Text Clustering as Classification with LLMs (Appendix)

Chen Huang
Singapore University of Technology and Design
Singapore, Singapore
chen_huang@mymail.sutd.edu.sg

Guoxiu He
East China Normal University
Shanghai, China
gxhe@fem.ecnu.edu.cn

ACM Reference Format:
Chen Huang and Guoxiu He. 2025. Text Clustering as Classification with LLMs (Appendix). In *Proceedings of the 2025 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region (SIGIR-AP 2025), December 7–10, 2025, Xi'an, China.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3767695.3769519

Appendix

A Prompt template

We design different prompt templates ($\mathcal{P}g$, $\mathcal{P}m$, $\mathcal{P}a$) and instructions ($\mathcal{I}$ generate, $\mathcal{I}$ merge, $\mathcal{I}$ assign) for the three sub-tasks in our framework: label generation, label aggregation & merging, and given-label classification. Each template is carefully constructed to guide the LLMs toward producing high-quality, task-specific outputs with minimal ambiguity. Table 1 provides an overview of the prompt templates and corresponding instructions for each task, illustrating how we tailor the query to maximize the LLM's performance.

To improve the reliability and consistency of responses, we integrate format-control instructions into the prompts. For example, we explicitly include directives such as "Please return the output in JSON format" and provide a concrete JSON structure example. This approach ensures that the LLM outputs are not only correct but also well-structured for downstream processing. Moreover, by demonstrating the expected output format within the prompt, we reduce the chance of incomplete or incorrectly formatted responses.

To showcase the practical effectiveness of these templates and instructions, we present a case study on the MTOP-I dataset for each sub-task. Table 2 highlights how the designed prompts drive the LLM to produce interpretable labels, merge semantically redundant labels, and assign the final labels accurately.

B Calculation of Cost and Time

In this section, we describe how the monetary cost and wall-clock time are calculated for both the API-based method and the fine-tuning-based method. Let N denote the size of the evaluation dataset, and let D be the size of the training set with sequence length L, trained for E epochs. The input and output token lengths for the LLM are represented by T_{in} and T_{out}, respectively. Since API calls can be executed in parallel, we assume a parallel throughput of R requests per second.

The standard API price for GPT-3.5-turbo is \$0.50 per million input tokens and \$1.50 per million output tokens[1]. For fine-tuning, we use the average rental cost of \$1.73 per A100 (80GB) GPU per hour across providers[2]. Following Zhang et al. [1], we set D = 60,000, $E = 15$, and $L = 512$. We fix $T_{\text{in}} = 200$, $T_{\text{out}} = 50$, and $R = 100$ for all calculations.

Throughput estimation. We estimate throughput for fine-tuning and inference based on the effective processing rate of transformer encoders on 4×A100 GPUs (80GB). The training throughput is

$$\tau_{\text{train}} = \frac{B \cdot G}{t_{\text{iter}}}, \qquad \tau_{\text{inf}} = \frac{B_{\text{inf}} \cdot G}{t_{\text{iter}}^{\text{inf}}}, \tag{1}$$

where B is the per-GPU training batch size, B_{inf} is the per-GPU inference batch size, G is the number of GPUs, t_{iter} is the iteration time during training, and $t_{\text{iter}}^{\text{inf}}$ is the iteration time during forward-only inference. In practice, with $B = 64$, $G = 4$, and $t_{\text{iter}} \approx 0.85$s for sequence length $L = 512$, we obtain $\tau_{\text{train}} \approx 64 \times 4/0.85 \approx 300$ samples/s. For inference, with $B_{\text{inf}} = 128$ and $t_{\text{iter}}^{\text{inf}} \approx 0.5$s, we obtain $\tau_{\text{inf}} \approx 128 \times 4/0.5 \approx 1000$ requests/s. These values are consistent with reported throughput benchmarks for BERT-style encoders on A100 GPUs and represent conservative but realistic estimates that balance compute, memory, and data loader efficiency.

API-based method. The API cost and time are given by

$$\text{Cost}_{\text{API}} = \frac{N}{10^6}\Big(T_{\text{in}} \cdot 0.50 + T_{\text{out}} \cdot 1.50\Big), \qquad t_{\text{API}} = \frac{N}{R}. \tag{2}$$

Note that the batch size does not affect the total token usage, and therefore does not appear in the formula.

