TestNUC: Enhancing Test-Time Computing Approaches and Scaling through Neighboring Unlabeled Data Consistency

Henry Peng Zou[1]*, **Zhengyao Gu**[1]*, **Yue Zhou**[1], **Yankai Chen**[2], **Weizhi Zhang**[1], **Liancheng Fang**[1], **Yibo Wang**[1], **Yangning Li**[3], **Kay Liu**[1], **Philip S. Yu**[1]
[1]University of Illinois Chicago, [2]Cornell University [3]Tsinghua University
{pzou3, zgu24}@uic.edu

Abstract

Test-time computing approaches, which leverage additional computational resources during inference, have been proven effective in enhancing large language model performance. This work introduces a novel, linearly scaling approach, TestNUC, that improves test-time predictions by leveraging the local consistency of neighboring unlabeled data-it classifies an input instance by considering not only the model's prediction on that instance but also on neighboring unlabeled instances. We evaluate TestNUC across eight diverse datasets, spanning intent classification, topic mining, domain discovery, and emotion detection, demonstrating its consistent superiority over baseline methods such as standard prompting and self-consistency. Furthermore, TestNUC can be seamlessly integrated with existing test-time computing approaches, substantially boosting their performance. Our analysis reveals that TestNUC scales effectively with increasing amounts of unlabeled data and performs robustly across different embedding models, making it practical for real-world applications. Our code is available at https://github.com/HenryPengZou/TestNUC.

1 Introduction

Test-time computing approaches, which leverage additional computational resources during inference to enhance performance, have gained increasing attention in the era of large language models (LLMs) (Snell et al., 2024; Dong et al., 2024). There are two primary strategies for modifying an LLM's distribution at test time: (1) **at the *input level***: augmenting the prompt with additional tokens (e.g., few-shot in-context learning (Mosbach et al., 2023)); or (2) **at the *output level***: sampling multiple candidate answers and aggregating them (e.g., self-consistency (Wang et al., 2023b), best-of-N (Beeching et al., 2024)). Despite demonstrating promising capabilities, input-augmentation approaches incur a computational cost that scales *quadratically* with the number of added tokens in the prompt, making them more computationally expensive than output-sampling methods. Meanwhile, output-sampling approaches typically overlook the potential of *large amounts of unlabeled data* that are often available in real-world settings (Berthelot et al., 2019; Sohn et al., 2020; Zou and Caragea, 2023; Zou et al., 2025; Gu et al., 2025).

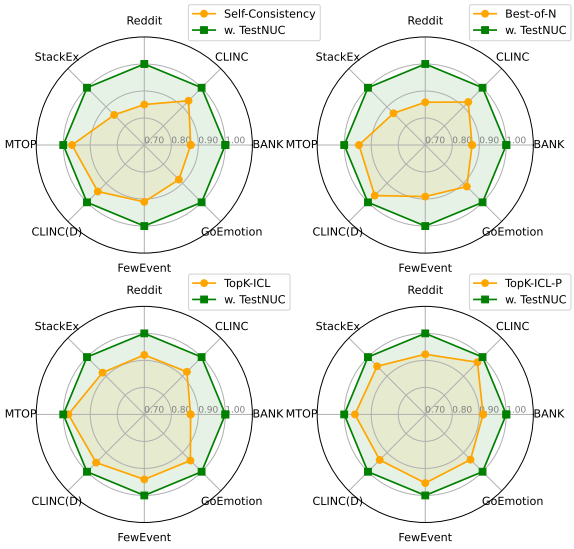

Figure 1: TestNUC effectively integrates with both *output-level* (e.g., Self-Consistency, Best-of-N) and *input-level* (e.g., ICL-based) test-time computing methods, consistently enhancing their performance across eight datasets. More details in Section 4 and Table 2.

To bridge these gaps, we present an initial exploration of how unlabeled data can be efficiently leveraged to enhance test-time computing approaches. We hypothesize that instances with similar embeddings are likely to share the same semantic label, which can provide unsupervised signals for improving inference consistency, particularly for challenging instances (Van Gansbeke

*Equal Contribution.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 30750–30762
July 27 - August 1, 2025

et al., 2020). Our pilot experiments across various benchmarks reveal strong semantic label consistency among neighboring instances, and we find that aggregating these neighborhood labels through simple aggregation methods such as majority voting leads to stable and accurate predictions (as shown in Figure 2, 3 in Section 2).

Motivated by these findings, we propose **TestNUC**, a simple yet effective approach that enhances test-time LLM predictions by leveraging neighboring unlabeled data consistency. Concretely, TestNUC consists of two key steps: ❶ **Neighbor Retrieval**, where we identify the top-K nearest unlabeled neighbors of a test sample based on feature similarity; and ❷ **Collaborative Prediction**, where the LLM generates predictions for both the test sample and its retrieved neighbors, which are then aggregated to obtain the final answer. The intuition behind TestNUC is that samples in close proximity within the embedding space are likely to share similar labels. By incorporating predictions of nearby unlabeled samples, the LLM can exploit the consistency of local data structures to better contextualize and refine its decision-making, effectively using unlabeled examples as an auxiliary signal to boost test-time performance while reducing noise and uncertainty (Van Gansbeke et al., 2020; Zhou et al., 2024; Wang et al., 2025).

We evaluate our approach across diverse tasks, including intent classification, topic mining, domain discovery, and emotion detection, using eight datasets that cover a wide spectrum of granularities, with class sizes ranging from 10 to 150. Our results demonstrate that TestNUC consistently outperforms baseline methods, such as standard prompting and self-consistency (Wang et al., 2023b), by a large margin across four large language models, showing its effectiveness in leveraging unlabeled data for test-time computation. Moreover, TestNUC can be seamlessly integrated with existing test-time computing approaches, such as TopK-ICL (Peng et al., 2024; Gao et al., 2024), best-of-N (Lightman et al., 2024; Beeching et al., 2024) and self-consistency (Wang et al., 2023b), significantly boosting their performances (as illustrated in Figure 1). In addition, TestNUC is effective across various embedders of different sizes and scales well with increasing amounts of unlabeled data (as shown in Figure 5), making it applicable to real-world scenarios.

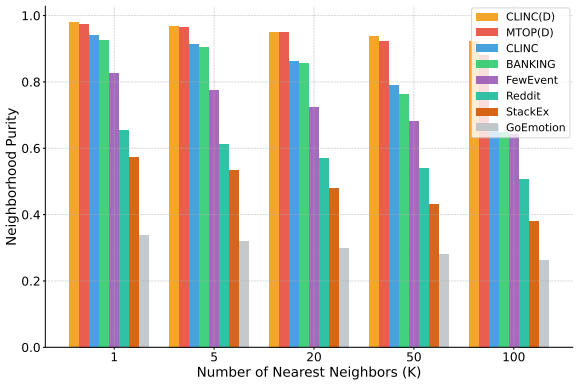

Figure 2: Neighboring samples tend to be instances of the same semantic class.

