# TOKENFLOW: INFERENCE-FREE TOKEN SELECTION WITH DIVERSITY PRESERVATION FOR MODEL FINE-TUNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Supervised fine-tuning (SFT) is critical for aligning large language models (LLMs), yet its efficiency is hindered by the uneven utility of tokens in instruction datasets. Existing token-level selection methods rely on reference models or auxiliary inference, introducing significant computational overhead. In this paper, we propose TokenFlow, a plug-and-play, inference-free framework for token-efficient SFT. TokenFlow exploits intrinsic signals from the standard SFT forward pass and consists of two synergistic modules that determine which tokens to optimize and how to optimize them: **MINT** (**M**aximizing **IN**struction-aware **T**oken utility) and **Tempered Cross-Entropy (TCE)**. MINT reformulates token selection as an instruction-aware subset utility maximization problem. Unlike existing methods that independently rank tokens, MINT considers token interactions and subset-level utility to select complementary token subsets under a fixed budget. TCE preserves response diversity by incorporating semantically valid alternatives from the model's predictive distribution while maintaining correctness supervision. Extensive experiments across diverse LLM families demonstrate the effectiveness of TokenFlow. Training on only 60% of the tokens, TokenFlow outperforms full-dataset fine-tuning by 2.60 points on the OpenLLM Leaderboard and consistently surpasses state-of-the-art selection baselines.

## 1 INTRODUCTION

Supervised fine-tuning (SFT) is a critical step for aligning large language models (LLMs) with human intent (Sun et al., 2024; Li et al., 2024; Chang et al., 2024). However, real-world instruction datasets inevitably contain tokens with widely varying utility, ranging from highly informative to trivially redundant or noisy. Training indiscriminately on all tokens is inefficient and can degrade model performance, highlighting the need for fine-grained, token-level data selection (Lin et al., 2025; Chai et al., 2023).

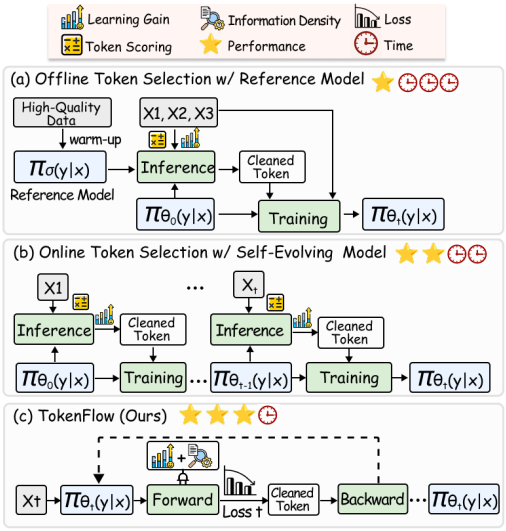
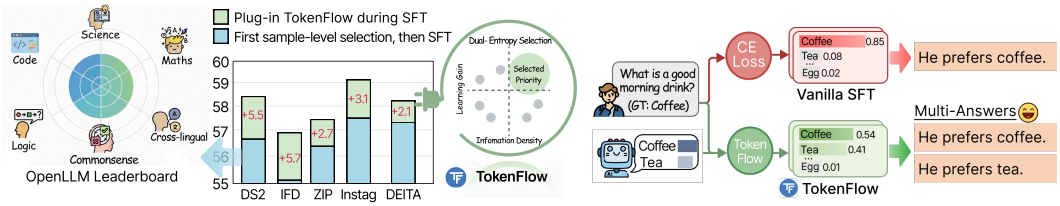



Figure 1: Comparison of token selection methods.

Given this context, recent studies (Lin et al., 2024; Pang et al., 2025) primarily rely on reference models to calculate token utility. As shown in Figure 1(a), they employ a pre-trained, static reference model to score tokens before training. This paradigm is fundamentally limited: it incurs substantial computational overhead via extra training and inference steps, and the static reference model cannot adapt to the target model's evolving learning state. In response, self-evolving methods (Figure 1(b)) attempt to mitigate this static nature by using the model itself for scoring. However, they require



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.




**(a)** Plug-and-play Performance. **(b)** An Example of Token Diversity.

Figure 2: Overview of TokenFlow: plug-and-play gains (+2.3 avg) on top of existing sample-level selection methods (DS2/IFD/ZIP/Instag/DEITA), evaluated on the OpenLLM Leaderboard.

extra inference steps, which incur unacceptable computation overhead Pang et al. (2025). This raises a natural **research question**: *Can we score tokens on-the-fly during training, without any additional inference passes or external reference models, while ensuring both high quality and efficiency?*

In this work, we posit that the answer is yes. Predictive uncertainty and attention-based semantics are two widely adopted signals for assessing token utility (Liu et al., 2026; Lin et al., 2025). The former indicates learning potential, while the latter reflects instruction grounding. Existing methods obtain these signals through expensive external models or auxiliary inference. **Our key observation** is that the standard SFT forward pass *itself* naturally produces both signals as inherent by-products, without additional inference. As shown in Figure 1(c), during standard SFT, the model first computes decoder *self-attention weights* to capture token dependencies. It then produces *token-level logits* and predictive distributions, which are used to compute the SFT loss and update model parameters via backpropagation. The self-attention weights provide instruction-aware relevance signals by capturing the dependency between response tokens and instruction tokens, while the token representations derived from the forward pass characterize semantic similarity among response tokens. Together, these complementary signals enable an *inference-free* token utility function. However, effectively utilizing them remains ***challenging***.