Fine-tuning-based method. The training time and cost are

$$t_{\text{train}} = \frac{D \cdot E}{\tau_{\text{train}}}, \qquad \text{Cost}_{\text{train}} = \frac{t_{\text{train}}}{3600} \cdot 4 \cdot C_{\text{GPU}}, \tag{3}$$

where C_{GPU} is the hourly rental price per GPU. Inference requires

$$t_{\text{inf}} = \frac{N}{\tau_{\text{inf}}}, \qquad \text{Cost}_{\text{inf}} = \frac{t_{\text{inf}}}{3600} \cdot 4 \cdot C_{\text{GPU}}. \tag{4}$$

The total cost and time for the fine-tuned model are therefore

$$\text{Cost}_{\text{FT}} = \text{Cost}_{\text{train}} + \text{Cost}_{\text{inf}}, \qquad t_{\text{FT}} = t_{\text{train}} + t_{\text{inf}}. \tag{5}$$

References

[1] 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*. 13903–13920.

[1] https://platform.openai.com/docs/pricing
[2] Sources: https://lambda.ai/service/gpu-cloud, https://www.runpod.io/gpu-models/a100-pcie

SIGIR-AP 2025, Xi'an, China

ACM ISBN 979-8-4007-2218-9/2025/12
https://doi.org/10.1145/3767695.3769519

Table 1: Prompt template and instructions used in this paper. In this template, words inside {} should be replaced by corresponding variables during experiments.

Task	Prompt template with instruction $\mathcal{I}$
Label Generation $\mathcal{P}_g$	Given the labels, under a text classification scenario, can all these text match the label given? If the sentence does not match any of the label, please generate a meaningful new label name. Labels: {given_labels} Sentences: {sentence_list} You should NOT return meaningless label names such as 'new_label_1' or 'unknown_topic_1' and only return the new label names, please return in json format like: {json_example}
Aggregating and merging labels $\mathcal{P}_m$	Please analyze the provided list of labels to identify entries that are similar or duplicate, considering synonyms, variations in phrasing, and closely related terms that essentially refer to the same concept. Your task is to merge these similar entries into a single representative label for each unique concept identified. The goal is to simplify the list by reducing redundancies without organizing it into subcategories or altering its fundamental structure. Here is the list of labels for analysis and simplification:{label_list}. Produce the final, simplified list in a flat, JSON-formatted structure without any substructures or hierarchical categorization like: {json_example}
Given label classification $\mathcal{P}_a$	Given the label list and the sentence, please categorize the sentence into one of the labels. Label list: {label_list} Sentence: {sentence} You should only return the label name, please return in json format like: {json_example}

Table 2: Case study in the MTOP-I dataset for different tasks. The '...' in the prompts and LLM responses indicate omitted labels to provide a clear presentation of the case study.

Task	Question and Answer	
Label Generation	Prompt	Given the labels, under a text classification scenario, can all these text match the label given? If the sentence does not match any of the label, please generate a meaningful new label name. Labels: ['update call', 'get weather', 'update method call', ... , 'resume music', 'get stories news', 'get reminder location'] Sentences: ['do not play any song by lil Yachty', 'show me video messages from my grandma', ... , 'please add a one and a half minute timer', 'read national news headlines'] You should NOT return meaningless label names such as 'new_label_1' or 'unknown_topic_1' and only return the new label names, please return in json format like: {'labels': ['label name', 'label name']}

	LLM response	{
		"labels": ["get info contact", "get stories news", "create reminder"]
		}

Aggregating and merging labels	Prompt	Please analyze the provided list of labels to identify entries that are similar or duplicate, considering synonyms, variations in phrasing, and closely related terms that essentially refer to the same concept. Your task is to merge these similar entries into a single representative label for each unique concept identified. The goal is to simplify the list by reducing redundancies without organizing it into subcategories or altering its fundamental structure. Here is the list of labels for analysis and simplification:['update call', 'get weather', ... , 'get reminder location', 'create reminder']. Produce the final, simplified list in a flat, JSON-formatted structure without any substructures or hierarchical categorization like: {'merged_labels': ['label name', 'label name']}

	LLM response	{
		"merged_labels": ["update call","get weather","update method call",... "get stories news","get reminder location"]
		}

Given label classification	Prompt	Given the label list and the sentence, please categorize the sentence into one of the labels. Label list: ['update call', 'get weather', ... , 'get reminder location', "create reminder"] Sentence: "Show me dates for music festivals in 2018." You should only return the label name, please return in json format like: {'label_name': 'label'}

	LLM response	{
		"label_name": "get event"
		}