Batch Prompting for Text Classification: A Multi-Objective Study of Quality, Cost, and Batch Design

Anonymous ACL submission

Abstract

Batch prompting can reduce token costs in LLM-based text classification by amortizing shared prompt components across multiple documents, but larger and differently composed batches may alter prediction quality. This paper formulates batch prompting as a multi-objective optimization problem that jointly maximizes Macro-F1 and minimizes total token consumption under two controls: the maximum number of documents per batch and the maximum total batch size in tokens. Using binary, multiclass, and multilabel classification tasks, we compare batching strategies that vary in document grouping and within-batch ordering through Pareto front analysis, common-reference hypervolume, pairwise non-domination, and regime-specific operating-point selection. The results show that no universally optimal batching strategy exists: the strongest frontiers and recommended operating-points depend on both the dataset and the target regime of use. Fake News dataset favors more careful batch composition, GoEmotions permits especially strong cost reduction with competitive quality, and Yahoo! Answers shows the most heterogeneous trade-off structure. In several regimes, batch prompting matches or exceeds single-document prompting while using substantially fewer tokens. The ablation analysis further shows that document grouping is the more stable design axis, whereas document ordering contributes more conditionally and with greater dataset dependence.

1 Introduction

Large language models are increasingly used for text classification tasks (Edwards and Camacho-Collados, 2024; Guo et al., 2024; Ma et al., 2025; Rahman and Caragea, 2025; Sabri et al., 2025; Sun et al., 2023). In many applications, classification is performed over an entire corpus rather than over isolated documents. In this setting, the total cost is determined not only by the input data itself, but also by prompt components that recur for every document, such as instructions, class descriptions, document wrappers, and the output template. If each document is processed in a separate call, these shared components are repeatedly transmitted, which can substantially increase total token consumption (Cheng et al., 2023; Fan et al., 2023). Batch prompting addresses this overhead by processing multiple documents jointly within one call (Cheng et al., 2023; Fan et al., 2023; Feng et al., 2024; Ji et al., 2025; Lin et al., 2024). However, larger batches also create longer contexts, which may alter response quality (Hsieh et al., 2024b; Liu et al., 2024; Peysakhovich and Lerer, 2023). The central problem is therefore a quality–cost trade-off, not simply maximizing batch size.

For text classification, this trade-off is not governed by document count alone. Batch formation is controlled by at least two related but non-interchangeable constraints: the maximum number of documents in a batch and the maximum total batch length in tokens. A batch may therefore contain many short documents or a few long ones, and these settings may differ in both token cost and predictive quality. In addition, batching strategy depends on two separable design choices: how documents are grouped into batches and how they are ordered within a batch. This distinction is plausible because LLM behavior may depend on context position and composition (Fan et al., 2023; Hsieh et al., 2024b; Ji et al., 2025; Liu et al., 2024; Peysakhovich and Lerer, 2023; Rubin et al., 2022; Wang et al., 2024).

Existing work on batch prompting shows that this approach can reduce overhead and that quality may depend on prompt structure, document position in context, and batch construction (Cheng et al., 2023; Fan et al., 2023; Feng et al., 2024; Ji et al., 2025; Lin et al., 2024). However, for text classification tasks, the literature does not offer a comparison of batching strategies framed explic-

itly as a quality–cost trade-off in which both constraints — the number of documents and the total batch length in tokens — are considered jointly, while document grouping and ordering are treated as separate design axes.

In this work, we formulate batch prompting for text classification as a multi-objective optimization problem that jointly maximizes classification quality and minimizes total token consumption. Using binary, multiclass, and multilabel classification tasks, we compare batching strategies under both batching controls and evaluate them through Pareto front analysis, hypervolume comparison, pairwise non-domination, and regime-specific operating-point selection. Rather than asking only whether batching helps, we compare strategies at both the frontier level and the operating-point level, and we additionally examine realized batching behavior and the separate contributions of document grouping and document ordering.

Our contributions are as follows:

- we propose and empirically investigate a batch-prompting scheme for text classification with two constraints: a maximum number of documents per batch and a maximum total batch size in tokens;
- we propose a framework for comparing batching strategies in terms of the quality–cost trade-off, combining Pareto front analysis, hypervolume comparison, pairwise non-domination relations, and the selection of locally preferred operating-points;
- we show that the strongest fronts and the preferred operating-points depend both on the dataset and on whether the target regime is low-cost, balanced, or quality-oriented;
- we show that document grouping is the more stable design axis, whereas document ordering also makes a meaningful contribution, but does so less consistently and with stronger dependence on task context.

2 Related Work

2.1 Batch Prompting

Prior works on batch prompting show that shared prompt components can be amortized across multiple inputs to reduce token cost, but that quality may degrade as batch size grows (Cheng et al., 2023; Feng et al., 2024; Larionov and Eger, 2025; Lin et al., 2024). Lin et al. (2024) further show that performance depends not only on how many documents are batched, but also on their position within the batch. Subsequent work explores both prompt-side and batch construction-side responses to this trade-off. Lin et al. (2024) propose BatchPrompt, which aggregates predictions across multiple within-batch permutations and uses early stopping, while Feng et al. (2024) reuse generated outputs as demonstrations for later items in the same batch. On the batch-construction side, Fan et al. (2023) show that cost-effective batching depends not only on document packing but also on the choice of demonstration examples, and Ji et al. (2025) formalize the resulting cost–quality optimization problem. Taken together, these studies establish batch prompting as a promising cost-reduction technique whose effectiveness depends on batch size, document position within a batch, and batch construction.

2.2 Position, Composition, and Order in Batches

The relevance of within-batch order is supported by research on long-context use. Peysakhovich and Lerer (2023) and Liu et al. (2024) show that model performance depends on where relevant information appears in the context, with information near the beginning or end often used more reliably than information in the middle. Hsieh et al. (2024b) relate this pattern to positional attention bias, and long-context benchmarks likewise show that effective context use may degrade as inputs become longer (Bai et al., 2024; Hsieh et al., 2024a; Zhang et al., 2024). This makes document order within a batch a plausible design variable rather than a neutral formatting choice.

The relevance of batch composition is supported by adjacent work on in-context example selection and ordering. Demonstration-based prompting studies show that model outputs depend on which examples are chosen and how they are arranged (Brown et al., 2020; Min et al., 2022; Rubin et al., 2022; Wang et al., 2024). Related work also shows that performance can be sensitive to the order of the same examples and that order selection can itself be optimized (Lu et al., 2022; Xiang et al., 2024; Xu et al., 2024). In batch-oriented settings, these considerations extend naturally to how documents are grouped together and ordered within the shared context (Fan et al., 2023; Ji et al., 2025; Lin et al., 2024).

However, for text classification tasks, the literature still does not provide a comparison of batching

strategies framed explicitly as a quality–cost trade-off in which both controls — the maximum number of documents and the maximum total batch length in tokens — are considered jointly, while document grouping and document ordering are treated as separate design axes. That is the gap addressed in this paper.

3 Problem Formulation

Let

$$D = \{d_1, \ldots, d_N\}$$

be a set of documents, and let

$$l(x) = |\operatorname{tok}(x)|$$

denote the length of text x in tokens under a fixed tokenizer. Let W be the model context-window size, P the fixed part of the prompt, $l_P = l(P)$ its length, r the number of tokens reserved for the model output, and o the per-document overhead within a batch such as document wrappers, separators, indexing markers, and other repeated formatting tokens.