**First**, *how can we identify high-utility tokens using these intrinsic signals?* Existing token selection methods typically estimate the utility of each token independently, which overlooks the interactions among selected tokens. In instruction fine-tuning, a valuable token should not only provide informative supervision by itself, but also complement other selected tokens to form a diverse and effective training subset. Therefore, it is challenging to design a token utility function that jointly captures individual token importance and subset-level diversity.

**Second**, *how can we train on a selected subset of tokens without losing response diversity?* Standard cross-entropy (CE) loss forces the model to assign all probability mass to a single ground-truth token. For example, consider the instruction *"What is a good morning drink?"* with the target *"Coffee"* (shown in Figure 2) (b). A capable model might initially assign high probability to both *"Coffee"* and *"Tea"*, as both are semantically valid. However, since the dataset only encodes *"Coffee"*, standard CE loss aggressively penalizes *"Tea"*. This sharp one-hot supervision eliminates valid alternatives and inevitably causes a collapse in generation diversity.

**Our Methodology and Contributions.** To address these challenges, we propose **TokenFlow**, a *plug-and-play* framework for inference-free token-efficient supervised fine-tuning. TokenFlow addresses the two challenges through two synergistic modules.

To tackle the first challenge, we introduce **MINT** (**M**aximizing **IN**struction-aware **T**oken utility), an inference-free token subset selection module. Different from existing methods that independently estimate token utility and rank tokens individually, MINT reformulates token selection as an instruction-aware subset utility maximization problem. It jointly models (i) *instruction-aware token utility*, which estimates the individual contribution of each response token based on its relevance to the given instruction, and (ii) *token similarity*, which captures interactions among response tokens and encourages a diverse and non-redundant selected subset. By maximizing the resulting subset utility under a fixed token budget, MINT identifies a globally valuable response token subset using intrinsic signals from the standard SFT forward pass.

To address the second challenge, we propose **Tempered Cross-Entropy (TCE)**, a diversity-preserving objective for selected tokens. Instead of treating the annotated token as the only valid target, TCE incorporates high-probability plausible alternatives from the model's predictive distribu-

tion. This strategy preserves the supervised correctness signal while reducing the over-penalization of semantically valid continuations, thereby maintaining diverse response distributions during token-efficient training. Together, MINT and TCE transform TokenFlow into a plug-and-play training framework that seamlessly integrates with standard SFT pipelines and existing data selection methods (see Figure 2(b)).

Furthermore, extensive experiments across diverse LLM families (LLaMA-3.1, Qwen-2, OLMo-2) demonstrate that TokenFlow, while training on only 60% of the tokens, outperforms full-token SFT by 2.78 points on the OpenLLM Leaderboard and consistently surpasses state-of-the-art data selection baselines.

## 2 Problem Formulation

### 2.1 Token Selection with Diversity Preservation for SFT

Given an instruction-response pair $(x, y)$, where $y = \{y_1, \cdots, y_T\}$ denotes the response tokens, token selection introduces a binary mask $m_t \in \{0, 1\}$ into the SFT objective:

$$\mathcal{L}(m, \theta) = -\mathbb{E}_{(x,y)\sim\mathcal{D}} \left[ \sum_{t=1}^{T} m_t \ell_\theta(y_t | x, y_{<t}) \right]. \tag{1}$$

Existing methods determine the mask based on a token utility function $U(\cdot)$, typically requiring an external reference model $\theta_{\text{ref}}$:

$$m_t = \mathbb{I}\left(U(y_t; \theta, \theta_{\text{ref}}) > \tau\right). \tag{2}$$

Therefore, token-efficient SFT involves two key designs: (1) a **token utility function** $U(\cdot)$ to identify valuable tokens, and (2) a **token-level optimization objective** $\ell_\theta(\cdot)$ to optimize selected tokens while preserving response diversity. However, reliance on $\theta_{\text{ref}}$ introduces additional training and inference costs for reference model optimization and utility estimation.

### 2.2 Inference-Free Token Selection with Diversity Preservation

To eliminate the dependency on external models, we formulate inference-free token selection by deriving utility directly from intrinsic signals produced by the current model $\theta$ during standard training. Specifically, the selection mask is defined as:

$$m_t^{(k)} = \mathbb{I}\left(f(z_t^{(k)}; \theta^{(k)}) > \tau\right), \tag{3}$$

where $z_t^{(k)}$ denotes training-time signals of token $y_t$ at step $k$, such as logits and attention patterns. This formulation enables $U(y_t) \approx f(z_t^{(k)}; \theta^{(k)})$, achieving inference-free selection without reference model optimization or additional forward passes.

Beyond selecting valuable tokens, the optimization objective should further preserve the diversity of learned response distributions. Therefore, our goal is to jointly design an inference-free token utility function $U(\cdot)$ and a diversity-preserving token-level objective $\ell_\theta(\cdot)$ for efficient SFT.

## 3 TokenFlow

In this section, we introduce the overall design of TokenFlow, a plug-and-play framework for inference-free token-efficient supervised fine-tuning. As shown in Figure 3, TokenFlow consists of two main modules: a *MINT token subset utility maximization* module and a *Tempered Cross-Entropy* objective. To determine *which tokens should be optimized*, MINT (**M**aximizing **IN**struction-aware **T**oken utility) reformulates token selection as an instruction-aware token subset utility maximization problem, which selects valuable response tokens under a fixed token budget (Section 3.1). To determine *how the selected tokens should be optimized*, the *Tempered Cross-Entropy* objective further improves token-level supervision by preserving diverse supervision signals while maintaining the correctness of the original supervision (Section 3.2).