2 Preliminary Analysis

Leveraging neighboring examples at inference time has been shown to improve the generalization of language models (Khandelwal et al., 2020), mitigate prompting bias (Xu et al., 2023), and improve retrieval-augmented generation (Shi et al., 2022). Building on these findings, we explore a more focused question: To what extent can semantically similar neighborhood data serve as effective prediction proxies and potentially enhance LLM predictions at test time?

To understand this, we introduce *neighborhood purity*, which measures how often semantically similar examples share the same label. Formally, let $\mathcal{D} = (x_i, y_i)_{i=1}^{N}$ be a set of inputs and corresponding ground truth labels, where N is the total number of data points. We denote the K-nearest *neighborhood* of an input x as $\mathcal{N} = \text{argtop}_K\{\mathcal{S}_f(x, x_i) \mid i = 0, \ldots, N\}$, representing the set of indices corresponding to the most similar instances according to an embedding function f. We refer to x as the anchor of the neighborhood and measure the consistency of its neighborhood with *purity* ϕ, defined as:

$$\phi(\mathcal{N}) = \frac{1}{KN} \sum_{i=1}^{N} \sum_{j \in \mathcal{N}} \mathbf{1}(y_i = y_j) \qquad (1)$$

Intuitively, purity measures the proportion of instances that share the same label as the anchor.

We conduct our preliminary experiments across eight datasets spanning class granularities from 10 to 150. Detailed dataset descriptions and statistics are provided in Section 4.1 and Table 5. As shown in Figure 2, nearest neighbors frequently belong to the same semantic class as the anchor. In the worst

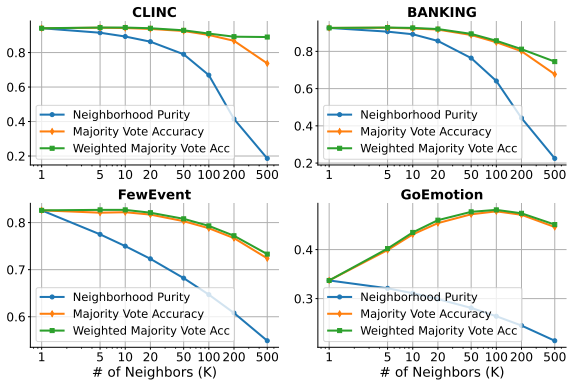

Figure 3: Majority vote over neighborhood ground-truth labels leads to stable and accurate predictions. Incorporating feature similarity-based weighting further improves stability for large K values by mitigating noise.

case, purity still reaches around 0.3 when $K = 20$ on the GoEmotion dataset.

Then, we ascertain how accurately the aggregation over neighboring ground-truth labels predicts the anchor's label. To this end, we consider two aggregation strategies: *majority vote* and *weighted majority vote*. Majority vote returns the most frequent class label in the neighborhood:

$$\hat{y}_m(\mathcal{N}) = \arg\max_y \sum_{i \in \mathcal{N}} \mathbf{1}(y = y_i) \quad (2)$$

while weighted majority vote adjusts label counts based on similarity in representation space:

$$\hat{y}_w(\mathcal{N}) = \arg\max_y \sum_{i \in \mathcal{N}} \mathcal{S}_f(x, x_i) \mathbf{1}(y = y_i) \quad (3)$$

Figure 3 compares majority-vote accuracy with neighborhood purity across different K values, revealing several key insights: (1) Majority voting over neighboring labels consistently produces accurate anchor predictions; (2) While larger K values decrease neighborhood purity due to noise introduction, majority-vote accuracy remains notably stable, indicating its robustness to the hyperparameter K. (3) Similarity-based weighting improves prediction stability for large K values by reducing the impact of less relevant neighbors. These findings suggest semantically similar neighborhood data can serve as effective prediction proxies, offering a potential means to enhance LLM predictions at test time.

3 Method

Motivated by our findings in Section 2, we propose TestNUC, a test-time computing strategy that leverages neighboring unlabeled data consistency to enhance LLM predictions. Our approach introduces a complementary dimension to test-time computing by integrating signals from unlabeled data during inference.

Algorithm 1 TestNUC algorithm.

Input: Embedder f, test sample x_0, unlabeled data $\mathcal{U} = \{u_i\}_{i=1}^N$, number of neighbors K, threshold θ.
$z_0 = f(x_0)$, $\mathcal{Z} = \{z_i = f(u_i)\}_{i=1}^N$
{*Extract embeddings for test sample and unlabeled data*}
$\mathcal{N} = \text{argtop}_K\{\mathcal{S}_f(x_0, x_i) | \; i = 0, \ldots, N\}$
{*Mine top-K neighbors based on similarity, note that test sample x_0 is included*}
for $k = 1$ to K **do**
  $(y_{\mathcal{N}_k}, conf_{\mathcal{N}_k}) = P_{\text{LLM}}(u_{\mathcal{N}_k})$
  {*Prompt LLM to obtain predictions and confidences*}
  $w_k = \text{sim}(z_0, z_{\mathcal{N}_k})$
  {*Compute neighbor weights based on similarity*}
  $c_k = \mathbb{1}(conf_{\mathcal{N}_k} \geq \theta)$
  {*Filter out unconfident predictions*}
end for
$y_{\text{final}} = \arg\max_y \sum_{k=1}^K c_k \, w_k \, \mathbb{1}(y_{\mathcal{N}_k} = y)$
{*Aggregate neighbors' predictions by majority voting*}
Return y_{final}

3.1 Framework Overview

TestNUC consists of two key steps:

- **Step 1: Neighbor Retrieval.** Identify the top-K nearest neighbors of a test sample based on feature similarity.
- **Step 2: Collaborative Prediction.** Prompt the LLM to generate predictions for both the test sample and its K retrieved neighbors. These predictions are combined through a designed aggregation strategy.

Note that TestNUC is based on LLM predictions instead of the ground truth label. The intuition behind TestNUC is that samples in close proximity within the embedding space are likely to share similar labels. By incorporating predictions on nearby unlabeled samples, the LLM can better contextualize and refine its decision-making. This approach aims to exploit the consistency of local data structures, effectively using unlabeled examples as an auxiliary signal to boost inference-time performance and reduce the noise and uncertainty associated with isolated predictions.