A batching configuration is

$$\mathcal{B} = (B_1, \ldots, B_J),$$

where each batch B_j is an ordered sequence

$$B_j = (c_1, \ldots, c_{k_j}), \qquad c_p \in D \, \forall p \in \{1, \ldots, k_j\}.$$

Here k_j is the number of documents in batch B_j and c_p is the corpus document at position p in batch B_j. Thus, $\mathcal{B}$ forms an ordered partition of D: each document belongs to exactly one batch, and each batch specifies an order of its documents.

The length of batch B_j, including per-document overhead, is

$$l_j = \sum_{p=1}^{k_j} \big(l(c_p) + o\big),$$

and the corresponding input length of the model call is

$$t_j^{\text{in}} = l_P + l_j.$$

Batch formation is governed by two separate controllable constraints:

$$k_j \le k_{\max}, \tag{1}$$

$$l_j \le l_{\max}. \tag{2}$$

In addition, each batch must satisfy the context-window constraint under the reserved output budget:

$$t_j^{\text{in}} + r \le W. \tag{3}$$

After processing batch B_j, the model generates t_j^{out} output tokens, where

$$0 \le t_j^{\text{out}} \le r.$$

The total token consumption of batch B_j is

$$t_j = t_j^{\text{in}} + t_j^{\text{out}},$$

and the total token consumption over the dataset is

$$T(\mathcal{B}) = \sum_{j=1}^{J} t_j.$$

After all documents have been processed, the dataset-level quality metric is

$$Q(\mathcal{B}) \in [0, 1].$$

For a fixed batching algorithm A, the output configuration is

$$\mathcal{B} = A(D; k_{\max}, l_{\max}),$$

that is, an ordered partition of D satisfying the stated constraints. Selecting batching parameters can therefore formulated as a bi-objective optimization problem that maximizes $Q(\mathcal{B})$ and minimizes $T(\mathcal{B})$.

By evaluating different values of $(k_{\max}, l_{\max})$ for algorithm A, we obtain the set

$$\mathcal{Z}_A = \left\{ \big(Q(\mathcal{B}), T(\mathcal{B})\big) \;\middle|\; \mathcal{B} = A(D; k_{\max}, l_{\max}) \right\}.$$

The non-dominated points in $\mathcal{Z}_A$ form the Pareto front of algorithm A.

4 Methodology

4.1 Experimental setup and search over batching controls

For a fixed dataset, fixed prompt and model setup, batching strategy A, and nominal controls $(k_{\max}, l_{\max})$, one evaluated configuration is produced by computing the per-document features required by A (token length and, where needed, embeddings or difficulty scores), constructing an ordered partition

$$\mathcal{B} = A(D; k_{\max}, l_{\max})$$

subject to (1)–(3), submitting the resulting batches sequentially to the LLM with a fixed task-specific prompt template, and aggregating dataset-level quality $Q(\mathcal{B})$ and total token consumption $T(\mathcal{B})$. Thus, each evaluated point is defined by a dataset, a fixed model setup, a batching strategy, and one nominal parameter setting $(k_{\max}, l_{\max})$.

For each strategy A, we perform multi-objective search over

$$k_{\max} \in \{1, \ldots, k_{\max}^{\mathrm{up}}\}, \qquad l_{\max} \in [l_{\max}^{\mathrm{down}}, l_{\max}^{\mathrm{up}}],$$

with the objectives of maximizing classification quality Q and minimizing total token consumption T. The search is carried out with Optuna (Akiba et al., 2019) in multi-objective mode. Because the count constraint and token-budget constraint can interact asymmetrically, a large nominal $k_{\max}$ may become effectively inactive when the token budget binds earlier. To reduce the search space devoted to such configurations, we use the conditional upper bound

$$k_{\max}^{\mathrm{cond}}(l_{\max}) = \left\lfloor \frac{l_{\max}}{l_{\mathrm{ref}}} \right\rfloor,$$

where l_{ref} is a conservative lower estimate of document length, taken as the fifth percentile of the empirical document-length distribution in the corresponding dataset. Hence,

$$k_{\max} \in \{1, \ldots, \min(k_{\max}^{\mathrm{up}}, k_{\max}^{\mathrm{cond}}(l_{\max}))\}.$$

In addition to the two objective values, each evaluated configuration retains realized batching descriptors that are used later to interpret selected operating-points.

4.2 Batching strategy design space

We treat a batching strategy as a pair of components: a *grouping rule* (G-...), which determines the documents assigned to a given batch, and an *ordering rule* (O-...), which determines the order of documents within an already formed batch. This factorization is central to the comparison framework, because the same nominal batching controls can yield different quality–cost trade-offs depending on both batch composition and within-batch arrangement.

We include two reference strategies. **Best-Fit Decreasing (G-BFD) and Random ordering (O-R)** is a token-oriented packing baseline that attempts to use the available budget as fully as possible under the feasibility constraints. **BatchPrompt with SEAS early stopping (G-BP) and Identity ordering (O-I)** (Lin et al., 2024) serves as a literature-grounded baseline in which documents are evaluated under multiple within-batch permutations, predictions are aggregated across runs, and high-confidence cases may be removed from later rounds.

Beyond these references, the compared grouping families span three types of batch construction: length-based grouping (**G-LA**); semantic grouping, including semantically homogeneous, semantically diverse, and hybrid semantic-plus-length variants (**G-SH**, **G-SD**, **G-SHLA**); and difficulty-aware grouping, including homogeneous, deliberately mixed, and routing-based variants (**G-DH**, **G-DD**, **G-DR**). Within-batch ordering is treated as an independent axis. We use random order as the neutral reference (**O-R**); identity order (**O-I**) only for the BatchPrompt baseline; edge-based policies that place longer or more difficult documents near batch boundaries (**O-LE**, **O-DE**); and composition-oriented policies that encourage local difficulty diversity or semantic homogeneity/diversity (**O-DD**, **O-SH**, **O-SD**).

4.3 Data, model, and evaluation framework

We evaluate the methods on three text classification datasets chosen to vary in both label structure and document-length profile: the **ISOT Fake News Dataset** (Ahmed et al., 2018) for binary classification, **Yahoo! Answers** (Zhang et al., 2015) for multiclass topic classification, and **GoEmotions** (Demszky et al., 2020) for multilabel emotion classification. Among them, Fake News contains the longest documents, GoEmotions the shortest, and Yahoo! Answers lies between them.

For semantic grouping and ordering, each document d_i is represented by an embedding

$$\mathbf{e}_i = E(x_i) \in \mathbb{R}^m,$$

where x_i is the dataset-specific informative text of the example: `title + text` for Fake News, `title + content + answer` for Yahoo! Answers, and `text` for GoEmotions. We use `Qwen3-Embedding-8B` (Zhang et al., 2025) and cosine similarity. For difficulty-aware strategies, each document is assigned a scalar score

$$u(d_i) \in \mathbb{R},$$

used only for ranking and batching; in the present experiments, this score is based on perplexity.

All experiments are conducted with a single locally deployed generative model, `Qwen3-4B-Instruct-2507` (Yang et al., 2025). Fixing one model keeps the comparison controlled and isolates the contribution of batching design from model-to-model variation.