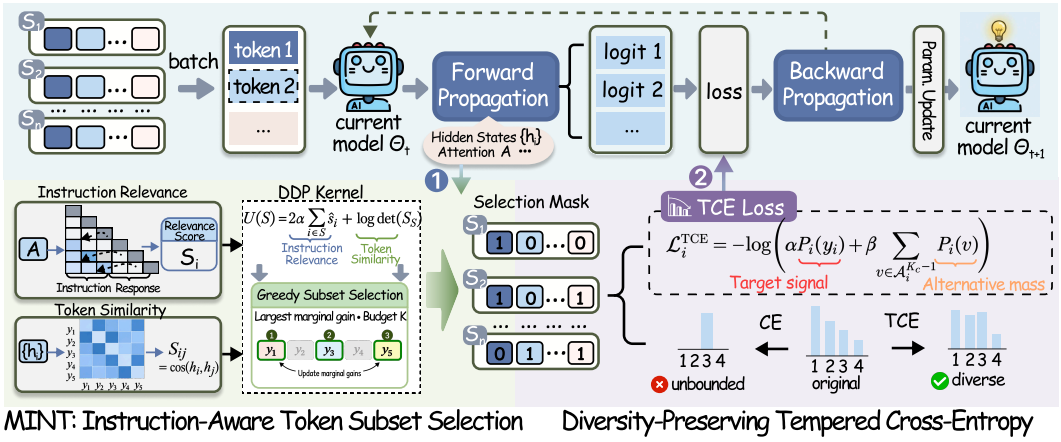



Figure 3: Overview of the TokenFlow framework. **Left**: MINT performs instruction-aware token subset selection. It estimates token utility by jointly considering instruction relevance and token similarity, and employs a DPP-based subset optimization to select diverse and informative response tokens under a limited budget. **Right**: Selected tokens are optimized with Diversity-preserving Tempered Cross-Entropy (TCE), which preserves the target supervision signal while redistributing part of the probability mass to plausible alternatives, maintaining response diversity during SFT.

### 3.1 MINT: Maximizing INstruction-aware Token Utility

Existing token selection methods typically estimate the utility of each response token independently and select tokens according to their individual scores. However, independently ranking tokens ignores the interactions among selected tokens, where multiple tokens may provide overlapping supervision signals. To address this limitation, we introduce MINT, which reformulates token selection as an instruction-aware utility maximization problem. Instead of optimizing individual token utility, MINT aims to identify the most valuable response token subset under a limited optimization budget.

#### 3.1.1 From Individual Token Utility to Subset Utility Maximization

Given an instruction-response pair $(x, y)$, where $x$ denotes the instruction and $y = \{y_1, \ldots, y_{L_r}\}$ denotes the response tokens, existing token selection methods generally define an individual utility score for each response token and select the top-$K$ tokens.

$$u_i = U(y_i), \qquad \mathcal{S} = \mathrm{TopK}(\{u_i\}_{i=1}^{L_r}), \tag{4}$$

where $\mathcal{S}$ represents the selected token subset.

Such a formulation implicitly assumes that the utility of each token is independent of other selected tokens. However, the contribution of a token may depend on which other tokens are included in the subset. For example, multiple tokens with high individual utility may provide similar supervision signals, while a combination of complementary tokens can offer higher overall training value. Therefore, we reformulate token selection as a subset-level utility maximization problem:

$$\mathcal{S}^* = \arg \max_{\mathcal{S} \subseteq Y, |\mathcal{S}| = K} U(\mathcal{S}), \tag{5}$$

where $U(\mathcal{S})$ denotes the overall utility of the selected token subset.

#### 3.1.2 Token Subset Utility Modeling with Determinantal Point Processes

To model the utility of a selected token subset, we adopt Determinantal Point Processes (DPPs) (Kulesza & Taskar, 2012), which provide a principled framework for modeling the joint utility of a set of elements. Given a response token set $Y = \{y_1, \ldots, y_{L_r}\}$, DPP measures the utility of a subset $\mathcal{S} \subseteq Y$ through a positive semi-definite kernel matrix $L \in \mathbb{R}^{L_r \times L_r}$, where the subset utility is determined by the determinant of its corresponding kernel submatrix:

$$U(\mathcal{S}) = \det(L_{\mathcal{S}}),$$

where $L_{\mathcal{S}}$ denotes the principal submatrix of $L$ corresponding to the selected token subset $\mathcal{S}$.

Following the DPP formulation, selecting the most valuable token subset can be formulated as a maximum a posteriori (MAP) inference problem:

$$\mathcal{S}^* = \arg \max_{\mathcal{S} \subseteq Y, |\mathcal{S}|=K} \det(L_{\mathcal{S}}), \tag{6}$$

To construct the kernel matrix, we model the pairwise relationships among response tokens based on their hidden representations. Specifically, for each response token $y_i$, we extract its causal hidden representation $h_i$ used for predicting $y_i$ and normalize it before computing the cosine similarity:

$$\tilde{h}_i = \frac{h_i}{\|h_i\|_2}, \qquad L_{ij} = \tilde{h}_i^\top \tilde{h}_j = \frac{h_i^\top h_j}{\|h_i\|_2 \|h_j\|_2}. \tag{7}$$

### 3.1.3 Instruction-aware Token Utility Maximization

Although DPP provides a general formulation for token subset utility optimization, the kernel matrix constructed in Eq. (7) only captures the interactions among response tokens. However, for instruction fine-tuning, the utility of a response token should not only depend on its interaction with other tokens, but also on whether it is relevant to the given instruction.