3.2 Aggregation Strategy

The aggregation strategy in Step 2 affects the sensitivity of TestNUC to noise. In this work, we explore three types of aggregation strategies.

Naive Majority Voting. The naive approach simply selects the most consistent answer across the K unlabeled data predictions.

Weighted Majority Voting. As demonstrated in our analysis in Section 2, when using a large K, neighborhood purity tends to decline rapidly. This indicates that distant neighbors can introduce significant noise and negatively impact the accuracy of majority voting. To mitigate this issue, we additionally use cosine similarity distance between the test sample and its neighbors as weights for majority voting.

Filtered Weighted Majority Voting. The quality of LLM's predictions for neighboring unlabeled data can affect the accuracy of the aggregated results. In this approach, we explore leveraging verbalized confidence to filter out low-quality predictions during majority voting. Specifically, for each unlabeled data, we ask LLM to generate both the prediction and confidence in its predictions and only high confidence predictions are kept for majority voting.

A complete algorithm for Filtered Weighted Majority Voting is presented in Algorithm 1. The algorithms for the other two voting strategies mentioned above can be obtained by removing the blue- and red-colored code. More complex aggregation strategies can also be explored, such as adding additional distance-based filtering mechanisms or confidence-weighting mechanisms, which we leave for interested researchers to explore.

4 Experiments

4.1 Experiment Setup

Tasks and Datasets. We consider eight datasets across diverse tasks with various perspectives and granularities as follows.

- **Intent Detection.** Intent detection aims to discover fine-grained intents in customer utterances. We use BANKING (Casanueva et al., 2020) and CLINC (Larson et al., 2019) for evaluation.
- **Topic Mining.** We use Reddit and StackExchange from MTEB (Muennighoff et al., 2023) and ClusterLLM (Zhang et al., 2023a) to evaluate models' ability to categorize discussion topics.
- **Domain Discovery.** For this task, we use MTOP (Li et al., 2021) and CLINC(D) (Zhang et al., 2023a) to allow evaluations of models' capability in discovering domain-specific knowledge.
- **Type Discovery.** We use the FewEvent dataset (Deng et al., 2020) that focuses on extracting event types from the given text and event triggers.
- **Emotion Recognition.** We use GoEmotion (Demszky et al., 2020), which is a dataset of Reddit comments labeled with fine-grained emotions, such as amusement, fear and gratitude.

Dataset statistics are summarized in Appendix A.

Baselines. We consider three types of baselines: ❶ **Standard Prompting**, which prompts the LLM in a standard way to select a label from the provided options to a test sample. The details of the prompt template are available in Appendix B.

❷ Test-time computing approaches that operate *at the input level* by augmenting the given prompt with additional demonstrations to enhance inference performance. Since our proposed method combines decisions based on similar examples, we compare it with two varieties of in-context learning counterparts: **TopK-ICL** (Peng et al., 2024), where the input text of the nearest neighbors of the test example are added to the prompt as context information. **TopK-ICL-P**, where we additionally append each neighbor's Standard Prompting prediction result to its text as demonstrations.

❸ Test-time computing approaches that operate *at the output level* through multiple candidate answer sampling and aggregation to boost output quality. For this category, we consider three representative approaches: **Self-Consistency** (Wang et al., 2023b), **Best-of-N** (Snell et al., 2024; Beeching et al., 2024), and **Weighted Best-of-N** (Beeching et al., 2024). Specifically, Best-of-N selects the most confident predictions out of multiple predictions based on the LLM's own verbalized confidence (Xiong et al., 2024). Weighted Best-of-N aggregates the decisions by assigning weights based on their respective confidence score.

Model	Method	Intent Detection		Topic Mining		Domain Discovery		Type	Emotion	AVG
		BANKING	CLINC	Reddit	StackEx	MTOP	CLINC(D)	FewEvent	GoEmotion	
GPT-4o-mini	Standard Prompting	0.652	0.792	0.534	0.482	0.896	0.536	0.630	0.378	0.613
	Self-Consistency	0.666	0.802	0.586	0.494	0.902	0.530	0.640	0.382	0.625
	TestNUC	0.712	0.858	0.614	0.528	0.936	0.544	0.674	0.410	0.660
	TestNUC†	**0.764**	**0.864**	**0.646**	**0.540**	**0.948**	**0.554**	**0.680**	**0.414**	**0.676**
Llama-3.1-8B	Standard Prompting	0.572	0.726	0.502	0.492	0.892	0.528	0.530	0.332	0.572
	Self-Consistency	0.620	0.774	0.564	0.526	0.902	0.518	0.564	0.340	0.601
	TestNUC	0.694	0.806	0.618	0.558	0.934	0.528	0.596	0.356	0.636
	TestNUC†	**0.724**	**0.812**	**0.646**	**0.576**	**0.940**	**0.542**	**0.614**	**0.360**	**0.652**
Claude-3-Haiku	Standard Prompting	0.680	0.848	0.486	0.564	0.892	0.552	0.594	0.336	0.619
	Self-Consistency	0.702	0.870	0.510	0.578	0.904	0.564	0.568	0.350	0.631
	TestNUC	0.762	0.894	0.596	0.588	0.940	0.590	0.620	0.348	0.667
	TestNUC†	**0.804**	**0.902**	**0.612**	**0.600**	**0.946**	**0.622**	**0.660**	**0.368**	**0.689**
GPT-4o	Standard Prompting	0.746	0.924	0.712	0.674	0.962	0.614	0.682	0.406	0.715
	Self-Consistency	0.758	0.922	0.720	0.688	0.958	0.624	0.696	0.426	0.724
	TestNUC	0.804	0.934	0.744	**0.710**	0.974	0.644	0.692	0.446	0.744
	TestNUC†	**0.824**	**0.940**	**0.750**	**0.710**	**0.978**	**0.654**	**0.708**	**0.464**	**0.754**

Table 1: Accuracy comparison with Standard Prompting and Self-Consistency across four diverse LLMs. TestNUC consistently improves the inference performance on all benchmark datasets. † denotes that 50 neighbors are utilized.