An anonymized repository containing the code, prompts, and configuration files used in this work is provided in Appendix A.[1]

The two primary evaluation metrics are *quality*, measured by Macro-F1 over the full dataset $Q(\mathcal{B})$, and *cost*, measured by total token consumption $T(\mathcal{B})$. To interpret how nominal controls are realized in practice, we additionally record the effective batch size k_{eff} (reported as the median realized number of documents per batch), fill rate, and the share of batches in which the token-budget constraint is binding. These quantities are descriptive rather than optimization objectives. For compact summaries, we further map the token-bound share to a realized regime label: *count-bound* if $\%\text{token_bound} \leq 0.2$, *token-bound* if $\%\text{token_bound} \geq 0.8$, and *mixed* otherwise.

For each dataset and batching strategy, we construct the Pareto front from the full set of evaluated configurations in the objective space $(Q \uparrow,\ T \downarrow)$. A configuration belongs to the front if it is non-dominated with respect to higher Macro-F1 and lower total token cost. Our primary whole-front comparison criterion is hypervolume. To make hypervolume comparable across strategies within the same dataset, objective values are normalized against shared dataset-wide ranges, and hypervolume is computed in minimization space against the common reference point

$$\mathrm{r}_{\text{point}} = (1+\delta,\ 1+\delta), \qquad \delta = 0.05.$$

Because hypervolume summarizes overall frontier strength but not head-to-head robustness, we also compute a pairwise retained-share criterion. For every pair of strategies (A_1, A_2) within the same dataset, we pool their frontier points, recompute non-dominated points in the union, and measure

$$s(A_1, A_2) = \frac{|\mathrm{ND}(\mathcal{F}_{A_1} \cup \mathcal{F}_{A_2}) \cap \mathcal{F}_{A_1}|}{|\mathcal{F}_{A_1}|},$$

where $\mathcal{F}_A$ is the frontier of strategy A and $\mathrm{ND}(\cdot)$ denotes the non-dominated subset. At the strategy level, we use the average retained share across pairwise comparisons within a dataset.

[1] https://anonymous.4open.science/r/acl_2026-1A60.

Whole-front comparison and practical recommendation address different questions. To obtain locally selectable configurations, we partition the evaluated configurations of each dataset into three budget regions using dataset-specific quantiles of total token cost: **low-budget**, **balanced**, and **quality-oriented**, corresponding to the lower, middle, and upper thirds of the observed token-cost distribution. Within each region, we select one operating-point by the following rule: first maximize Macro-F1, then break ties by lower total token consumption. These points are selected from all evaluated configurations in the region rather than only from reconstructed frontier points. Each selected batched point is then compared with the non-batched single-document baseline, in which one document is submitted per LLM call under the same dataset and prompt setting. For each selected point, we report the quality difference

$$\Delta Q = Q_{\text{batched}} - Q_{\text{single}}$$

and the token ratio

$$\rho_T = \frac{T_{\text{batched}}}{T_{\text{single}}},$$

together with the corresponding token savings. The nominal controls $(k_{\max}, l_{\max})$ are interpreted jointly with the realized descriptors k_{eff}, fill rate, and token-bound share.

4.4 Grouping-versus-ordering ablation

To separate the contributions of grouping and ordering, we perform two complementary component-level ablations. In the grouping ablation, we fix an ordering family and compare all compatible grouping families by the hypervolume of the frontier produced by each grouping–ordering combination. In the ordering ablation, we fix a grouping family and compare all compatible ordering families in the same way.

For each family and dataset, we then compute two summary statistics: *mean hypervolume*, defined as the average frontier hypervolume across all fixed-counterpart comparisons, and *stability*, defined as the share of fixed-counterpart settings in which that family attains the highest hypervolume. This yields separate family-level summaries for grouping and ordering. The factorized ablation excludes the BatchPrompt grouping family (G-BP) and the Identity ordering family (O-I), because these two components are used only in combination with each other and therefore do not admit a clean comparison with the remaining families.

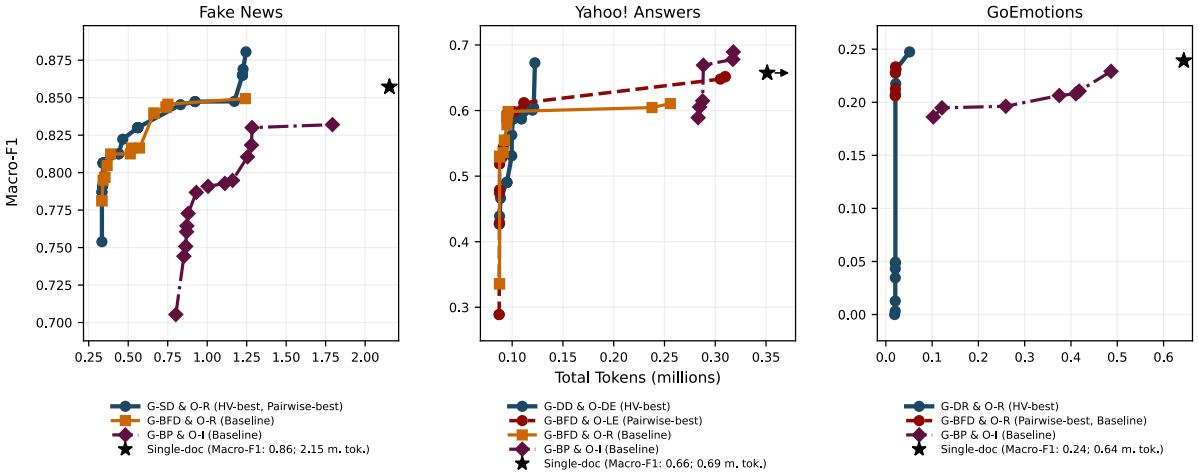

Figure 1: Pareto front comparison across Fake News, Yahoo! Answers, and GoEmotions. Each panel shows the hypervolume-best frontier, the pairwise-best frontier, and reference frontiers. Black stars indicate the single-document baseline.

5 Results

Figure 1 shows that no universal best batching strategy emerges at the frontier level. The strongest trade-off surfaces differ across datasets in both geometry and algorithmic composition. For Fake News, the upper frontier is most consistently associated with semantically diverse grouping (G-SD), indicating that stronger whole-front results come from more careful batch composition rather than from aggressive amortization alone. GoEmotions exhibits a different profile: strong frontier points concentrate in the low-token-cost region, so much of the attainable quality is already available under highly economical batching. Yahoo! Answers is the most heterogeneous case. Here the strongest frontiers vary more across the trade-off space, and agreement between hypervolume and pairwise retained-share views is weaker than on Fake News. Complete Pareto frontiers are provided in Appendix B, and detailed top 10 frontier rankings by hypervolume and pairwise retained share are reported in Appendix Tables C1 and C2.

Table 1 provides the information on local operating-points selection question. The resulting recommendations are again dataset-dependent. For Fake News, the quality-oriented point improves on the single-document baseline by $+0.023$ Macro-F1 while using only 57.8% of the baseline token budget; the balanced and low-budget points trade some quality for sharper cost reduction, down to 30.8% and 17.7% of baseline cost. For Yahoo! Answers, the balanced and quality-oriented points both outperform the baseline, by $+0.015$ and $+0.032$, while using 17.7% and 46.0% of the baseline token budget. GoEmotions shows the strongest amortization effect: the low-budget and balanced points use only 3.5% and 6.6% of baseline tokens while matching or exceeding baseline quality, and the quality-oriented point still improves Macro-F1 by $+0.011$ at 52.4% of baseline cost. Thus, for all three datasets, at least one selected batched regime matches or exceeds single-document prompting while reducing total token consumption.