Therefore, we further incorporate instruction-aware token utility into the DPP kernel. To achieve this, we first introduce an instruction relevance estimation method to measure the alignment between response tokens and instruction tokens. Then, we incorporate the estimated instruction relevance into the DPP-based token subset utility maximization objective.

**Instruction Relevance Utility Estimation.** Given an instruction $x = \{x_1, \ldots, x_{L_p}\}$ and a response $y = \{y_1, \ldots, y_{L_r}\}$, we extract the response-to-instruction attention matrix from the decoder self-attention layers $A \in \mathbb{R}^{L_r \times L_p}$, where $L_p$ and $L_r$ denote the numbers of instruction and response tokens, respectively. Each element $A_{ij}$ represents the attention weight from the prediction position of response token $y_i$ to instruction token $x_j$. Therefore, each row of $A$ captures the dependency of a response token on the instruction context. A direct aggregation of each row of $A$ assumes that all instruction tokens contribute equally. However, different instruction tokens may contain different levels of task-relevant information. Therefore, we first estimate the importance of instruction tokens and propagate such importance to response tokens:

$$w_p = A^T \mathbf{1}_{L_r}, \quad u = A w_p, \quad s = \log(u + \epsilon), \tag{8}$$

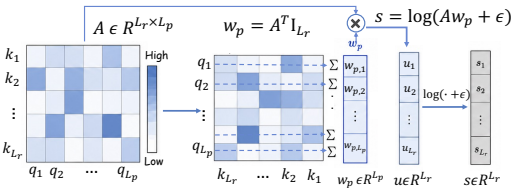



Figure 4: Instruction Relevance Estimation.

where $\mathbf{1}_{L_r} \in \mathbb{R}^{L_r}$ is an all-one vector. The resulting vector $w_p \in \mathbb{R}^{L_p}$ represents the importance of instruction tokens, where larger values indicate that the corresponding instruction tokens receive more attention from response tokens. Then, $u \in \mathbb{R}^{L_r}$ measures the relevance of each response token to the important instruction components. Finally, $s \in \mathbb{R}^{L_r}$ denotes the instruction relevance score after logarithmic transformation.

To reduce scale variations across samples, we normalize the relevance scores as $\hat{s}_i = \mathrm{Norm}(s_i)$, where $\hat{s}_i$ denotes the normalized instruction relevance of response token $y_i$. The normalized scores are then used as the individual utility term in the diversity-aware subset objective.

**Instruction-aware Utility Maximization.** The above relevance score only reflects the degree of association between response tokens and instructions. To optimize token subset utility, we first transform the instruction relevance scores into non-negative quality weights:

$$q_i = \exp(\gamma \hat{s}_i), \qquad \tilde{L}_{ij} = q_i S_{ij} q_j, \tag{9}$$

where $\gamma > 0$ controls the contribution of instruction relevance, and $q_i$ reflects the individual utility of response token $y_i$ with respect to the given instruction. Following the DPP formulation, $\tilde{L}$ denotes the IRD kernel matrix, where $S_{ij}$ measures the semantic similarity between response tokens $y_i$ and $y_j$. Therefore, the kernel jointly captures individual token utility and the interactions among selected tokens.

Based on the DPP objective, we define the subset-level token utility as:

$$U(\mathcal{S}) = \log\det(\tilde{L_{\mathcal{S}}}) = 2\alpha \sum_{i\in\mathcal{S}} \hat{s}_i + \log\det(S_{\mathcal{S}}). \tag{10}$$

Finally, given the token budget $K$, MINT selects the optimal token subset by maximizing the instruction-aware token utility:

$$\mathcal{S}^* = \arg\max_{\mathcal{S}\subseteq Y, |\mathcal{S}|=K} 2\alpha \sum_{i\in\mathcal{S}} \hat{s}_i + \log\det(S_{\mathcal{S}}), \tag{11}$$

#### 3.1.4 Token Subset Utility Optimization and Selection

The objective in Eq. (11) aims to identify the token subset with the maximum instruction-aware utility under a fixed token budget. However, directly solving the DPP-based MAP inference problem is NP-hard in general. Therefore, MINT adopts a greedy optimization strategy to efficiently approximate the optimal token subset. Formally, starting from an empty token subset $\mathcal{S}_0 = \emptyset$, MINT iteratively selects the token that provides the largest marginal utility gain:

$$y^* = \arg\max_{y_i\notin\mathcal{S}_t} \left[U(\mathcal{S}_t \cup \{y_i\}) - U(\mathcal{S}_t)\right], \tag{12}$$

where $\mathcal{S}_t$ denotes the selected token subset after the $t$-th iteration. The selected token is added to the current subset and the final subset is converted into a token-level selection mask:

$$\mathcal{S}_{t+1} = \mathcal{S}_t \cup \{y^*\}, \qquad m_i = \mathbb{I}(y_i \in \mathcal{S}^*). \tag{13}$$

where $m_i = 1$ indicates that response token $y_i$ is retained as an effective training signal, while $m_i = 0$ indicates that it is excluded from subsequent optimization. The generated token mask is finally incorporated into the token-level training objective described in Section 3.2.

### 3.2 Diversity-preserving Tempered Cross-Entropy

After obtaining the MINT-based token selection mask $m = (m_1, \ldots, m_{L_r})$ from Section 3.1.4, MINT identifies response tokens with high supervision utility. However, token selection only determines *which* tokens to optimize, while the training objective determines *how* they are optimized. Therefore, we further introduce Tempered Cross-Entropy (TCE) to preserve response diversity during optimization.