Method	Intent Discovery		Topic Mining		Domain Discovery		Type	Emotion	AVG
	BANKING	CLINC	Reddit	StackEx	MTOP	CLINC(D)	FewEvent	GoEmotion	
KNN-ICL	0.664	0.768	0.670	0.520	0.942	0.518	0.570	0.386	0.630
w. TestNUC	0.762	0.832	0.728	0.566	0.960	0.544	0.606	0.410	0.676
Improvement	**14.76%**	**8.33%**	**8.66%**	**8.85%**	**1.91%**	**5.02%**	**6.32%**	**6.22%**	**7.51%**
KNN-ICL-P	0.702	0.870	0.620	0.556	0.922	0.548	0.624	0.416	0.657
w. TestNUC	0.768	0.894	0.672	0.584	0.960	0.584	0.654	0.444	0.695
Improvement	**9.40%**	**2.76%**	**8.39%**	**5.04%**	**4.12%**	**6.57%**	**4.81%**	**6.73%**	**5.98%**
Self-Consistency	0.666	0.802	0.586	0.494	0.902	0.530	0.640	0.382	0.625
w. TestNUC	0.750	0.878	0.706	0.562	0.928	0.566	0.670	0.420	0.685
Improvement	**12.61%**	**9.48%**	**20.48%**	**13.77%**	**2.88%**	**6.79%**	**3.69%**	**9.95%**	**9.56%**
Best-of-N	0.662	0.814	0.606	0.492	0.902	0.544	0.620	0.378	0.627
w. TestNUC	0.758	0.880	0.706	0.568	0.954	0.564	0.696	0.412	0.692
Improvement	**14.50%**	**8.11%**	**16.50%**	**15.45%**	**5.76%**	**3.68%**	**12.26%**	**8.99%**	**10.36%**
Weighted Best-of-N	0.658	0.820	0.602	0.484	0.900	0.532	0.612	0.372	0.623
w. TestNUC	0.752	0.876	0.710	0.558	0.938	0.566	0.672	0.422	0.687
Improvement	**14.29%**	**6.83%**	**17.94%**	**15.29%**	**4.22%**	**6.39%**	**9.80%**	**13.44%**	**10.32%**

Table 2: TestNUC can significantly enhance various existing test-time computing approaches - both those that prepend demonstrations at the input level (ICL-based) and those that do sampling and "post-hoc" candidate refinements at the output level (Self-Consistency, Best-of-N). The relative improvement is visualized in Figure 1.

Implementation Details. We utilize both open-sourced and close-sourced LLMs with varying scales: GPT-4o-mini, GPT-4o (OpenAI, 2024), Llama-3.1-8B (Dubey et al., 2024), Claude 3 Haiku (Anthropic, 2024). We set temperature $T = 0.7$ and Top-p $= 1.0$ for sampling decoding for all evaluated language models. By default, the number of candidate answers N we sampled for Self-Consistency, Best-of-N and Weighted Best-of-N is 10. Similarly, the number of retrieved neighbors, i.e., K, for TopK-ICL, TopK-ICL-P, and our TestNUC is 10 unless stated otherwise. We adopt NV-Embed-v2-7B (Lee et al., 2024) as the embedding model for all methods. Due to resource constraints, we randomly sample 500 data points from each dataset for evaluation and use the remaining for neighboring sample retrieval.

4.2 Main Results

Comparison with Standard Prompting and Self-Consistency. Table 1 presents the comparison results with Standard Prompting and Self-Consistency across four large language models. It can be observed that TestNUC significantly improves the inference performance of four large lan-

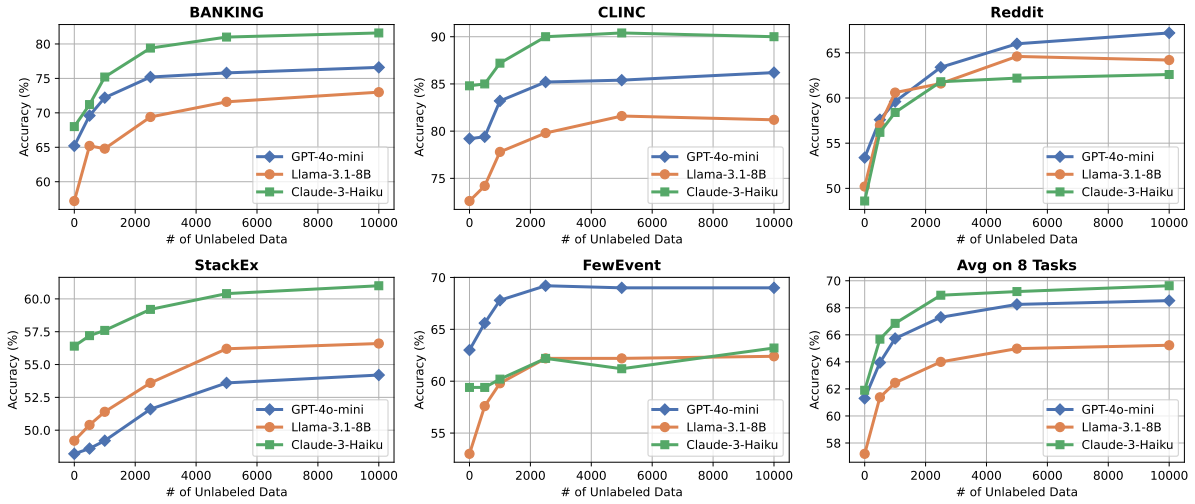

Figure 4: Increasing the amount of unlabeled data consistently boosts performance across all evaluated LLMs and datasets. The scaling trends are more distinctly visible in the logarithmic version of the figure (Figure 5).

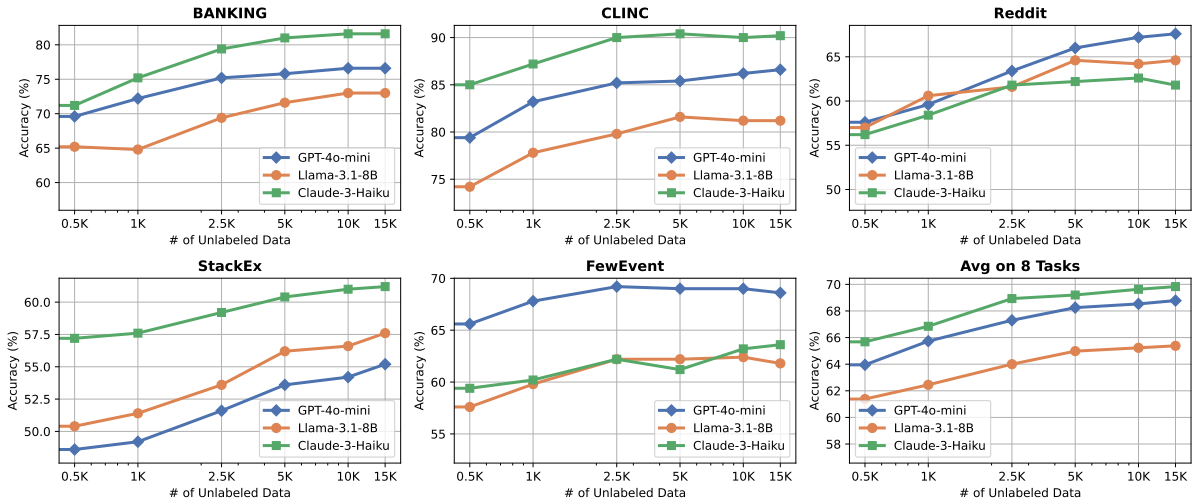

Figure 5: The logarithmic version of Figure 4.