The selected points also show why nominal controls should be interpreted together with realized batching behavior. On Fake News, the strongest quality-oriented solution reverts to micro-batches ($k = 2$) and remains far from token-limited, so the upper part of the trade-off is driven more by careful composition than by aggressive packing. GoEmotions also uses $k = 2$ in the quality-oriented regime, but in economical regimes it scales to much larger realized batches, reaching $k = 240$ and $k_{\text{eff}} = 200$, which is consistent with its much stronger amortization gains. Yahoo! Answers differs in mechanism as well as in frontier shape: its strong points are more often token-bound, with token-bound shares of 88.9% in the low-budget regime and 87.5% in the quality-oriented regime, so realized behavior is constrained more by total batch length than by document count. Table 1 and Appendix Table D1 therefore show that the same nominal two-control formulation can yield markedly different realized regimes across datasets, helping explain why batching pays off through careful composition on Fake News, aggressive amortization on GoEmotions, and more

Dataset	Region	Algorithm	Macro-F1	Tokens	ΔQ vs. SD	Tokens (% of SD)	k	t	k_{eff}	Fill	% token-bound	Regime
Fake News	Low budget	G-SD & O-SD	0.824	381156	-0.033	17.7%	20	14400	20	74.9%	0.0%	count-bound
	Balanced	G-BFD & O-R	0.840	663138	-0.018	30.8%	5	7400	5	36.8%	1.0%	count-bound
	High quality	G-SD & O-R	0.881	1245396	+0.023	57.8%	2	1600	2	66.9%	4.2%	count-bound
Yahoo! Answers	Low budget	G-BFD & O-DD	0.593	93618	-0.064	13.6%	152	9200	52	91.6%	88.9%	token-bound
	Balanced	G-DD & O-DE	0.673	122393	+0.015	17.7%	17	3600	17	63.8%	24.2%	mixed
	High quality	G-BP & O-I	0.689	317807	+0.032	46.0%	181	9800	54	96.8%	87.5%	token-bound
GoEmotions	Low budget	G-SD & O-R	0.240	22341	+0.000	3.5%	240	5600	200	78.0%	66.7%	mixed
	Balanced	G-DR & O-DE	0.244	42733	+0.005	6.6%	27	1600	27	43.1%	5.3%	count-bound
	High quality	G-BFD & O-DE	0.250	337219	+0.011	52.4%	2	400	2	12.8%	0.0%	count-bound

Table 1: Selected operating-points in the low-budget, balanced, and quality-oriented regimes, with comparison to the single-document baseline (SD) and realized batching behavior.

criterion-sensitive trade-offs on Yahoo! Answers.

6 Discussion

The results show that there is no universal strategy for batch prompting, because dataset properties affect both the geometry of the quality–cost frontier and the mechanism by which batching yields gains. Fake News dataset favors more careful batch composition and conservative high-quality solutions, GoEmotions permits much stronger amortization at competitive quality, and Yahoo! Answers is the most criterion-sensitive case, with stronger divergence between globally strong frontiers and locally preferred operating-points. The main point is therefore not simply that different algorithms win on different datasets, but that batching behaves as a dataset-dependent optimization problem whose strong solutions arise through different trade-off structures.

This dataset dependence also clarifies why frontier-level comparison and operating-point recommendation should be kept distinct. Hypervolume characterizes how well a strategy covers the frontier as a whole, pairwise retained share emphasizes robustness in head-to-head comparison, and regime-specific operating-point selection answers the local practical question of which configuration is preferable under a given budget region. These views are therefore complementary rather than interchangeable: a strategy may define a strong global frontier without yielding the best low-budget or quality-oriented recommendation, especially when the trade-off space is heterogeneous. Treating them separately is what allows us to distinguish overall frontier strength from actionable regime-specific choice.

The selected points further show that nominal controls $(k_{\max}, l_{\max})$ are not self-interpreting without realized batching behavior. Effective batch size, fill rate, and token-bound incidence explain whether a solution achieves its gains through cautious composition or aggressive amortization, and the same control formulation can therefore induce different operating regimes across datasets. For practical use, this suggests a regime-first tuning procedure: first determine whether the goal is low-budget, balanced, or quality-oriented use; and only then treat one strategy as preferable. In particular, quality-oriented use is more reliably associated with micro-batches or otherwise conservative realized batching ($k_{\max} < 5$–10), balanced use may allow a slight increase in the document number ($k_{\max} < 20$–30), and low-budget use may justify much larger batches when the data support it.

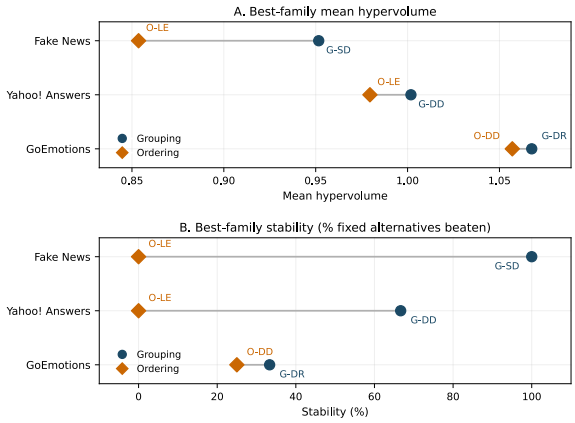

Figure 2: Grouping vs. Ordering ablation results.

The ablation in Figure 2 examines the two strategy components separately: how documents are grouped into batches and how they are ordered within a batch. For each dataset, Panel A compares the strongest grouping family and the strongest ordering family by mean frontier hypervolume, while Panel B compares the same families by stability, measured as the share of fixed-counterpart settings in which that family attains the highest hypervolume. The strongest grouping family typically shows both a clearer mean-hypervolume advantage and a stronger stability advantage when the ordering component is varied, indicating that

grouping more often determines the shape of strong frontier solutions. Ordering should not be treated as negligible, because the best ordering family does change across datasets and can improve the resulting frontier, but its effect is weaker and less consistently dominant once grouping is fixed. The most careful interpretation is therefore that grouping and ordering are separable but not equally important design dimensions: grouping is the more stable and interpretable axis, whereas ordering acts more as a conditional amplifier whose usefulness depends on the chosen grouping family, the dataset, and the frontier region of interest. In practical terms, this means that when tuning resources are limited, it is more reliable to identify a strong grouping strategy first and refine within-batch ordering second. For more detailed information on the grouping vs. ordering comparison see the Appendix Tables E1 and E2.

7 Limitations

This study evaluates batch prompting with a single generative model, `Qwen3-4B-Instruct-2507`. The reported trade-offs should therefore not be assumed to transfer automatically to other LLM families, and the present evidence is limited to the studied prompt-based classification setting. The experiments also do not include reasoning-oriented models, so they do not establish whether the same batching patterns would persist in settings that require more explicit multi-step reasoning.

The evaluation further covers only three datasets. Although these datasets span binary, multiclass, and multilabel classification settings, the results should not be read as a universal characterization of batching effects across text classification tasks more broadly. In addition, because the datasets vary along multiple dimensions at once, the comparison does not isolate the contribution of any single factor, such as label structure, text length, or domain, to the observed quality–cost trade-offs.

