]. Injected into Figure 4 based on the mode.

A.1 Dynamic Placeholder Content

Our framework modulates the prompt's behavior by programmatically switching between two operational modes. Specifically, we dynamically populate two placeholders: `[SYSTEM_GUIDELINE]` (system-level guidance) and `[USER_CONSTRAINT]` (user-specific constraints), as defined in Figure 4. The actual content injected into these placeholders consists of the detailed instructions and constraints shown in Figures 6 and 7. The transition between

```
[USER_CONSTRAINT] Content
-------------------------
# Relaxed Mode
CONSIDERATION: If current known labels approach or exceed {target_max_clusters},
    please be very cautious about creating NEW_LABEL. Strongly prefer assigning to
    an EXISTING label (interpret its scope broadly) or identifying a MERGE.

# Strict Mode
CRITICAL CHECK: If current known labels approach or exceed {target_max_clusters},
    creating a NEW_LABEL is FORBIDDEN unless all other options are exhausted. You
    MUST first attempt to assign to an EXISTING label (interpret its scope EXTREMELY
     broadly) or identify a MERGE. Only if the input is unequivocally unique and NO
    existing label can accommodate it even with the broadest interpretation, and NO
    merge is possible, then, as a final resort, create a NEW_LABEL.
```

Figure 7: Content for placeholder [USER_CONSTRAINT]. Injected into Figure 4 based on the mode.

operational modes is governed by the number of discovered clusters relative to a user-defined upper bound, $K_{\max}$. The system operates in its **Relaxed Mode**, employing soft advisory language, as long as the cluster count remains below this threshold. Once $K_{\max}$ is reached or exceeded, the system transitions to **Strict Mode**, using restrictive language to enforce the cluster cardinality.

B Dataset Overview

Table 3 lists the datasets used in our experiments, detailing each one's primary task or domain, the total number of samples, and the number of ground-truth clusters, denoted by K. We follow the experimental setup of Zhang et al. (2023) and use their processed versions of these datasets. The original sources are as follows: ArxivS2S is adapted from the MTEB benchmark (Muennighoff et al., 2023), consisting of scientific abstracts; Massive-I and Massive-D are subsets of the MASSIVE dataset (FitzGerald et al., 2023), focusing on intent detection and domain classification, respectively; MTOP-I is derived from the MTOP benchmark (Li et al., 2021), specifically focusing on intent classification; FewNerd is a large-scale, fine-grained named entity recognition dataset (Ding et al., 2021); and FewRel is a benchmark dataset for few-shot relation classification (Gao et al., 2019).

Dataset	Primary Task/Domain	# Samples	K
ArxivS2S	Scientific Abstracts	3,674	93
Massive-I	Intent Detection	2,974	59
MTOP-I	Intent Detection	4,386	102
Massive-D	Conversational Domain	2,974	18
FewNerd	Named Entity Recognition	3,789	58
FewRel	Relation Extraction	4,480	64

Table 3: Statistics of the datasets used in our experiments. K denotes the number of ground-truth clusters.

C Detailed Ablation Study

C.1 Analysis of Component Effectiveness

Tables 4 and 5 provide the detailed quantitative analysis comprehensively supplementing the ablation study discussion in Section 3.3.

The Roles of Memory and Grounding Our results in Table 4 clearly validate that both Dynamic Memory and few-shot grounding are critical for the framework's overall success.

- **Dynamic Memory (`w/o Memory`):** Deactivating the memory module leads to a near-total collapse in performance across all six datasets. The average ARI consequently plummets from **45.4%** to a mere **6.1%**. This confirms that an external, stateful memory is essential to overcome the inherent statelessness of LLMs for iterative tasks like clustering.
- **Few-shot Grounding (`w/o Few-shot`):** Removing the few-shot examples also causes a significant performance degradation, with the average ARI dropping from **45.4%** to **32.1%**. The effect is particularly dramatic on semantically nuanced datasets like FewNerd, where the ARI score collapses from **53.1%** to just **6.1%**. This highlights that for complex domains, providing in-context examples is crucial for effectively guiding the model to produce accurate and consistent outputs.