guage models on all eight evaluated datasets over standard prompting. TestNUC can also outperform Self-Consistency when utilizing the same amount of sampling paths and neighboring unlabeled data (i.e., K=10 in both cases). For example, TestNUC surpasses Self-Consistency by 5.87% on average when using Llama-3.1-8B model and 5.48% on average when using GPT-4o-mini. Besides, TestNUCperformance can be further boosted by utilizing more neighboring unlabeled data. TestNUC†, which utilizes 50 neighbors, can improve the performance over standard prompting up to 11.35% on average across 8 datasets when using Claude-3-Haiku. Additionally, performance improvements are observed across all four language models, even in the already powerful GPT-4o model.

TestNUC can enhance various existing test-time computing approaches. The results are shown in Table 2. Across all baselines and all datasets, incorporating TestNUC boosts performance. The average improvements range from about +6% (TopK-ICL-P) to +10% (Best-of-N and Weighted Best-of-N), indicating that TestNUC is complementary to diverse inference strategies—both those that prepend demonstrations at the input level (ICL-based) and those that do sampling and "post-hoc" candidate refinements at the output level (Self-Consistency, Best-of-N). Methods that work at the output level (e.g., Self-Consistency, Best-of-N) present larger average gains (9–10%), compared to input-level approaches (6–7%). The biggest performance gains often appear on the Topic Mining

Aggregation Strategy	BANKING	CLINC	Reddit	StackEx	MTOP	CLINC(D)	FewEvent	GoEmotion	AVG
Standard Prompting	0.652	0.792	0.534	0.482	0.896	0.536	0.630	0.378	0.613
Naive Majority Voting	0.756	0.862	0.644	0.538	0.948	0.550	0.668	0.392	0.670
Weighted Majority Voting (Distance)	0.764	0.864	0.646	0.540	0.948	0.554	0.680	0.414	0.676
Weighted Majority Voting (Distance & Confidence)	**0.768**	0.870	0.656	0.542	0.948	0.552	0.684	**0.416**	0.680
Filtered Weighted Majority Voting	0.762	**0.876**	**0.688**	**0.542**	**0.954**	**0.572**	**0.688**	0.410	**0.687**

Table 3: Comparison of aggregation strategies across diverse datasets. Naive majority voting already significantly improves accuracy over standard prompting. Weighted Majority Voting with distance and confidence further enhances performance, and filtering low-confidence predictions achieves the highest average result.

Embedder	BANKING	CLINC	Reddit	MTOP	AVG
Standard Prompting	0.652	0.792	0.534	0.896	0.719
all-MiniLM-L12-v2-120M	0.706	0.832	0.584	0.928	0.763
all-distilroberta-v1-290M	0.690	0.840	0.586	0.938	0.764
all-mpnet-base-v2-420M	0.712	0.852	0.586	0.942	0.773
gte-Qwen2-1.5B-instruct	0.694	0.844	0.614	0.946	0.775
stella-en-400M-v5	0.728	0.834	0.618	0.946	0.782
NV-Embed-v2-7B	**0.764**	**0.864**	**0.646**	**0.948**	**0.806**

Table 4: Results with varying embedding models. TestNUC is effective on diverse embedding models from different companies and of different sizes.

tasks (e.g., Reddit, StackEx), where improvements can exceed +15–20%. This suggests that adding TestNUC is especially helpful in scenarios involving more open-ended or noisy textual inputs, where post-hoc aggregations can more effectively disambiguate the model's initial outputs.

5 Additional Studies

5.1 Influence of Unlabeled Data Size

Increasing unlabeled data helps boost performance across tasks. Figure 4 reports the linear-scale results on BANKING, CLINC, Reddit, StackEx, FewEvent, and an overall average across eight tasks. In all cases, increasing the unlabeled set yields notable accuracy improvements for GPT-4o-mini, Llama-3.1-8B, and Claude-3-Haiku. Adding even a modest number (e.g., 500–1k) of unlabeled instances yields substantial accuracy gains, especially on BANKING and Reddit. However, improvements taper off after roughly 8–10k unlabeled samples, suggesting a saturation point where additional unlabeled data provides diminishing returns. The log-scale plots in Figure 5 further highlight these trends, confirming that the utility of unlabeled examples gradually diminishes but still delivers meaningful improvements up to the 10K–15K range. This pattern holds consistently across all tasks, confirming that increasing unlabeled data universally improves performance but with predictable saturation effects.

5.2 Aggregation Strategy Comparison

We find that **naive majority voting greatly surpasses standard prompting performance, with advanced strategies further enhancing results.** As shown in Table 3, simply aggregating multiple predictions with naive majority voting can already boost average accuracy significantly from 0.613 to 0.670. Introducing distance and confidence weighting further refines these gains from the average to 0.680. Finally, filtering out low-confidence predictions yields the highest performance, although on certain tasks (e.g., GoEmotion), Weighted Majority Voting (Distance & Confidence) can be more effective, suggesting that a carefully tuned confidence threshold may be necessary for each dataset.

5.3 Varying Embedding Models

TestNUC works on different sizes of embedders. The embedders used to generate data embeddings for neighbor retrieval play a crucial role in the success of TestNUC. In this work, we explore diverse embedding models, including public encoders from different companies and embedders of different sizes ranging from 120M to 7B. As shown in Table 4, TestNUC is effective when applied to various embedding models. Not surprisingly, TestNUC achieves significant improvements with larger and more advanced embedders, such as NV-Embed-v2-7B, which records the highest average performance (0.755) and excels across all datasets. Mid-sized embedders like stella-en-400M-v5 (0.738) and gte-Qwen2-1.5B-instruct (0.732) also perform well, demonstrating that TestNUC can effectively leverage diverse embedding architectures. Even smaller models, such as all-MiniLM-L12-v2-120M (0.720), deliver competitive results over standard prompting (0.682), showcasing TestNUC's robustness across varying model sizes and complexities.

