

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

Yuanjie Zhu¹ **Liangwei Yang¹** **Ke Xu¹** **Weizhi Zhang¹**
Zihe Song¹ **Jindong Wang²** **Philip S. Yu¹**

¹University of Illinois Chicago

²William & 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., 2022).

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., 2024a), 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 a 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 language models, requires no model fine-tuning or in-

tegration 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 full potential of end-to-end LLMs.

In summary, our contributions are threefold:

- Dynamic Memory Mechanism that enables LLMs to overcome their inherent statelessness and facilitates the iterative refinement of final cluster quality.
- Granularity Control Mechanism employing a novel dual-prompt strategy to achieve stable, precise, and user-guided control over the final number of clusters.
- 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, $\mathcal{D} = \{x_1, x_2, \dots, x_N\}$, the objective is to derive a partition of the corpus, $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_K\}$. 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 must cover all instances, and clusters must be mutually exclusive. The number of clusters, K , is determined dynamically during the process, adhering to the natural constraint $1 \leq K \leq N$.

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 $\mathcal{A} = \{(x_j, l_j)\}_{j=1}^N$, which records the assigned label l_j for each processed instance x_j . This set of assignments \mathcal{A} is crucial for the iterative refinement process and is used at the conclusion 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

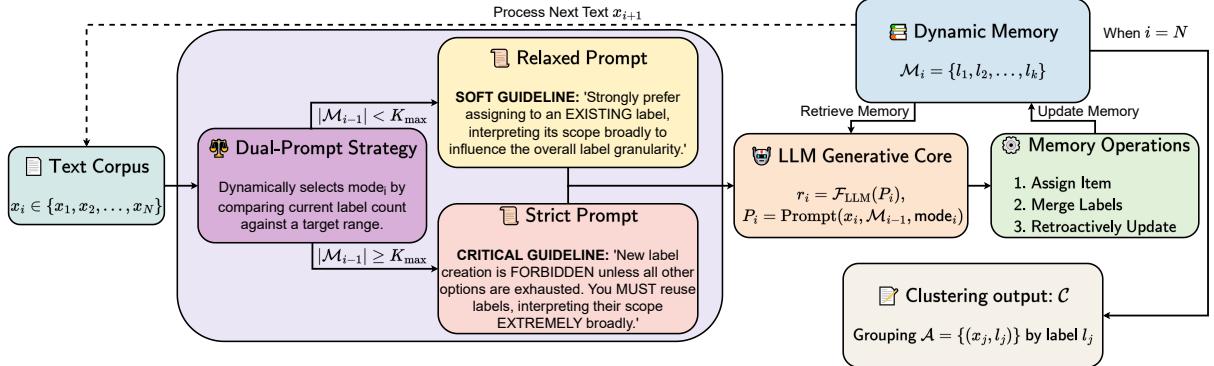


Figure 1: An overview of our proposed LLM-MemCluster framework. This figure illustrates the core iterative process, which is driven by a Dynamic Memory mechanism and the Dual-Prompt Strategy.

operates in a single-pass, completing the clustering of N instances in exactly N steps, unlike iterative methods like K-Means.

The memory module, denoted as \mathcal{M}_{mem} , maintains a dynamically updated set of descriptive labels representing the discovered clusters (e.g., ‘‘Arts’’, ‘‘Science’’). At each step i (for $i = 1, \dots, N$), the framework processes instance x_i . 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}_{LLM} , with a prompt P_i 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. This function is formalized as:

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

The framework then uses the returned label l_i and merge suggestion s_i to update its state, progressing through a continuous cycle of 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 (see Appendix A for prompt details), enabling proactive consolidation of semantically similar or redundant labels. Crucially, this is not a post-processing phase but an optional, concurrent action that allows real-time optimization of the label space. The memory state \mathcal{M}_i is then 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 not only updates the memory module \mathcal{M}_{mem} by replacing the antecedent labels with the new, consolidated one, but also performs a retroactive update across all historical assignments. This procedure ensures that any instance previously assigned to a deprecated label is remapped, thereby guaranteeing global consistency across the entire dataset (Algorithm 1). Specifically, for any assignment $(x_j, l_j) \in \mathcal{A}$ where j is the index of a previously processed instance, this update transforms it into a new

one, (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} . The prompt mode for P_i depends on 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 introducing new clusters, making their formation more difficult.
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 the discovery and identification of clusters.

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.