Specifically, for each response position $i$, we construct a candidate set $\mathcal{A}_i^{K_c-1} = \mathrm{Top}_{K_c-1}\{P_i(v) \mid v \in \mathcal{V}, v \neq y_i\}$ of plausible non-annotated tokens. The TCE objective is defined as

$$\mathcal{L}_i^{\mathrm{TCE}} = -\log\left(\alpha P_i(y_i) + \beta P_i(\mathcal{A}_i)\right), \tag{14}$$

where $P_i(\mathcal{A}_i) = \sum_{v\in\mathcal{A}_i} P_i(v)$ and $\alpha, \beta$ balance the annotated token and alternative tokens. The annotated term maintains the original supervision signal, while the alternative term relaxes excessive penalties on semantically plausible predictions.

Combining IRD-based token selection and TCE-based supervision calibration, the final token-efficient SFT objective is formulated as:

$$\mathcal{L} = -\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\sum_{i=1}^{L_r} m_i \mathcal{L}_i^{TCE}\right]. \tag{15}$$

Here, $m_i$ determines which response tokens participate in optimization, while TCE controls how the selected tokens are supervised. Therefore, MINT and TCE jointly improve token-efficient SFT by allocating optimization resources to informative tokens while preserving diverse response distributions.

## 4 Experiments

### 4.1 Experimental Setups

**Datasets.** Following the setup in (Pang et al., 2025), we construct our experimental data pool by sampling 50k examples from a collection of five widely used SFT datasets (300$K$ samples in total): Flan v2, OpenAssistant, Stanford Alpaca, Dolly, and WizardLM.

Table 1: Performance comparison of different token selection methods across models and benchmarks.

| Method | HellaSwag | BoolQ | GSM8K | TyDiQA | LogiQA | TriviaQA | ARC-C | OpenBookQA | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| *Base model: LLaMA-3.1-8B* | | | | | | | | | |
| Base | 59.92 | 82.18 | 50.31 | 22.80 | 26.51 | 61.38 | 52.97 | 41.80 | 49.73 |
| Full Tokens | 60.57 | 83.11 | 54.13 | 48.21 | 28.06 | 62.78 | 56.07 | 45.45 | 54.80 |
| Random | 60.14 | 82.80 | 51.98 | 49.83 | 27.91 | 63.34 | 56.33 | 43.64 | 54.50 |
| Rho-1 | 60.46 | 83.57 | 54.21 | 47.53 | 28.06 | 63.46 | 55.90 | 45.66 | 54.86 |
| TokenCleaning | 61.92 | 82.71 | 54.13 | 54.86 | 27.60 | 63.60 | 54.26 | 46.46 | 55.69 |
| DFT | 60.81 | 82.92 | 54.74 | 50.97 | 26.51 | 62.80 | 54.61 | 45.86 | 54.90 |
| EFT | 60.21 | 83.17 | 55.05 | 49.33 | 28.06 | 58.69 | 55.12 | 45.66 | 54.41 |
| **TokenFlow (Ours)** | **66.48** | **85.79** | **55.90** | **58.72** | **29.30** | **67.13** | **57.19** | **47.27** | **58.47** |
| *Base model: Qwen-2-7B* | | | | | | | | | |
| Base | 59.38 | 84.81 | 76.72 | 50.20 | 30.72 | 50.03 | 49.78 | 44.44 | 55.76 |
| Full Tokens | 58.83 | 84.96 | **78.10** | 51.07 | 31.16 | **56.43** | **57.97** | 45.45 | 58.00 |
| Random | 57.02 | 83.13 | 76.88 | 51.85 | 31.32 | 52.81 | 56.93 | 45.45 | 56.92 |
| Rho-1 | 57.02 | 84.68 | 75.57 | 49.57 | 32.56 | 54.01 | 55.59 | 45.86 | 56.86 |
| TokenCleaning | 57.55 | 84.71 | 75.57 | 49.53 | 30.54 | 54.78 | 55.28 | **46.87** | 56.85 |
| DFT | 57.23 | 81.96 | 76.65 | 56.38 | 29.77 | 53.72 | 56.68 | 43.64 | 57.00 |
| EFT | 58.92 | 84.43 | 77.49 | 48.26 | 31.01 | 53.74 | 54.13 | 45.66 | 56.71 |
| **TokenFlow (Ours)** | **64.19** | **85.24** | 77.95 | **65.14** | **33.33** | 55.50 | 56.07 | 46.06 | **60.44** |
| *Base model: OLMo-2-7B* | | | | | | | | | |
| Base | 60.31 | 80.20 | 68.84 | 41.59 | 25.27 | 62.45 | 57.11 | 46.67 | 55.31 |
| Full Tokens | 60.54 | 83.45 | 68.53 | 44.87 | 25.89 | 62.25 | 59.00 | 46.06 | 56.32 |
| Random | 60.66 | 78.51 | 67.08 | 42.15 | **26.20** | 61.54 | 56.07 | 46.87 | 54.89 |
| Rho-1 | **61.26** | 82.61 | 71.29 | 45.26 | 24.50 | 61.99 | 57.36 | 46.46 | 56.34 |
| TokenCleaning | 61.23 | 81.96 | 70.60 | 50.46 | 24.19 | 61.73 | 58.31 | 46.87 | 56.92 |
| DFT | 60.37 | 81.90 | 66.85 | 43.74 | 24.81 | 62.53 | 56.07 | 45.66 | 55.24 |
| EFT | 60.70 | 81.59 | 68.91 | 43.49 | 25.58 | 57.94 | 58.05 | 45.86 | 55.27 |
| **TokenFlow (Ours)** | **63.93** | **83.69** | **71.67** | **59.74** | 25.89 | **62.62** | **59.87** | **47.88** | **59.41** |

**Benchmarks and Metrics.** To comprehensively evaluate the efficacy of TokenFlow, we evaluate TokenFlow on OpenLLM Leaderboards, including TyDiQA (Clark et al., 2020), HellaSwag (Zellers et al., 2019), BoolQ (Clark et al., 2019), GSM8K (Cobbe et al., 2021), TriviaQA, ARC-Challenge (ARC-C), OpenBookQA and LogiQA (Liu et al., 2020).