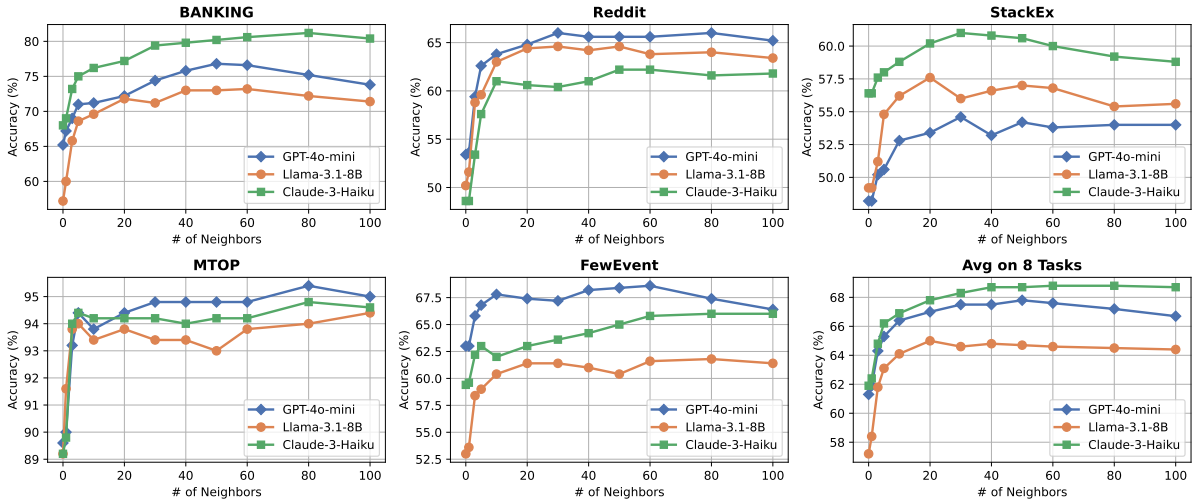

Figure 6: Influence of the number of neighbors. The results show that even a small set of neighbors can significantly boost performance for all three LLMs, significantly surpassing their zero-neighbor baselines.

5.4 Influence of Neighbor Size

Influence of the Number of Neighboring Unlabeled Data. Figure 6 shows the results of TestNUC from GPT-4o-mini, Llama-3.1-8B, and Claude-3-Haiku as the number of neighbors increases. The results show that even a small set of neighbors can significantly boost performance for all three models, significantly surpassing their zero-neighbor baselines. Additionally, all three models generally benefit from increasing the number of neighbors from 0 to 60, although the gains tend to plateau after approximately 40–60 neighbors. Notably, the performance of GPT-4o-mini and Llama-3.1-8B slightly decreases when the number of neighbors increases from 60 to 80 on certain datasets, likely due to the introduction of more noisy neighbors. In contrast, Claude-3-Haiku often achieves higher accuracies with relatively larger neighborhood sizes (e.g., 60–100), indicating greater robustness to noise.

6 Related Work

6.1 Test-Time Computing

Test-time compute (Snell et al., 2024) improves LLM performance by modifying the prediction distribution during test time. Such modification is usually accompanied with extra computational cost. Instead of decoding greedily, the model may sample multiple decoding paths before aggregating them into a response. Chain of Thought (Wei et al., 2022) modifies the output distribution through hand-crafted prompts that contain reasoning chains. Self-consistency (Wang et al., 2023b) samples multiple chain-of-thought paths and aggregate the sample with majority voting. (Portillo Wightman et al., 2023) observed improved accuracy and robustness by querying the model with semantically equivalent prompts before responding with the majority answer. (Gao et al., 2021) uses sentence embeddings to retrieve k-nearest-neighbor demonstration for in-context learning. (Ye et al., 2023) retrieves relevant and diverse demonstrations by training a model that predicts the relevance of a demonstration via contrastive learning (Chen et al., 2020). Our work is directly inspired by the KNN method proposed by (Gao et al., 2021). Later work has revealed that similarity based demonstration retrieval improves in-context learning because LLMs attend to the most similar demonstration during few-shot prompting (Wang et al., 2023a). Instead of using similar demonstrations for in-context learning, we explore using them as near neighbors in the fashion of non-parametric prediction.

6.2 In-Context Learning

Apart from Chain-of-Though, many work explore the possibility of using self-generated content by the LLM to aid with reasoning or classification. STaR (Zelikman et al., 2022) iteratively add self-generated rationales that are proved correct by a verifier to the exist pool of demonstrations. A significant limitation of STaR is that it relies on knowing the correct answer to the questions the LLM is generating rationale for. Our method simply make predictions for neighboring examples,

which does not require ground truth labels. Auto-CoT(Zhang et al., 2023b) uses self-generated rationales as demonstrations for similar inputs. The generated data by Auto-CoT incurs a quadratically scaling overhead to the final prediction. Our proposed method only incurs a linearly scaling overhead due to the nature of nearest-neighbor algorithm. Self-ICL (Chen et al., 2023) generated its own demonstration and their pseudo-labels and uses them as demonstrations. We disagree with Self-ICL's premise that even unlabeled data are hard to come by in realistic settings, and posit that unlabeled data are abundant and inexpensive to obtain (Zou et al., 2023a,b). Thus, self-generated demonstration inputs are unnecessary. Like Auto-CoT, Self-ICL's test-time compute overhead also scales quadratically. Lastly, Auto-CoT, STaR, and Self-ICL all focuses on reasoning tasks, whereas our work primarily focuses on classification tasks.

7 Conclusion

In this work, we introduced TestNUC, a simple yet effective approach that leverages the consistency of neighboring unlabeled data to enhance test-time predictions in large language models. Extensive experiments across eight datasets and multiple LLMs demonstrate that TestNUC consistently outperforms baselines like standard prompting and self-consistency. It can be seamlessly integrated with existing methods such as TopK-ICL, self-consistency, and best-of-N to yield further gains. These results highlight the practical value of leveraging unlabeled data during inference, which not only boosts label consistency but also offers a scalable path to better generalization in real-world applications where labeled data may be scarce.

8 Limitation

Our evaluation of TestNUC is limited to classification tasks and does not include generative tasks. We leave this extension for future work. Due to computational resource constraints and limited budgets, we did not evaluate recent powerful reasoning models such as o3-mini and DeepSeek-R1.

Acknowledgements

This work is supported in part by NSF under grants III-2106758, and POSE-2346158.

References

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. *https://www.anthropic.com/news/claude-3-family*.

Edward Beeching, Lewis Tunstall, and Sasha Rush. 2024. Scaling test-time compute with open models.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. 2019. Mixmatch: A holistic approach to semi-supervised learning. *Advances in neural information processing systems*, 32.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45, Online. Association for Computational Linguistics.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR.