2.5 Algorithmic Implementation and 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, achieving a predictable linear time complexity. For each instance, the primary computational costs stem from the LLM API call, denoted as C_{LLM} , and a potential retroactive update. A retroactive update, which occurs only upon a merge suggestion, requires traversing previously processed assignments, incurring a cost of $O(i)$ at step i . However, since merge events are infrequent, the amortized update cost is low. The total complexity is therefore $O(N \cdot (C_{LLM} + C_{\text{update}}))$, where C_{update} is the amortized update cost. Crucially, this single-pass, predictable complexity avoids the non-deterministic, multi-pass nature of iterative algorithms like K-Means, which depend on an uncertain number of iterations to converge.

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.

3 Experiments

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

Algorithm 1 Core Clustering Operation

Input: A text instance x_i , a memory of labels \mathcal{M}_{mem} , a set of assignments \mathcal{A} , a mode $mode$

Output: Updated \mathcal{M}_{mem} , updated \mathcal{A}

```
1:  $\mathcal{L}_{\text{seen}} \leftarrow \mathcal{M}_{\text{mem}}$ 
2:  $P_{\text{assign}} \leftarrow$  Construct prompt using  $x_i$ ,  $\mathcal{L}_{\text{seen}}$ , and  $mode$ 
3:  $(l_i, s_i) \leftarrow \mathcal{F}_{\text{LLM}}(P_{\text{assign}})$  // Eq. (1)
4:  $\mathcal{A} \leftarrow \mathcal{A} \cup \{(x_i, l_i)\}$ 
5: if  $l_i \notin \mathcal{L}_{\text{seen}}$  then
6:     Add  $l_i$  to memory  $\mathcal{M}_{\text{mem}}$  // Eq. (2)
7: end if
8: if  $s_i$  is not null then
9:      $\mathcal{L}_{\text{old}}, l_{\text{new}} \leftarrow$  Extract labels from  $s_i$ 
10:     $\mathcal{M}_{\text{mem}} \leftarrow (\mathcal{M}_{\text{mem}} \setminus \mathcal{L}_{\text{old}}) \cup \{l_{\text{new}}\}$  // Eq. (3)
11:    for each  $(x_j, l_j) \in \mathcal{A}$  do
12:       if  $l_j \in \mathcal{L}_{\text{old}}$  then
13:            $l_j \leftarrow l_{\text{new}}$  // Eq. (4)
14:       end if
15:    end for
16: end if
17: return updated  $\mathcal{M}_{\text{mem}}$ ,  $\mathcal{A}$ 
```

- **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:** What are the individual contributions of Dynamic Memory and the Dual-Prompt Strategy to LLM-MemCluster’s effectiveness?
- **RQ3:** How robust is the performance of our proposed LLM-MemCluster framework to variations in the dual-prompt transition threshold hyperparameter?
- **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

Algorithm 2 The Workflow of LLM-MemCluster

Input: Unlabeled text corpus $\mathcal{D} = \{x_1, \dots, x_N\}$; LLM \mathcal{F}_{LLM} ; Target K range $[K_{\min}, K_{\max}]$

Output: A partition of the corpus, \mathcal{C} .

```
1: Initialize memory  $\mathcal{M}_{\text{mem}} \leftarrow \emptyset$  and assignments  $\mathcal{A} \leftarrow \emptyset$ 
2: for each text instance  $x_i$  in  $\mathcal{D}$  do
3:     if  $|\mathcal{M}_{\text{mem}}| \geq K_{\max}$  then
4:          $mode \leftarrow$  Strict
5:     else
6:          $mode \leftarrow$  Relaxed
7:     end if
8:      $(\mathcal{M}_{\text{mem}}, \mathcal{A}) \leftarrow \text{CoreOp}(x_i, \mathcal{M}_{\text{mem}}, \mathcal{A}, mode)$ 
9: end for
10: Generate the final partition  $\mathcal{C}$  by grouping all instances in  $\mathcal{A}$  by their assigned label
11: return Final partition  $\mathcal{C}$ 
```

assess the generalization and effectiveness of our proposed 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 best 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

To evaluate its effectiveness, we benchmark our framework against baselines from three distinct paradigms:

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

- **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., 2022) and the BERTopic pipeline (Grootendorst, 2022).
- **LLM-based Method:** We compare against ClusterLLM, a recent baseline method (Zhang et al., 2023) that uses an LLM to generate pseudo-labels for training a smaller sentence encoder, enabling a highly scalable, multi-stage clustering approach.