**Base Models.** We adopt four widely used open-source LLMs of varying sizes and families, including LLaMA-3.1-8B, Qwen-2-7B, and Olmo-2-7B. These models are fine-tuned on our curated data pool and compared against different token selection methods.

**Baselines.** For each model, we evaluate several strategies for comparison: (1) FULL-TOKEN FINE-TUNING, (2) UNIFORM RANDOM, (3) RHO-1 (Lin et al., 2024), (4) TOKENCLEANING (Pang et al., 2025), (5) ENTROPY-BASED TUNING (Wang et al., 2025), (6) DYNAMIC FINE-TUNING (Wu et al., 2025). Appendix A.1 shows more details.

**Implementation Details of TokenFlow.** We utilize Low-Rank Adaption (LoRA) Hu et al. (2022) for parameter-efficient fine-tuning. The maximum learning rate is set as $2 \times 10^{-5}$ with a linear decay schedule, and the batch size is 8. We also fix the maximum input sequence length to 2080. Models are trained for 1 epoch on 4 A800 GPUs.

### 4.2 EXP-1: OVERALL PERFORMANCE

We evaluate TokenFlow against state-of-the-art baselines across three models (LLaMA-3.1-8B, Qwen-2-7B, OLMo-2-7B) on eight OpenLLM benchmarks. Table 1 shows three key insights:

**(1) Superior to SOTA Baselines.** TokenFlow consistently outperforms all baselines across different LLMs, achieving substantial improvements over the strongest competing methods on all three models. The gains are especially pronounced on TydiQA, where TokenFlow provides consistent improvements, demonstrating its effectiveness in identifying highly informative tokens for fine-tuning.

**(2) Fewer Tokens, Big Gains.** TokenFlow achieves superior performance by training on only $\sim$60% of tokens. On LLaMA-3.1-8B, selecting 60% tokens achieves better average performance than FULL TOKENS trained on 100% tokens. This finding validates that token quality, rather than token quantity, determines fine-tuning effectiveness.

Table 2: Ablation study of different components of TokenFlow on LLaMA-3.1-8B.

| Method | HellaSwag | BoolQ | GSM8K | TyDiQA | LogiQA | TriviaQA | ARC-C | OpenBookQA | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| **TokenFlow (Ours)** | **66.48** | **85.79** | **55.90** | **58.72** | **29.30** | **67.13** | **57.19** | **47.27** | **58.47** |
| **w/o TCE** | 63.65 | 83.29 | 54.67 | 55.26 | 27.44 | 64.63 | 56.68 | 46.06 | 56.46 |
| **w/o MINT** | | | | | | | | | |
| Only Relevance | 64.67 | 84.06 | 54.29 | 57.04 | 27.29 | 57.94 | 54.87 | 45.05 | 55.65 |
| Only DPP | 63.79 | 83.48 | 54.06 | 55.26 | 28.37 | 61.54 | 55.26 | 46.06 | 55.98 |

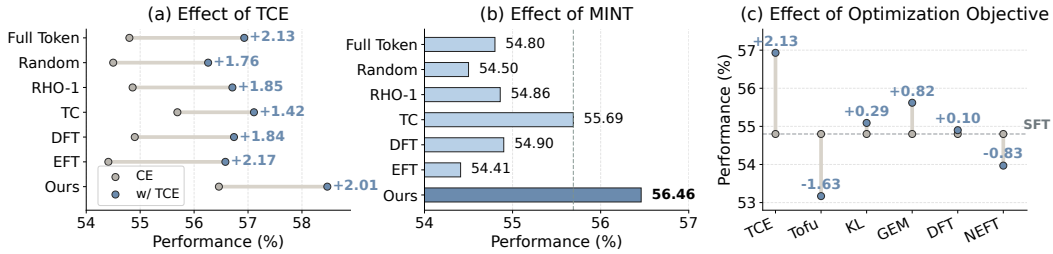



Figure 6: Analysis of Selection Strategy and Training Objective.

**(3) Consistent across LLMs.** TokenFlow generalizes across diverse architectures, consistently outperforming strong baselines on LLaMA-3.1-8B, Qwen-2-7B, and OLMo-2-7B. The consistency across models with different pre-training objectives and tokenization schemes indicates that intrinsic training signals capture universal properties of token utility rather than model-specific artifacts.

### 4.3 Exp-2: Ablation Study of TokenFlow

**Exp-2.1: Ablation Study of TokenFlow Components.** We evaluate the contribution of each component in TokenFlow, as shown in Table 2. The full model achieves the best average score of 58.47. Removing TCE decreases the average score to 56.46 ($-2.01$), demonstrating the importance of optimizing the supervision targets for selected tokens. For MINT, using only relevance or only DPP yields average scores of 55.65 ($-2.82$) and 55.98 ($-2.49$), respectively. These results show that relevance-based token utility and diversity-aware subset selection are complementary, and jointly modeling both is important for effective token selection.