Effectiveness of the Dual-Prompt Strategy By conducting a cross-referenced analysis of the performance metrics in Table 4 and the generated clusters from Table 5, we can see how the Dual-Prompt

Method Variant	ArxivS2S			Massive-I			MTOP-I			Massive-D			FewNerd			FewRel			AVG		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
Default	**28.4**	**57.4**	**16.3**	**54.8**	**73.5**	**47.9**	**64.0**	**77.5**	**68.9**	**57.6**	**67.7**	**53.8**	**59.3**	**63.3**	**53.1**	**43.2**	**63.6**	**32.7**	**51.2**	**67.2**	**45.4**
w/o Memory	4.7	71.1	0.5	8.5	65.2	1.8	11.2	61.7	2.8	17.0	57.7	9.2	15.1	61.4	7.7	20.9	69.8	14.8	12.9	64.5	6.1
w/o Few-shot	28.1	58.5	17.0	50.2	70.9	43.5	61.4	74.3	60.5	55.5	65.7	44.0	27.3	48.7	6.1	33.5	55.3	21.2	42.7	62.2	32.1
w/o M+FS	4.2	71.0	0.4	18.5	67.4	11.9	23.1	65.2	18.5	11.7	55.4	5.5	4.3	57.2	0.2	5.3	66.2	1.2	11.2	63.7	6.3
Strict Prompt	18.6	49.0	12.1	52.6	69.7	42.3	56.8	74.8	57.1	50.2	67.6	43.1	57.8	62.6	47.5	38.1	64.1	26.5	45.7	64.6	38.1
Relaxed Prompt	17.5	60.6	10.0	54.7	74.3	45.5	65.4	78.3	67.3	41.7	65.4	37.0	39.4	52.1	14.7	33.7	56.6	20.8	42.1	64.5	32.5

Table 4: Ablation study of LLM-MemCluster supporting the analysis for RQ2. We report ACC, NMI, and ARI (%), highlighting the importance of each component by comparing performance to the Default setting.

Method Variant	ArxivS2S	Massive-I	MTOP-I	Massive-D	FewNerd	FewRel
Ground-Truth	**93**	**59**	**102**	**18**	**58**	**64**
Dual-Prompt	159	97	80	60	68	122
Strict Prompt	70	86	41	50	66	89
Relaxed Prompt	1208	184	91	168	92	141

Table 5: Comparison of the number of clusters (K) produced by different model variants.

strategy is superior to single-prompt variants.

- **`Relaxed Prompt` Variant:** This variant consistently generates a vastly larger number of clusters than the ground-truth (e.g., **1208** vs. 93 on ArxivS2S; **184** vs. 59 on Massive-I). This tendency to over-split the data results in poor semantic grouping and leads to the lowest average ARI of **32.5%**.
- **`Strict Prompt` Variant:** In contrast, the variant is overly conservative, often producing a cluster count that is suboptimal for that dataset (e.g., only **41** clusters for MTOP-I, where the ground-truth is 102). While this consolidation can be beneficial, it often merges distinct topics, capping its average ARI at **38.1%**.
- **`Dual-Prompt` Strategy:** The Dual-Prompt demonstrates a powerful adaptive capability. It navigates the trade-off between over-splitting and over-consolidating, producing a cluster count (e.g., **97** on Massive-I, **80** on MTOP-I) that better reflects the underlying data structure. This adaptive, dynamic control over granularity is the key reason it achieves the state-of-the-art average ARI of **45.4%**, outperforming both single-prompt baselines by a significant margin.

D Detailed Robustness Analysis

D.1 Hyperparameter Sensitivity

Table 6 details the full numerical results supporting the robustness claims in Section 3.4.

Overall Performance Stability The average performance across all datasets demonstrates remarkable stability. The average ARI remains high across a wide spectrum of offsets, from an accelerated transition (offset -10, ARI **42.6%**) to a significantly extended exploratory phase (offset +200, ARI **44.0%**). The peak performance is achieved at the default offset of 0 (ARI **45.4%**), but even extreme variations do not lead to a collapse in performance, underscoring the inherent self-correcting nature of the Dual-Prompt strategy.

Dataset-Specific Robustness The framework's demonstrated robustness is not merely a statistical artifact of averaging; rather, it is consistently evident at the individual dataset level.