8 Conclusion

This study formulated batch prompting for text classification as a multi-objective optimization problem that jointly considers predictive quality and token cost under two batching controls: the maximum number of documents per batch and the maximum total batch size in tokens. Across binary, multiclass, and multilabel classification tasks, the results show that the effectiveness of batching is determined not by batch size alone, but by the interaction between batching constraints, document composition, and within-batch order. The quality–cost trade-off therefore has to be understood as a property of the batching strategy as a whole rather than as a simple consequence of increasing or decreasing the number of documents processed together.

Empirically, no universally optimal batching strategy emerged. The strongest Pareto frontiers and the most attractive operating-points varied across datasets and across low-cost, balanced, and quality-oriented regimes. For Fake News, the upper frontier was most consistently associated with semantically diverse grouping, indicating that high-quality solutions depend more on careful batch composition than on aggressive amortization. For GoEmotions, batching was especially efficient: selected points matched or exceeded the single-document baseline while using only a small fraction of its token budget. Yahoo! Answers showed the most heterogeneous trade-off structure, with the strongest global frontier, the most robust pairwise frontier, and the best local operating-points not always coming from the same strategy family. Taken together, these results show that frontier-level comparison and local operating-point selection address different but complementary questions.

The component-level ablation further showed that document grouping is the more stable design axis, whereas document ordering also affects performance but in a less consistent and more context-dependent way. This distinction is methodologically important because it clarifies which aspects of batching strategy most reliably shape the observed trade-offs. Within the scope of the studied model and datasets, the results position batch prompting not merely as a token-saving heuristic, but as a structured optimization problem in which quality, cost, and batching design must be considered jointly.

References

Hadeer Ahmed, Issa Traore, and Sherif Saad. 2018. Detecting opinion spams and fake news using text classification. *SECURITY AND PRIVACY*, 1(1):e9.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2623–2631, New York, NY, USA. Association for Computing Machinery.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. LongBench: A bilingual, multi-task benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Zhoujun Cheng, Jungo Kasai, and Tao Yu. 2023. Batch prompting: Efficient inference with large language model APIs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 792–810, Singapore.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. 2020. GoEmotions: A dataset of fine-grained emotions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4040–4054, Online.

Aleksandra Edwards and Jose Camacho-Collados. 2024. Language models for text classification: Is in-context learning enough? In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 10058–10072, Torino, Italia. ELRA and ICCL.

Meihao Fan, Xiaoyue Han, Ju Fan, Chengliang Chai, Nan Tang, Guoliang Li, and Xiaoyong Du. 2023. Cost-effective in-context learning for entity resolution: A design space exploration. *Preprint*, arXiv:2312.03987.

Longyu Feng, Mengze Hong, and Chen Jason Zhang. 2024. Auto-demo prompting: Leveraging generated outputs as demonstrations for enhanced batch prompting. *Preprint*, arXiv:2410.01724.

Yuting Guo, Anthony Ovadje, Mohammed Ali Al-Garadi, and Abeed Sarker. 2024. Evaluating large language models for health-related text classification tasks with public social media data. *Preprint*, arXiv:2403.19031.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024a. Ruler: What's the real context size of your long-context language models? *Preprint*, arXiv:2404.06654.

Cheng-Yu Hsieh, Yung-Sung Chuang, Chun-Liang Li, Zifeng Wang, Long Le, Abhishek Kumar, James Glass, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, and Tomas Pfister. 2024b. Found in the middle: Calibrating positional attention bias improves long context utilization. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14982–14995, Bangkok, Thailand.

Zhaoxuan Ji, Xinlu Wang, Zhaojing Luo, Zhongle Xie, and Meihui Zhang. 2025. Optimized batch prompting for cost-effective llms. *Proc. VLDB Endow.*, 18(7):2172–2184.

Daniil Larionov and Steffen Eger. 2025. Batchgemba: Token-efficient machine translation evaluation with batched prompting and prompt compression. *Preprint*, arXiv:2503.02756.

Jianzhe Lin, Maurice Diesendruck, Liang Du, and Robin Abraham. 2024. Batchprompt: Accomplish more with less. In *International Conference on Learning Representations*, volume 2024, pages 21590–21612.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland.

Marcus Ma, Georgios Chochlakis, Niyantha Maruthu Pandiyan, Jesse Thomason, and Shrikanth Narayanan. 2025. Large language models do multi-label classification differently. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 2472–2495, Suzhou, China.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates.

Alexander Peysakhovich and Adam Lerer. 2023. Attention sorting combats recency bias in long context language models. *Preprint*, arXiv:2310.01427.

Md Mezbaur Rahman and Cornelia Caragea. 2025. LLM-guided co-training for text classification. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 31104–31121, Suzhou, China. Association for Computational Linguistics.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of*

the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2655–2671, Seattle, United States.

Muhammad Irfan Fikri Sabri, Hansi Hettiarachchi, and Tharindu Ranasinghe. 2025. Detecting fake news in the era of language models. In *Proceedings of the 15th International Conference on Recent Advances in Natural Language Processing - Natural Language Processing in the Generative AI Era*, pages 1036–1043, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

Xiaofei Sun, Xiaoya Li, Jiwei Li, Fei Wu, Shangwei Guo, Tianwei Zhang, and Guoyin Wang. 2023. Text classification via large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8990–9005, Singapore.

Liang Wang, Nan Yang, and Furu Wei. 2024. Learning to retrieve in-context examples for large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1752–1767, St. Julian's, Malta.

Yanzheng Xiang, Hanqi Yan, Lin Gui, and Yulan He. 2024. Addressing order sensitivity of in-context demonstration examples in causal language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6467–6481, Bangkok, Thailand.

Zhichao Xu, Daniel Cohen, Bei Wang, and Vivek Srikumar. 2024. In-context example ordering guided by label distributions. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2623–2640, Mexico City, Mexico.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 649–657, Cambridge, MA, USA. MIT Press.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. 2024. ∞Bench: Extending long context evaluation beyond 100K tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277, Bangkok, Thailand.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *Preprint*, arXiv:2506.05176.

A Code repository

Anonymous repository. `https://anonymous.4open.science/r/acl_2026-1A60`. This repository includes prompt templates, batching-strategy implementations, and config files used to generate the data for this work. We acknowledge the use of OpenAI Codex[2] in the implementation of our research. All final code is verified by the authors.

B Full Pareto Front Atlases

The full Pareto front atlases presented in this appendix provide a whole-front view of the quality–cost trade-off. Each figure shows the full Pareto front atlas for one dataset in the quality–cost space, with Macro-F1 on the quality axis and total token consumption on the cost axis. The black star denotes the single-document baseline. Because its token cost lies outside the plotted range, it is indicated with an arrow.