**Exp-2.2: Ablation Study on the Token Selection Ratio.** We investigate the impact of different token selection ratios on TokenFlow. As shown in Figure 5, TokenFlow exhibits an inverted-U performance trend with respect to the selection ratio and achieves the best performance at a ratio of 0.6. This indicates that selecting an appropriate subset of response tokens is crucial: overly aggressive selection may remove complementary supervision signals, while retaining excessive tokens may introduce less informative signals and weaken the benefit of token-level optimization. The 0.6 ratio provides a favorable balance between token informativeness and coverage, enabling TokenFlow to achieve consistent improvements across benchmarks.

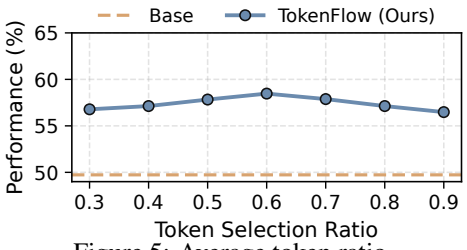



Figure 5: Average token ratio.

**Exp-2.3: The Effects of Selection Strategy and Training Objective.** We further analyze the effects of the selection strategy and optimization objective in TokenFlow, as shown in Figure 6. Figure 6(a) shows that replacing CE with TCE consistently improves different selection strategies, demonstrating the benefit of preserving plausible alternatives during optimization. Figure 6(b) verifies the effectiveness of MINT, which achieves the best performance by jointly modeling instruction-aware token quality and token similarity to select informative and complementary token subsets. Figure 6(c) further compares TCE with alternative optimization objectives, where TCE achieves the largest improvement over standard SFT, validating its effectiveness in maintaining response diversity while preserving correctness supervision.

### 4.4 EXP-3: THE EFFECIENCY OF TOKENFLOW

We evaluate the latency–performance trade-off of different token selection methods .As shown in Figure 7, TokenFlow achieves a strong balance between latency and performance, demonstrating the effectiveness of our SAIL utility function and TCE loss. This also validates TokenFlow 's training-free and inference-free design: it reuses intrinsic signals from the standard SFT forward pass without training a reference model or adding auxiliary inference. In contrast, TokenCleaning has the highest latency because it requires warm-up training for a reference model and performs token scoring with both the reference and base models, introducing two inference passes. Thus, TokenFlow provides a more practical efficiency–effectiveness trade-off.

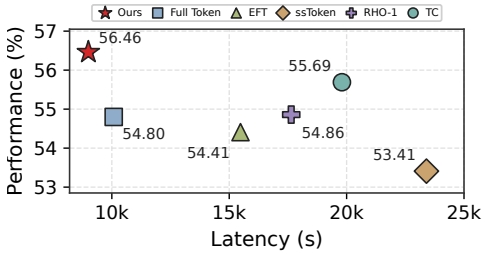



Figure 7: Average token ratio.

### 4.5 EXP-4: PLUG-AND-PLAY PERFORMANCE WITH SAMPLE SELECTION METHODS

We evaluate TokenFlow's plug-and-play compatibility using five representative sample-level selection methods (*e.g.,* DS2, IFD, ZIP, SelectIT, and DEITA). For each selector, we first obtain the selected data subset, then directly integrate TokenFlow during SFT on this subset without modifying any configurations of the sample selector. All experiments are conducted on LLaMA-3.1-8B and evaluated across seven OpenLLM benchmarks.

Table 3: Plug-and-play evaluation of TokenFlow on top of different sample selection methods.

| Selector | HS. | GSM | BoolQ | TydiQA | LogiQA | Avg. |
|---|---|---|---|---|---|---|
| DS2 | 60.57 | 54.13 | 83.11 | 48.21 | 28.06 | 54.82 |
| w/ Ours | 66.48 | 55.90 | 85.79 | 67.13 | 29.30 | **60.32** |
| IFD | 60.35 | 57.04 | 82.21 | 38.55 | 26.36 | 52.90 |
| w/ Ours | 64.10 | 57.35 | 82.40 | 63.02 | 26.20 | **58.61** |
| ZIP | 60.11 | 53.83 | 82.83 | 52.32 | 24.81 | 54.78 |
| w/ Ours | 62.79 | 60.18 | 83.51 | 55.25 | 25.75 | **57.50** |
| SelectIT | 60.21 | 56.37 | 83.29 | 44.44 | 27.94 | 54.45 |
| w/ Ours | 60.82 | 61.64 | 83.72 | 51.23 | 30.39 | **57.56** |
| DEITA | 64.74 | 57.35 | 82.86 | 44.81 | 30.08 | 55.97 |
| w/ Ours | 65.24 | 61.72 | 83.08 | 49.57 | 30.54 | **58.03** |

As shown in Table 3, TokenFlow consistently improves the average performance of all selectors, with gains of +5.5 for DS2, +5.71 for IFD, +2.72 for ZIP, +3.11 for SelectIT, and +2.06 for DEITA. These results show that TokenFlow provides complementary token-level refinement on top of sample-level selection, making it easy to integrate into existing data selection pipelines.