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 11.5% in ACC, 5.3% in NMI, and 20.8% 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 clustering 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. Our comprehensive generalization experiments in RQ4 (Section 3.5), which control for model capability, will substantiate this claim.

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 nuanced datasets, such grounding examples are indispensable.

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 54.5 ARI, compared to 45.7 for the strict-only and 39.0 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 achieving optimal clustering granularity.

To further analyze how optimal granularity is achieved, Figures 2c to 2f illustrate the frame-

| Method | ArxivS2S | | | Massive-I | | | MTOP-I | | | Massive-D | | | FewNerd | | | FewRel | | | AVG | | |
|-------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| | 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 | 30.6 | 51.0 | 8.2 |
| 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 | 22.3 | 24.7 | 2.1 |
| 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 | 49.7 | 69.6 | 26.7 |
| 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 | <u>11.7</u> | 31.1 | 50.7 | 9.7 | 44.9 | 64.1 | 20.2 |
| 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 | <u>42.7</u> | 28.2 | 43.3 | 6.1 | 34.8 | 53.1 | 22.5 | 46.6 | 71.2 | 30.4 |
| 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 | 50.0 | 75.3 | 33.7 |
| 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 | 61.5 | 80.6 | 54.5 |

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

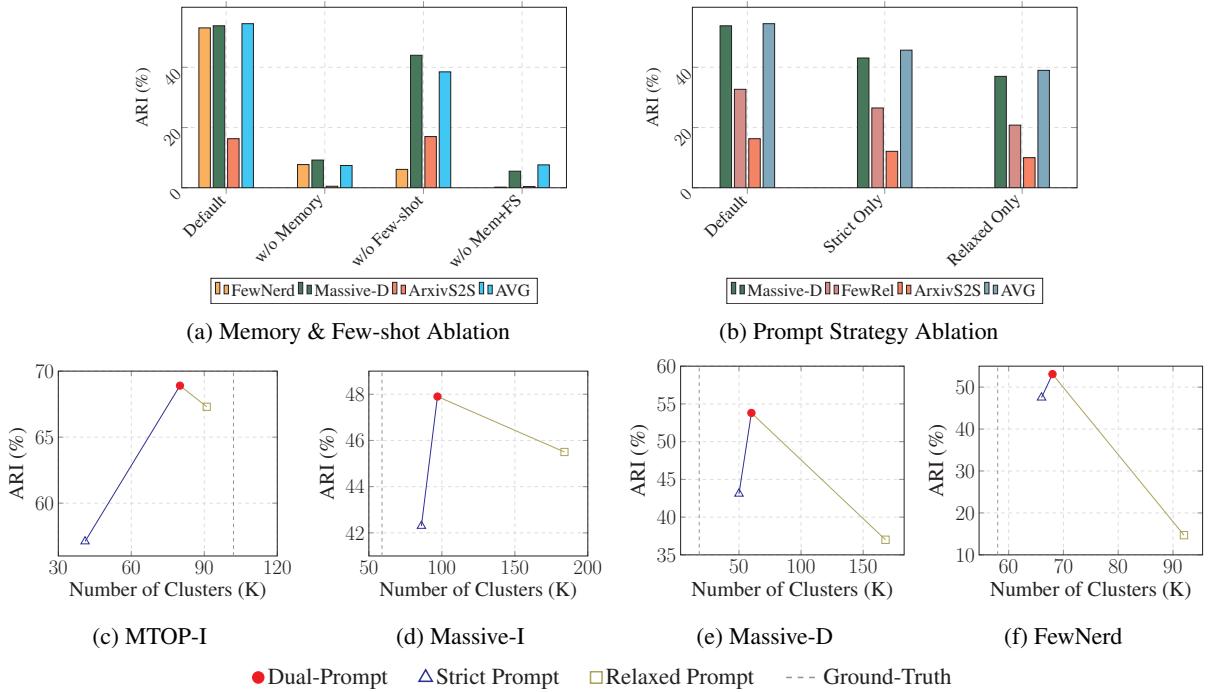


Figure 2: Comprehensive ablation study and adaptive clustering strategy comparison.

work'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-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 contingent on the intrinsic semantic properties of each dataset.