[2] `https://openai.com/codex/`

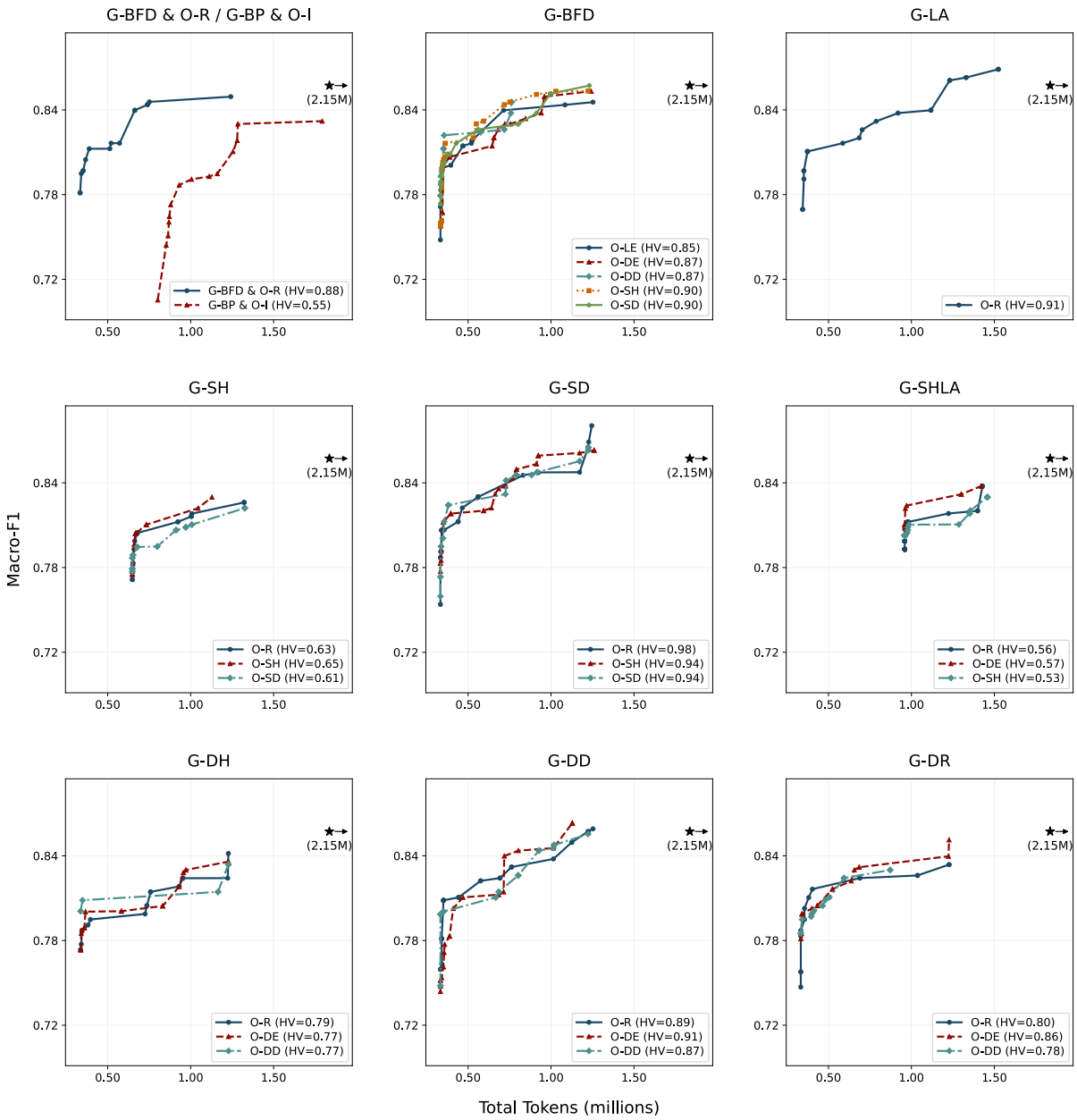

Figure B1: Full Pareto front atlas for the Fake News dataset.

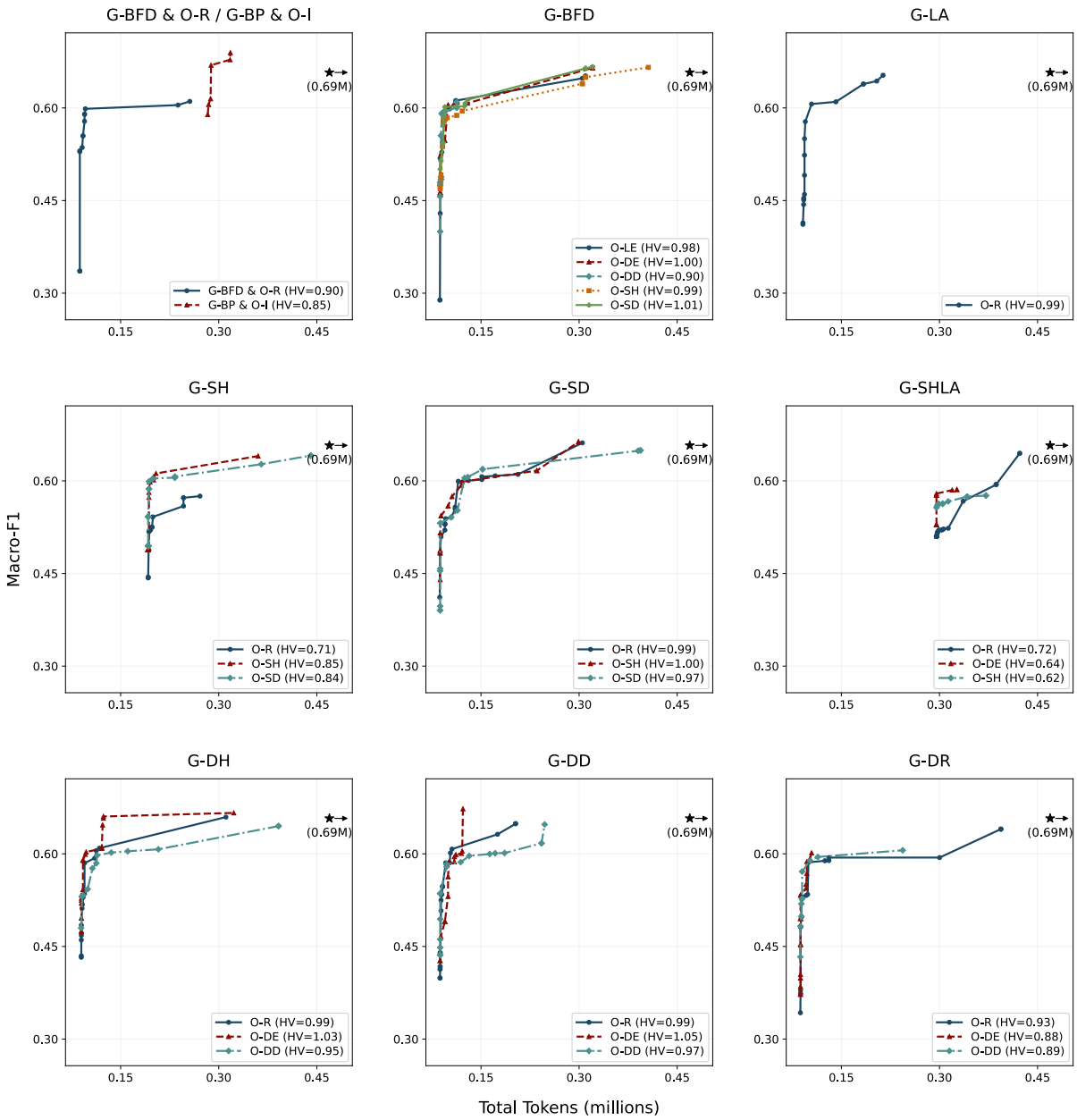

Figure B2: Full Pareto front atlas for the Yahoo! Answers dataset.