- On FewNerd, a semantically complex dataset, the ARI score proves to be exceptionally stable. It peaks at a high of **53.1%** (offsets 0 and -10), while its lowest point remains a robust **49.4%** (offset +10 and +50). This narrow range of fluctuation powerfully highlights the model's ability to achieve consistent clustering results, regardless of potential minor timing adjustments in the consolidation phase.
- On Massive-D, which exhibits more variance, performance still remains competitive. While the peak ARI of **53.8%** is at the default offset, even an early transition (offset -10) yields a respectable ARI of **40.7%**, and a late transition (offset +200) maintains an ARI of **45.9%**.
- Notably, on some datasets like Massive-I, strategically shifting to an earlier transition (offset -10, ARI **52.4%**) or a later one (offset +200, ARI **52.3%**) can even outperform the default setting (ARI **47.9%**), suggesting that while the default is a strong general-purpose choice, the framework is robust enough to accommodate diverse data distributions.

Offset	ArxivS2S			Massive-I			MTOP-I			Massive-D			FewNerd			FewRel			AVG		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
0	**28.4**	**57.4**	**16.3**	**54.8**	**73.5**	**47.9**	**64.0**	**77.5**	**68.9**	**57.6**	**67.7**	**53.8**	**59.3**	**63.3**	**53.1**	**43.2**	**63.6**	**32.7**	**51.2**	**67.2**	**45.4**
-10	27.4	57.0	15.9	58.2	72.9	52.4	59.7	75.5	63.4	50.1	66.9	40.7	61.4	63.9	53.1	41.6	63.7	30.5	49.7	66.6	42.6
+10	28.0	58.6	16.6	53.8	72.6	45.9	62.7	76.2	65.9	55.6	68.2	50.6	56.6	61.0	49.4	39.6	60.8	27.3	49.4	66.2	42.6
+50	25.0	56.7	14.2	55.0	74.6	48.8	61.2	76.1	66.2	53.2	67.1	48.5	56.6	62.8	49.4	39.1	60.8	27.0	48.3	66.3	42.4
+100	25.6	57.3	14.3	53.0	72.2	44.4	62.8	75.7	65.7	49.4	68.3	42.1	54.2	59.9	49.7	40.7	61.4	29.4	47.6	65.8	40.9
+200	27.7	61.5	16.7	57.9	74.4	52.3	63.7	76.9	68.2	51.5	68.4	45.9	55.8	61.9	50.8	42.0	62.2	30.5	49.8	67.5	44.0

Table 6: Hyperparameter analysis of the dual-prompt transition threshold. We report ACC, NMI, and ARI (%) across various switching offsets, with the default setting (offset 0) included for comparison.

Dataset	ACC	NMI	ARI
ArxivS2S	30.13±3.81	53.52±2.09	15.82±2.20
Massive-I	53.89±2.36	72.50±2.17	45.69±2.85
MTOP-I	64.46±2.28	79.50±0.40	67.14±3.75
Massive-D	63.16±2.77	69.48±1.72	53.38±4.05
FewNerd	64.04±2.47	69.99±2.30	61.86±2.06
FewRel	30.81±4.52	54.92±5.06	19.34±4.17

Table 7: Robustness results using Gemini-2.5-Flash-Lite. Values represent mean ± std (%) computed over 5 random permutations of the dataset.

Dataset	Input Tokens	Output Tokens	Total Cost (USD)
ArxivS2S	4,495,870	32,969	0.46
Massive-I	3,590,409	25,027	0.37
MTOP-I	4,537,051	33,653	0.47
Massive-D	3,231,711	31,226	0.34
FewNerd	4,821,283	33,393	0.50
FewRel	5,451,649	42,872	0.56
Total	**26,127,973**	**199,140**	**2.70**

Table 8: Token consumption and estimated cost for running LLM-MemCluster with Gemini-2.5-Flash-Lite on each dataset. Input token price is $0.10 per 1M tokens and output token price is $0.40 per 1M tokens.

Table 6 confirms that LLM-MemCluster is not dependent on precise hyperparameter tuning. Its strong performance across a wide range of offsets enables near-optimal results on diverse datasets without laborious optimization.

D.2 Seed Robustness Under Randomized Dataset Execution Order

Table 7 presents the stability of our framework under randomized dataset execution orders. We report the mean and standard deviation of ACC, NMI, and ARI across five random seeds using Gemini-2.5-Flash-Lite. These detailed statistics support the findings in Section 3.4, confirming that the framework achieves highly consistent performance independent of the specific input sequence.