Wei-Lin Chen, Cheng-Kuang Wu, Yun-Nung Chen, and Hsin-Hsi Chen. 2023. Self-icl: Zero-shot in-context learning with self-generated demonstrations.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. 2020. GoEmotions: A Dataset of Fine-Grained Emotions. In *58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Shumin Deng, Ningyu Zhang, Jiaojian Kang, Yichi Zhang, Wei Zhang, and Huajun Chen. 2020. Meta-learning with dynamic-memory-based prototypical network for few-shot event detection. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, WSDM '20, page 151–159. ACM.

Xiangjue Dong, Maria Teleki, and James Caverlee. 2024. A survey on llm inference-time self-improvement. *arXiv preprint arXiv:2412.14352*.

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

Hongfu Gao, Feipeng Zhang, Wenyu Jiang, Jun Shu, Feng Zheng, and Hongxin Wei. 2024. On the noise robustness of in-context learning for text generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. 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 3816–3830, Online. Association for Computational Linguistics.

Zhengyao Gu, Henry Peng Zou, Yankai Chen, Aiwei Liu, Weizhi Zhang, and Philip S Yu. 2025. Semi-supervised in-context learning: A baseline study. *arXiv preprint arXiv:2503.03062*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*.

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.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, and Yanai Elazar. 2023. Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12284–12314, Toronto, Canada. Association for Computational Linguistics.

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.

OpenAI. 2024. Gpt-4o system card. *https://arxiv.org/pdf/2410.21276*.

Keqin Peng, Liang Ding, Yancheng Yuan, Xuebo Liu, Min Zhang, Yuanxin Ouyang, and Dacheng Tao. 2024. Revisiting demonstration selection strategies in in-context learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9090–9101, Bangkok, Thailand. Association for Computational Linguistics.

Gwenyth Portillo Wightman, Alexandra Delucia, and Mark Dredze. 2023. Strength in numbers: Estimating confidence of large language models by prompt agreement. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pages 326–362, Toronto, Canada. Association for Computational Linguistics.

Weijia Shi, Julian Michael, Suchin Gururangan, and Luke Zettlemoyer. 2022. Nearest neighbor zero-shot inference. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3254–3265, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. 2020. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33:596–608.

Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. 2020. Scan: Learning to classify images without labels. In *European conference on computer vision*, pages 268–285. Springer.

Fangxin Wang, Kay Liu, Sourav Medya, and Philip S. Yu. 2025. BANGS: Game-theoretic node selection for graph self-training. In *The Thirteenth International Conference on Learning Representations*.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023a. Label words are anchors: An information flow perspective for understanding in-context learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9840–9855, Singapore. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*,

volume 35, pages 24824–24837. Curran Associates, Inc.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2024. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms.

Benfeng Xu, Quan Wang, Zhendong Mao, Yajuan Lyu, Qiaoqiao She, and Yongdong Zhang. 2023. $k$nn prompting: Beyond-context learning with calibration-free nearest neighbor inference.

Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. 2023. Compositional exemplars for in-context learning. In *International Conference on Machine Learning*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, volume 35, pages 15476–15488. Curran Associates, Inc.

Yuwei Zhang, Zihan Wang, and Jingbo Shang. 2023a. 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.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023b. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Yue Zhou, Yada Zhu, Diego Antognini, Yoon Kim, and Yang Zhang. 2024. Paraphrase and solve: Exploring and exploiting the impact of surface form on mathematical reasoning in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2793–2804, Mexico City, Mexico. Association for Computational Linguistics.

Henry Zou and Cornelia Caragea. 2023. JointMatch: A unified approach for diverse and collaborative pseudo-labeling to semi-supervised text classification. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7290–7301, Singapore. Association for Computational Linguistics.

Henry Zou, Yue Zhou, Weizhi Zhang, and Cornelia Caragea. 2023a. DeCrisisMB: Debiased semi-supervised learning for crisis tweet classification via memory bank. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6104–6115, Singapore. Association for Computational Linguistics.

Henry Peng Zou, Cornelia Caragea, Yue Zhou, and Doina Caragea. 2023b. Semi-supervised few-shot learning for fine-grained disaster tweet classification. In *Proceedings of the 20th International ISCRAM Conference*. ISCRAM 2023.

Henry Peng Zou, Siffi Singh, Yi Nian, Jianfeng He, Jason Cai, Saab Mansour, and Hang Su. 2025. Glean: Generalized category discovery with diverse and quality-enhanced llm feedback. *arXiv preprint arXiv:2502.18414*.

A Dataset Statistics

Table 5 provides the dataset statistics summary for all evaluated datasets.

Task	Dataset	# Classes	Total	Test
Intent Detection	BANKING	77	10,003	500
	CLINC	150	15,000	500
Topic Mining	Reddit	50	25,000	500
	StackExchange	121	25,000	500
Domain Discovery	MTOP	11	15,667	500
	CLINC(D)	10	15,000	500
Type Discovery	FewEvent	34	18909	500
Emotion Detection	GoEmotion	27	23485	500

Table 5: Dataset statistics.

B Prompt Template

The prompt template we used in the experiments is listed below. Note that we use the same prompt template for all methods for fair comparisons.

Prompt Template

Instruction: Please select a label from the provided options for the following testing samples and also show your confidence in the label assignment by providing a probability between 0 and 1.

Label Options: `[A List of Labels]`.

== Testing Samples ==
`[Testing Samples]`

C LLM Predictions Can Be Inaccurate and Unstable

Figure 7 demonstrates the error rate and inconsistency ratio of predictions by different LLMs on diverse datasets. The inconsistency ratio here refers to the proportion of prediction changes when an LLM is rerun N times for the same input query across the entire dataset. The results are obtained using standard zero-shot prompting with a temperature of 0.7 and $N = 10$. It can be observed that even in standard text classification tasks, LLMs can produce inaccurate and inconsistent prediction results for ambiguous or challenging data points.

D Robustness in Adversarial Scenarios

In real-world applications, suitable in-distribution data may not always be available, and retrieved samples could be out-of-distribution. This section demonstrates the robustness of our TestNUC method in such adversarial scenarios. We conducted an adversarial experiment by replacing in-distribution samples in the BANKING dataset with out-of-distribution samples (i.e., outliers) from the REDDIT dataset. To create sufficiently challenging scenarios, we replaced 60% and 75% of the in-distribution samples with OOD samples. As shown in the Table 6, even with 60%–75% OOD samples present, TestNUC still significantly improves baseline performance across many cases in these highly noisy scenarios, demonstrating its robustness and effectiveness.