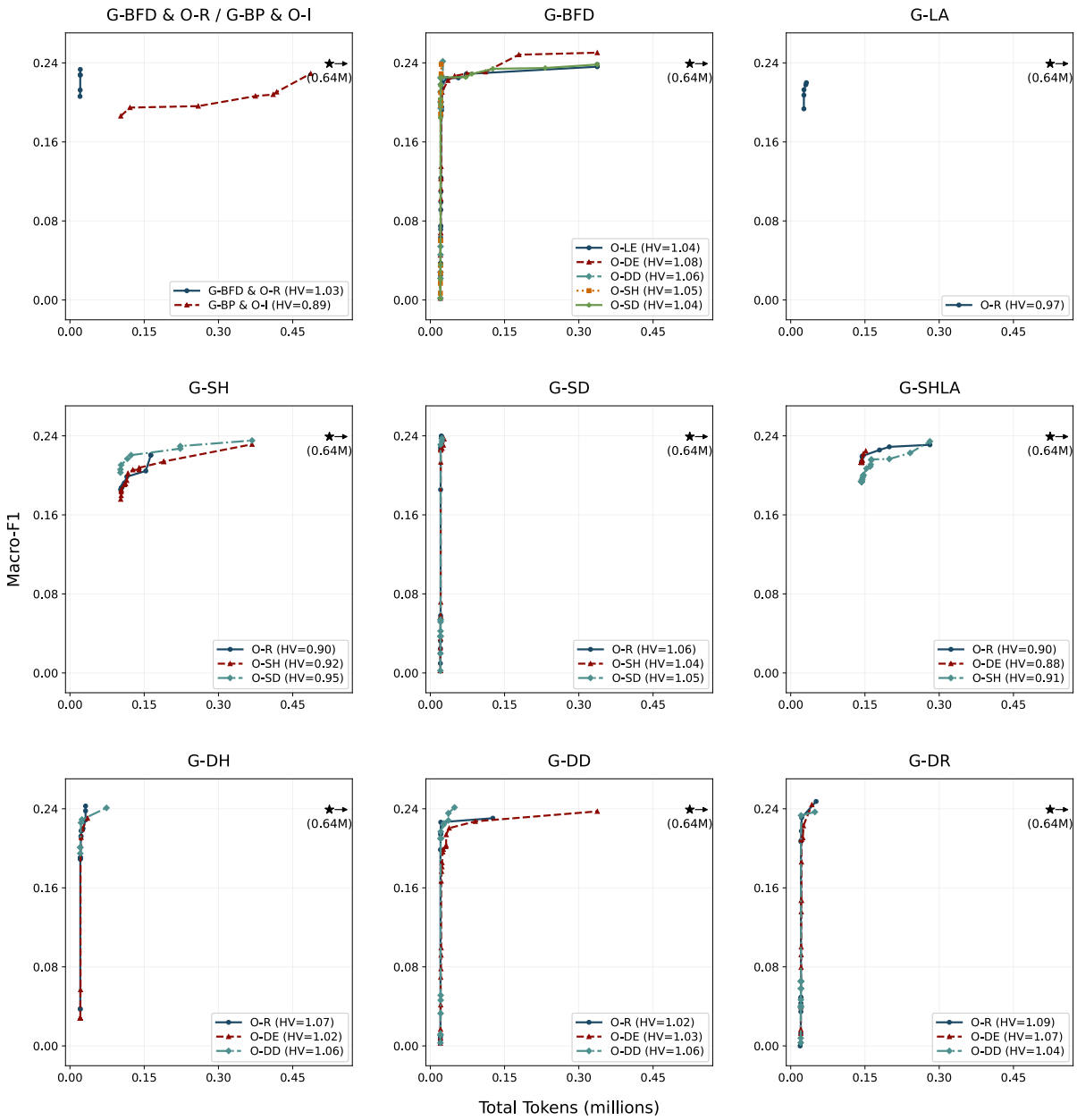

Figure B3: Full Pareto front atlas for the GoEmotions dataset.

C Frontier Ranking Tables

Dataset	Frontier	HV
Fake News	G-SD & O-R	0.979
	G-SD & O-SH	0.938
	G-SD & O-SD	0.937
	G-LA & O-R	0.911
	G-DD & O-DE	0.910
	G-BFD & O-SH	0.903
	G-BFD & O-SD	0.898
	G-DD & O-R	0.892
	G-BFD & O-R	0.878
	G-BFD & O-DE	0.874
Yahoo! Answers	G-DD & O-DE	1.048
	G-DH & O-DE	1.031
	G-BFD & O-SD	1.008
	G-BFD & O-DE	0.999
	G-SD & O-SH	0.996
	G-SD & O-R	0.993
	G-BFD & O-SH	0.991
	G-LA & O-R	0.991
	G-DH & O-R	0.989
	G-DD & O-R	0.986
GoEmotions	G-DR & O-R	1.087
	G-BFD & O-DE	1.085
	G-DR & O-DE	1.072
	G-DH & O-R	1.068
	G-BFD & O-DD	1.064
	G-DD & O-DD	1.062
	G-DH & O-DD	1.059
	G-SD & O-R	1.056
	G-BFD & O-SH	1.051
	G-SD & O-SD	1.045

Table C1: Top 10 frontiers for each dataset, ranked by common-reference hypervolume.

Dataset	Frontier	Avg. retained share
Fake News	G-SD & O-R	0.914
	G-SD & O-SD	0.895
	G-SD & O-SH	0.885
	G-BFD & O-DD	0.878
	G-BFD & O-SD	0.832
	G-BFD & O-R	0.810
	G-BFD & O-SH	0.780
	G-DR & O-DE	0.764
	G-DD & O-R	0.718
	G-DD & O-DD	0.707
Yahoo! Answers	G-BFD & O-LE	0.877
	G-BFD & O-SD	0.871
	G-DD & O-R	0.838
	G-BFD & O-DD	0.809
	G-DR & O-DE	0.807
	G-SD & O-SH	0.784
	G-DH & O-DE	0.774
	G-BFD & O-R	0.766
	G-DR & O-DD	0.748
	G-BFD & O-DE	0.732
GoEmotions	G-BFD & O-R	1.000
	G-SD & O-R	0.900
	G-DR & O-R	0.893
	G-SD & O-SD	0.886
	G-DR & O-DD	0.871
	G-DD & O-R	0.813
	G-BFD & O-SH	0.790
	G-SD & O-SH	0.774
	G-BFD & O-DD	0.755
	G-BFD & O-SD	0.740

Table C2: Top 10 frontiers for each dataset, ranked by average pairwise retained share.

D Nominal Selections and Realized Batching Profiles

This appendix provides a view of how frontier-relevant operating-points are realized as batching configurations. The table is formed from the union of the top three frontiers by common-reference hypervolume and the top three frontiers by average pairwise retained share. For each retained algorithm within each dataset and budget region, it reports the best observed nominal controls together with the corresponding realized batching profile.

The Near-optimal trials column denotes the number of evaluated trials within a fixed dataset–algorithm–budget region block whose Macro-F1 lies within 0.005 of the blockwise best value, and the Near-optimal share column denotes their proportion among all trials in that block. These columns indicate whether the reported nominal setting is isolated or belongs to a broader near-optimal neighborhood.

For the Fake News dataset, fewer frontiers are retained because the two ranking criteria overlap completely, whereas for Yahoo! Answers and GoEmotions they overlap only partially.