E Token Consumption and Cost

Table 8 details the token consumption and estimated monetary cost incurred by our framework using Gemini-2.5-Flash-Lite. We report the aggregated input and output tokens for each dataset, alongside the total cost derived from public pricing at the time of experimentation ($0.10 per 1M input tokens and $0.40 per 1M output tokens).

While tokenization and pricing vary across models, a consistent structural feature of our framework is that operational costs are predominantly driven by inexpensive input tokens, whereas the volume of costly output tokens remains minimal.

F Algorithmic Implementation and Computational Complexity

The procedural implementation of our framework is detailed across two algorithms. Algorithm 1 describes the core, single-step clustering operation, which encapsulates the Dynamic Memory mechanism. Algorithm 2 then presents the main workflow of LLM-MemCluster, illustrating how the core operation and the Dual-Prompt Granularity Control are integrated to process the entire dataset.

LLM-MemCluster processes a corpus of N instances in a single, deterministic pass. For each instance, the primary computational costs stem from the LLM API call, C_{LLM}, and a potential retroactive update. A retroactive update incurs a cost of $O(i)$ at step i. While the theoretical worst-case complexity is $O(N^2)$, empirical evidence consistently shows that merge events are remarkably rare (averaging fewer than 2 per dataset in our experiments with Gemini-2.5-Flash-Lite). Consequently, the effective complexity remains near-linear, $O(N \cdot (C_{LLM} + C_{\text{update}}))$, avoiding the non-deterministic convergence behavior inher-

Algorithm 1 Core Clustering Operation

Input: Text instance x_i; Memory of labels $\mathcal{M}_{\text{mem}}$; Assignments $\mathcal{A}$; Prompting $mode$ (Relaxed/Strict)
Output: Updated $(\mathcal{M}_{\text{mem}}, \mathcal{A})$

$\mathcal{L}_{\text{seen}} \leftarrow \mathcal{M}_{\text{mem}}$
$(l_i, s_i) \leftarrow \mathcal{F}_{\text{LLM}}(x_i, \mathcal{L}_{\text{seen}}, mode)$ Eq. (1)
$\mathcal{A} \leftarrow \mathcal{A} \cup \{(x_i, l_i)\}$
if $l_i \notin \mathcal{L}_{\text{seen}}$ **then**
  Add l_i to memory $\mathcal{M}_{\text{mem}}$ Eq. (2)
end if
if s_i is not null **then**
  $\mathcal{L}_{old}, l_{new} \leftarrow$ Extract labels from s_i
  $\mathcal{M}_{\text{mem}} \leftarrow (\mathcal{M}_{\text{mem}} \backslash \mathcal{L}_{old}) \cup \{l_{new}\}$ Eq. (3)
  for each $(x_j, l_j) \in \mathcal{A}$ **do**
    if $l_j \in \mathcal{L}_{old}$ **then**
      $l_j \leftarrow l_{new}$ Eq. (4)
    end if
  end for
end if

Algorithm 2 LLM-MemCluster Workflow

Input: Unlabeled text corpus $\mathcal{D} = \{x_1, \ldots, x_N\}$; LLM $\mathcal{F}_{\text{LLM}}$; Target K range $[K_{\min}, K_{\max}]$
Output: A partition of the corpus, $\mathcal{C}$.

Initialize memory $\mathcal{M}_{\text{mem}} \leftarrow \emptyset$ and assignments $\mathcal{A} \leftarrow \emptyset$
CoreOp updates $\mathcal{M}_{mem}$ and $\mathcal{A}$ in-place.
for each text instance x_i in $\mathcal{D}$ **do**
  if $|\mathcal{M}_{\text{mem}}| \geq K_{\max}$ **then**
    $mode \leftarrow$ Strict
  else
    $mode \leftarrow$ Relaxed
  end if
  CoreOp$(x_i, \mathcal{M}_{\text{mem}}, \mathcal{A}, \text{mode})$
end for
Generate the final partition $\mathcal{C}$ by grouping all instances in $\mathcal{A}$ by their assigned label
return Final partition $\mathcal{C}$

ent to iterative algorithms like K-Means.

In contrast to contemporary methods like ClusterLLM, which employs a multi-stage pipeline to fine-tune a separate encoder, our framework is a unified, single-pass procedure. This design avoids the costly overhead of intermediate model training and multiple algorithmic phases.