Furthermore, when using Weighted Majority Voting (WMV) instead of Naive Majority Voting (NMV), TestNUC's performance and robustness can be further enhanced. This is because OOD samples are likely to have lower semantic similarity with the test sample compared to in-distribution samples, and thus the model can assign lower weights to OOD samples when using WMV. This is also consistent with our previous findings that WMV outperforms NMV in most cases.

OOD Ratios	60%		75%	
# Neighbors	NMV	WMV	NMV	WMV
0	63.7±0.6	63.7±0.6	63.7±0.6	63.7±0.6
3	67.9±0.7	68.2±0.9	68.3±0.7	69.2±0.9
5	69.6±0.8	70.6±0.8	70.3±0.7	71.1±0.6
10	72.3±0.8	72.8±0.6	73.7±0.7	74.0±0.5
20	74.7±0.3	75.0±0.4	73.3±0.7	73.3±0.8
30	75.9±0.7	75.6±0.6	72.1±0.6	72.8±0.7
40	75.7±0.8	76.3±0.6	69.1±0.8	70.8±0.9
50	74.7±0.6	75.2±0.6	67.0±0.7	69.6±0.8
60	73.5±0.6	73.7±0.5	62.0±0.8	66.0±0.7
80	71.3±0.6	72.1±0.7	54.9±0.6	61.7±0.8
100	67.3±0.7	70.1±0.7	48.8±0.6	59.0±0.7

Table 6: Performance of TestNUC under adversarial conditions with out-of-distribution (OOD) samples.

E Runtime and Cost Analysis

This section presents an estimated analysis of the trade-off between gains in accuracy (average on all 8 tasks) and the compute/runtime cost required for the approach in different settings. The following table summarizes the estimated runtime and cost

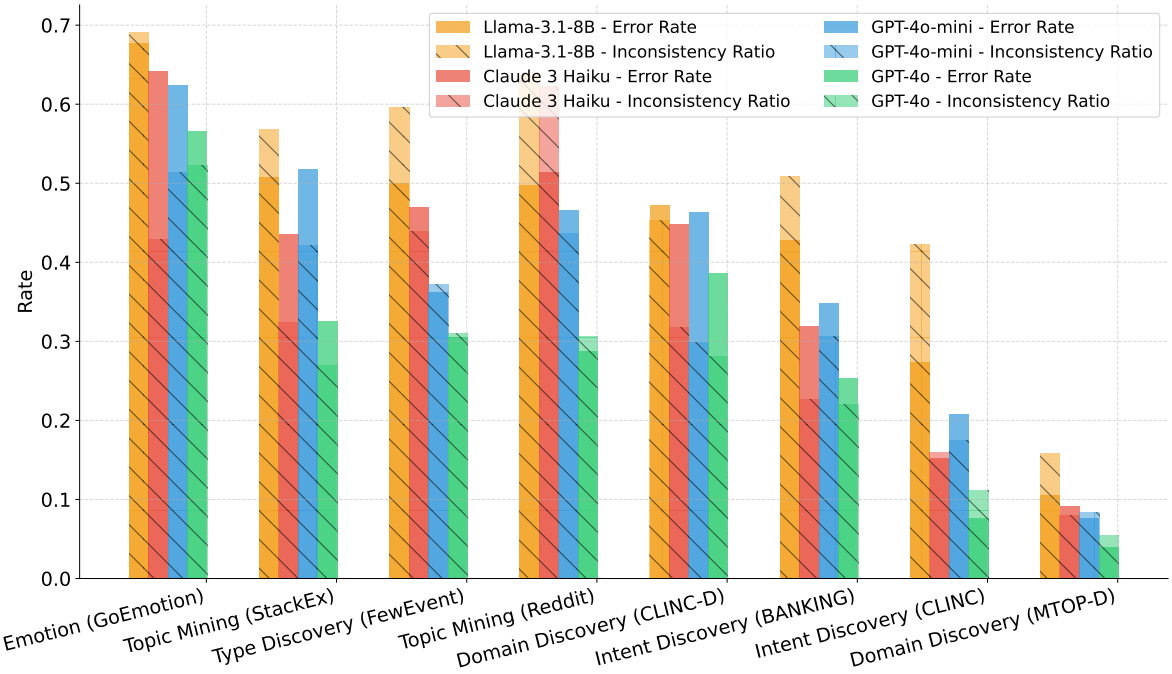

Figure 7: LLM predictions can be inaccurate and unstable.

per sample using GPT-4o-mini. The number of retrieved neighbors and sampled candidate answers is set to 10 by default, unless otherwise specified.

We include two variants of TestNUC: TestNUC and TestNUC-S. TestNUC-S is an efficient implementation of TestNUC that pre-computes and stores the embeddings and predictions of previously queried or seen samples. Thus, when a new sample arrives, TestNUC-S can simply retrieve the embeddings and predictions of the nearest neighbors from the stored set and use them to generate the final label without additional LLM calls—where the runtime cost for retrieval is negligible compared to querying the LLM. As shown in Table 7, when increasing the number of retrieved neighbors (K=5 to K=50), TestNUC can greatly improves performance although at the cost of increased runtime and cost. When using the same number of retrieved neighbors (K=10), TestNUC is more efficient than the best-performing baseline KNN-ICL-P, which incurs a computational cost that scales quadratically with K, whereas TestNUC incurs only a linear cost. Moreover, the efficient implementation TestNUC-S can significantly reduce runtime cost and is also a very practical solution for real-world applications, as storing the embeddings and queries of previously queried samples is both quite common and cheap in practice.

	Runtime (s)	Cost ($)	Performance
Standard Prompting	0.6820	$0.000028	0.613
w. TestNUC (K=5)	3.4100	$0.000140	0.648
w. TestNUC (K=10)	6.8200	$0.000280	0.660
w. TestNUC (K=50)	34.1000	$0.001400	0.676
w. TestNUC-S (K=5)	**0.6842**	**$0.000028**	**0.648**
w. TestNUC-S (K=10)	**0.6843**	**$0.000028**	**0.660**
w. TestNUC-S (K=50)	**0.6847**	**$0.000028**	**0.676**
KNN-ICL	0.6972	$0.000085	0.630
KNN-ICL-P	7.5510	$0.000371	0.657
Self-Consistency	6.8200	$0.000280	0.625
Best-of-N	6.8200	$0.000280	0.627

Table 7: Runtime and cost analysis. TestNUC improves performance with more neighbors at higher cost, while TestNUC-S achieves the same gains with negligible runtime by reusing stored embeddings and predictions.