Dataset	Region	Algorithm	Best k	Best t	Best k_eff	Best Fill	Best Tok.-bound	Best Regime	Near-opt. trials	Near-opt. share
Fake News	Low budget	G-SD & O-R	32	15200	28	97.1%	94.7%	token-bound	6	12.8%
	Low budget	G-SD & O-SD	20	14400	20	74.9%	0.0%	count-bound	2	4.4%
	Low budget	G-SD & O-SH	34	16000	29	97.4%	88.9%	token-bound	3	6.5%
	Balanced	G-SD & O-R	12	3800	7	98.4%	98.7%	token-bound	4	11.1%
	Balanced	G-SD & O-SD	16	4200	8	98.2%	98.5%	token-bound	6	16.7%
	Balanced	G-SD & O-SH	5	2800	5	91.9%	22.0%	mixed	2	5.6%
	High quality	G-SD & O-R	2	1600	2	66.9%	4.2%	count-bound	1	5.9%
	High quality	G-SD & O-SD	2	2000	2	54.6%	0.8%	count-bound	3	15.8%
	High quality	G-SD & O-SH	2	1400	2	75.3%	7.1%	count-bound	7	38.9%
Yahoo! Answers	Low budget	G-BFD & O-LE	85	22200	85	48.8%	14.3%	count-bound	2	4.8%
	Low budget	G-BFD & O-SD	126	10200	59	93.0%	75.0%	mixed	1	2.1%
	Low budget	G-DD & O-DE	115	7800	48	97.2%	90.0%	token-bound	3	6.5%
	Low budget	G-DD & O-R	144	11800	75	91.8%	85.7%	token-bound	3	5.5%
	Low budget	G-DH & O-DE	198	14600	60	64.9%	87.5%	token-bound	6	13.0%
	Balanced	G-BFD & O-LE	58	3200	17	98.8%	95.8%	token-bound	3	6.0%
	Balanced	G-BFD & O-SD	36	2000	10	97.2%	92.3%	token-bound	2	4.5%
	Balanced	G-DD & O-DE	17	3600	17	63.8%	24.2%	mixed	1	2.0%
	Balanced	G-DD & O-R	14	1000	6	98.5%	94.8%	token-bound	2	5.0%
	Balanced	G-DH & O-DE	36	2400	12	93.0%	88.2%	token-bound	2	4.2%
	High quality	G-BFD & O-LE	5	400	2	100.3%	80.4%	token-bound	4	50.0%
	High quality	G-BFD & O-SD	4	400	2	95.8%	69.2%	mixed	5	62.5%
	High quality	G-DD & O-DE	4	400	3	100.3%	70.9%	mixed	1	20.0%
	High quality	G-DD & O-R	6	1000	6	75.9%	33.0%	mixed	2	40.0%
	High quality	G-DH & O-DE	4	400	2	94.8%	69.0%	mixed	1	16.7%
GoEmotions	Low budget	G-BFD & O-DE	383	9600	256	68.3%	50.0%	mixed	2	6.9%
	Low budget	G-BFD & O-R	311	20000	256	32.8%	0.0%	count-bound	2	3.0%
	Low budget	G-DR & O-DE	372	7800	256	84.1%	50.0%	mixed	5	14.3%
	Low budget	G-DR & O-R	248	20200	248	21.6%	33.3%	mixed	2	4.1%
	Low budget	G-SD & O-R	224	5600	200	78.0%	66.7%	mixed	3	4.9%
	Balanced	G-BFD & O-DE	22	6000	22	9.1%	0.0%	count-bound	2	4.0%
	Balanced	G-BFD & O-R	72	2400	72	68.3%	25.0%	mixed	1	3.7%
	Balanced	G-DR & O-DE	27	1600	27	43.1%	5.3%	count-bound	1	1.8%
	Balanced	G-DR & O-R	23	800	23	71.3%	13.0%	count-bound	1	2.3%
	Balanced	G-SD & O-R	133	13600	133	24.1%	0.0%	count-bound	1	2.9%
	High quality	G-BFD & O-DE	2	400	2	12.8%	0.0%	count-bound	4	19.0%
	High quality	G-BFD & O-R	21	18600	21	2.8%	0.0%	count-bound	1	16.7%
	High quality	G-DR & O-DE	2	400	2	12.8%	0.0%	count-bound	2	20.0%
	High quality	G-DR & O-R	20	800	20	63.0%	7.7%	count-bound	1	12.5%
	High quality	G-SD & O-R	8	400	8	51.2%	0.0%	count-bound	1	25.0%

Table D1: Selected operating-points with nominal batching controls (k, t) and realized batching profiles. For each dataset and budget region, the table reports the selected algorithm, nominal batching settings, realized effective batch size, fill level, token-bound share, and realized regime.

E Grouping-versus-Ordering Ablation Details

Dataset	Grouping family	Mean HV	% fixed-ordering wins	n orderings
Fake News	G-SD	0.952	100.0%	3
	G-LA	0.911	0.0%	1
	G-DD	0.890	33.3%	3
	G-BFD	0.879	33.3%	6
	G-DR	0.813	0.0%	3
	G-DH	0.776	0.0%	3
	G-SH	0.632	0.0%	3
	G-SHLA	0.552	0.0%	3
Yahoo! Answers	G-DD	1.002	66.7%	3
	G-LA	0.991	0.0%	1
	G-DH	0.990	0.0%	3
	G-SD	0.985	66.7%	3
	G-BFD	0.962	33.3%	6
	G-DR	0.899	0.0%	3
	G-SH	0.801	0.0%	3
	G-SHLA	0.663	0.0%	3
GoEmotions	G-DR	1.068	33.3%	3
	G-BFD	1.052	66.7%	6
	G-SD	1.048	33.3%	3
	G-DH	1.048	0.0%	3
	G-DD	1.036	0.0%	3
	G-LA	0.968	0.0%	1
	G-SH	0.922	0.0%	3
	G-SHLA	0.896	0.0%	3

Table E1: Grouping-family ablation summary by dataset. For each grouping family, this table reports the mean frontier hypervolume across all compatible fixed-ordering comparisons, the percentage of fixed-ordering settings in which that family attains the highest hypervolume, and the number of ordering families considered.

Dataset	Ordering family	Mean HV	% fixed-grouping wins	n groupings
Fake News	O-LE	0.854	0.0%	1
	O-DD	0.822	0.0%	4
	O-SD	0.814	0.0%	3
	O-R	0.804	37.5%	8
	O-DE	0.798	60.0%	5
	O-SH	0.757	50.0%	4
Yahoo! Answers	O-LE	0.980	0.0%	1
	O-SD	0.939	33.3%	3
	O-DD	0.926	0.0%	4
	O-DE	0.920	40.0%	5
	O-R	0.903	37.5%	8
	O-SH	0.864	50.0%	4
GoEmotions	O-DD	1.057	25.0%	4
	O-LE	1.037	0.0%	1
	O-DE	1.017	20.0%	5
	O-SD	1.012	33.3%	3
	O-R	1.003	50.0%	8
	O-SH	0.981	25.0%	4

Table E2: Ordering-family ablation summary by dataset. For each ordering family, this table reports the mean frontier hypervolume across all compatible fixed-grouping comparisons, the percentage of fixed-grouping settings in which that family attains the highest hypervolume, and the number of grouping families considered.