3.4 Hyperparameter Analysis (RQ3)

In order to answer RQ3, we analyze the sensitivity of LLM-MemCluster to its 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. While the average performance across

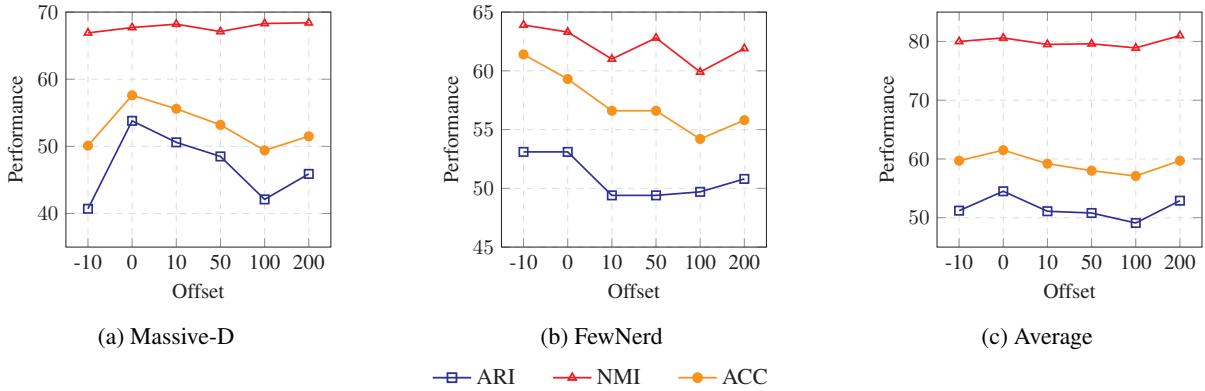


Figure 3: Hyperparameter sensitivity analysis of the prompt transition threshold, demonstrating robust and near-optimal performance across a wide range of values for representative datasets and on average.

all datasets (Figure 3c) shows relatively 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. This low sensitivity indicates that the model’s effectiveness is not critically dependent on precise hyperparameter tuning.

Notably, the framework performs well even at the extremes. An extended exploratory phase (offset +200) yields a strong ARI of 52.9 and the highest average NMI of 81.0. Conversely, an accelerated transition (offset -10) also maintains a robust 51.2 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 process.

In summary, the robustness of our Dual-Prompt Strategy, demonstrated by strong performance across a wide range of hyperparameter offsets, provides a key practical advantage: the ability to achieve near-optimal results on diverse datasets without laborious, dataset-specific tuning.

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 (54.5 ARI) with the more powerful GPT-4.1 yields a nearly identical ARI of 54.3. Similarly, competitive performance is observed with Gemini-2.5-flash-preview-05-20 (53.4 ARI). The strong results from other models, including DeepSeek-V3-0324 (46.4 ARI), confirm our design’s successful generalization across different LLMs.

The advantages of our design are most apparent when paired with less capable models. For instance, our framework achieves a 41.6 ARI using Gemini-2.0-flash, significantly surpassing the 33.7 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, is the primary driver of performance gains.

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, TECL (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 ini-

| Base LLM | ArxivS2S | Massive-I | MTOP-I | Massive-D | FewNerd | FewRel | Avg |
|------------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| | 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 | 61.5 80.6 54.5 |
| 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 | 50.8 76.5 39.3 |
| 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 | 64.4 86.2 54.3 |
| 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 | 51.9 73.3 41.6 |
| 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 | 52.2 73.0 46.4 |
| 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 | 62.8 88.1 53.4 |

Table 2: Framework generalization across different large language models, with performance measured 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).

tial clusters to enhance their integrity. A third approach, ClusterLLM (Zhang et al., 2023), leverages an LLM to generate supervisory signals from confusing document triplets (Diaz-Rodriguez, 2025) to 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 leading approach within this paradigm reframes clustering as a classification problem. The T-CLC framework (Huang and He, 2024), for example, operates in two distinct stages. First, it prompts an LLM with data samples to generate a set of human-readable topic labels—a process that often requires a separate step to merge and refine labels from different batches. In the second stage, it uses the finalized label set to classify each document. This two-stage design, while innovative, is not inherently iterative; the cluster structure is largely fixed after the first stage and lacks a mechanism for dynamic refinement as individual documents are processed. Other related works have focused on improving the prompting process. ZeroDL (Jo et al., 2024), 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 evolving state.

Our work, LLM-MemCluster, builds upon a generative paradigm but introduces a novel framework designed for 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 clus-

ter space (Xu et al., 2025; Liu et al., 2024b). 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 beyond hybrid approaches to unlock LLM potential in unsupervised tasks.

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.
- Jairo Diaz-Rodriguez. 2025. k-llmmeans: scalable, stable, and interpretable text clustering via llm-based centroids. *arXiv preprint arXiv:2502.09667*.

- Absalom E Ezugwu, Abiodun M Ikoronuno, Olaide O Oye-lade, 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.
- 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.
- Chen Huang and Guoxiu He. 2024. Text clustering as classification with llms. *arXiv preprint arXiv:2410.00927*.
- 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, and Taiwoo Park. 2024. Zerodl: Zero-shot distribution learning for text clustering via large language models. *arXiv preprint arXiv:2406.13342*.
- Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.
- Guangda Liu, Chengwei Li, Jieru Zhao, Chenqi Zhang, and Minyi Guo. 2024b. Clusterkv: Manipulating llm kv cache in semantic space for recallable compression. *arXiv preprint arXiv:2412.03213*.
- Stuart Lloyd. 1982. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.
- Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.
- 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. 2022. One embedder, any task: Instruction-finetuned text embeddings. *arXiv preprint arXiv:2212.09741*.
- 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. Cost-effective text clustering with large language models. *arXiv preprint arXiv:2504.15640*.
- 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.
- 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.

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 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 .

C Detailed Ablation Study

Table 4 presents the comprehensive and detailed results for the ablation study, which provides direct support for our analysis of RQ2. Additionally, Table 5 offers a side-by-side comparison of the total number of clusters produced by each variant, which explains the performance outcomes.

C.1 Analysis of Component Effectiveness (RQ2)

This section provides a detailed quantitative analysis for the ablation study (RQ2), leveraging the comprehensive results presented in Table 4 and 5 to supplement the findings discussed in the main paper.

The Roles of Memory and Grounding Our results in Table 4 validate that both Dynamic Memory and few-shot grounding are critical for the framework’s 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 **54.5%** to a mere **7.4%**. 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 **54.5%** to **38.5%**. 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 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 cluster counts from Table 5, we can clearly see how the Dual-Prompt 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 **39.0%**.
- **Strict Prompt Variant:** In contrast, the variant is overly conservative, producing fewer clusters than is optimal for most datasets (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 **45.7%**.
- **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 **54.5%**,

```

--- 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).

outperforming both single-prompt baselines by a significant margin.

D Detailed Hyperparameter Analysis

Table 6 presents the detailed results of the hyperparameter sensitivity analysis that supports RQ3.

D.1 Analysis of Framework Robustness (RQ3)

This section provides a detailed quantitative analysis of the hyperparameter sensitivity study, using the full results from Table 6 to substantiate the claims of robustness made in the main paper. The core hyperparameter, the transition threshold, is operationalized as an offset to the target K_{max} . A positive offset delays the switch to the Strict Prompt, while a negative offset accelerates it.

Overall Performance Stability The average performance across all datasets demonstrates remark-

able stability. The average ARI remains high across a wide spectrum of offsets, from an accelerated transition (offset -10, ARI **51.2%**) to a significantly extended exploratory phase (offset +200, ARI **52.9%**). The peak performance is achieved at the default offset of 0 (ARI **54.5%**), 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; it is 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

```

[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.

```

[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.

range of fluctuation powerfully highlights the model’s ability to achieve consistent results, regardless of 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 a diverse range of data distributions.

To sum up, the detailed results in Table 6 confirm that LLM-MemCluster is not critically dependent on precise hyperparameter tuning. Its strong per-

formance across a wide range of offsets provides a key practical advantage, enabling near-optimal results on diverse datasets without laborious, dataset-specific optimization.

| 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.

| Method Variant | ArxivS2S | | | Massive-I | | | MTOP-I | | | Massive-D | | | FewNerd | | | FewRel | | | AVG | | |
|----------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| | 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 | 61.5 | 80.6 | 54.5 |
| 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 | 15.5 | 77.4 | 7.4 |
| 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 | 51.2 | 74.7 | 38.5 |
| 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 | 13.4 | 76.5 | 7.6 |
| 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 | 54.8 | 77.6 | 45.7 |
| 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 | 50.5 | 77.4 | 39.0 |

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.

| Offset | ArxivS2S | | | Massive-I | | | MTOP-I | | | Massive-D | | | FewNerd | | | FewRel | | | AVG | | |
|----------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| | 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 | 61.5 | 80.6 | 54.5 |
| -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 | 59.7 | 80.0 | 51.2 |
| +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 | 59.2 | 79.5 | 51.1 |
| +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 | 58.0 | 79.6 | 50.8 |
| +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 | 57.1 | 78.9 | 49.1 |
| +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 | 59.7 | 81.0 | 52.9 |

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.