## 5 RELATED WORK

**Token-Level Data Selection for SFT.** Recent works (Xia et al., 2024; Zhou et al., 2023; Hanmo et al., 2024) explore finer-grained token selection along two main directions. *Loss-based methods*, such as RHO-1 (Lin et al., 2024) and TokenCleaning (Pang et al., 2025), score tokens via an "excess loss" computed against an external reference model, which incurs substantial overhead. *Entropy-based methods*, such as ProFit (Liu et al., 2026) and the 20/80 method (Wang et al., 2025), instead rely on the model's predictive uncertainty to identify learnable tokens. However, both lines depend on a one-dimensional signal that conflates valuable learning opportunities with noise. In contrast, TokenFlow synergistically combines uncertainty (for learning potential) and attention-based semantics (for information density) directly from the standard forward pass, achieving a robust and inference-free token utility assessment.

## 6 CONCLUSION

We introduced TokenFlow, an inference-free framework for token-efficient SFT that exploits intrinsic signals naturally produced during the standard SFT forward pass. TokenFlow jointly optimizes token selection and supervision calibration through MINT and Tempered Cross-Entropy, enabling informative token selection while preserving response diversity. Extensive experiments across multiple LLM families demonstrate that TokenFlow consistently outperforms existing token selection methods while training on only 60% of the tokens.

## AI USE STATEMENT

Generative AI tools were used for language polishing, grammar checking, and improving manuscript readability. The authors remain fully responsible for the final manuscript, experimental results, and released artifacts.

## REFERENCES


Chengliang Chai, Jiayi Wang, Yuyu Luo, Zeping Niu, and Guoliang Li. Data management for machine learning: A survey. *IEEE Trans. Knowl. Data Eng.*, 35(5):4646–4667, 2023.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Jonathan H Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. Tydi qa: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470, 2020.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

LIU Hanmo, DI Shimin, LI Haoyang, LI Shuangyin, CHEN Lei, and ZHOU Xiaofang. Effective data selection and replay for unsupervised continual learning. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 1449–1463. IEEE, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2-3):123–286, 2012.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7595–7628, 2024.

Xiaotian Lin, Yanlin Qi, Yizhang Zhu, Themis Palpanas, Chengliang Chai, Nan Tang, and Yuyu Luo. LEAD: iterative data selection for efficient LLM instruction tuning. *CoRR*, abs/2505.07437, 2025.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, Weizhu Chen, et al. Not all tokens are what you need for pretraining. *Advances in Neural Information Processing Systems*, 37:29029–29063, 2024.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*, 2020.

Tao Liu, Taiqiang Wu, Runming Yang, Shaoning Sun, Junjie Wang, and Yujiu Yang. Profit: Leveraging high-value signals in sft via probability-guided token selection. *arXiv preprint arXiv:2601.09195*, 2026.

Jinlong Pang, Na Di, Zhaowei Zhu, Jiaheng Wei, Hao Cheng, Chen Qian, and Yang Liu. Token cleaning: Fine-grained data selection for llm supervised fine-tuning. *arXiv preprint arXiv:2502.01968*, 2025.

Wangtao Sun, Haotian Xu, Xuanqing Yu, Pei Chen, Shizhu He, Jun Zhao, and Kang Liu. Itd: Large language models can teach themselves induction through deduction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2719–2731, 2024.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025.

Yongliang Wu, Yizhou Zhou, Zhou Ziheng, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, and Xu Yang. On the generalization of sft: A reinforcement learning perspective with reward rectification. *arXiv preprint arXiv:2508.05629*, 2025.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021, 2023.

# A EXPERIMENTAL DETAILS

## A.1 BASELINES

For each model, we evaluate several strategies for comparison:

**(1) FULL-TOKEN FINE-TUNING**: training on the entire pool of $50k$ high-quality samples using supervised fine-tuning.

**(2) UNIFORM RANDOM**: selecting a fixed proportion of $\rho$ tokens uniformly at random from the data pool for training.

**(3) RHO-1** (Lin et al., 2024): selecting the top-$\rho$ tokens within each sample based on excess loss.

**(4) TOKENCLEANING** (Pang et al., 2025): including Fixed-model Cleaning, which selects the global top-$\rho$ tokens from the entire data pool according to excess loss, and Self-evolving Cleaning, which partitions the data pool and iteratively updates the reference model while selecting global top-$\rho$ tokens from each partition.

**(5) ENTROPY-BASED TUNING** (Wang et al., 2025): a selective strategy that updates parameters only on tokens with entropy.

**(6) DYNAMIC FINE-TUNING** (Wu et al., 2025): a probability-aware method that assigns a dynamic scale to the loss of each token based on its confidence.

## A.2 EVALUATION METRICS AND BENCHMARKS

We evaluate TokenFlow on eight representative OpenLLM benchmarks covering diverse reasoning and knowledge-intensive capabilities. These benchmarks are not included in the training data construction process, and therefore evaluate the generalization ability of models after fine-tuning.

- **Commonsense Reasoning.** We use HELLASWAG (Zellers et al., 2019) to evaluate commonsense inference and contextual completion ability. We report *accuracy* as the evaluation metric.
- **Question Answering.** We adopt BOOLQ (Clark et al., 2019) and TYDIQA (Clark et al., 2020) to evaluate reading comprehension and question answering capabilities. For BOOLQ, we report *accuracy*. For TYDIQA, we report the standard *F1 score*.
- **Mathematical Reasoning.** We use GSM8K (Cobbe et al., 2021) to evaluate multi-step mathematical reasoning ability. We adopt the standard *exact match accuracy* metric.
- **Logical Reasoning.** We include LOGIQA (Liu et al., 2020) to measure formal logical reasoning ability. Performance is evaluated using *accuracy*.
- **Knowledge-based Reasoning.** We evaluate factual knowledge understanding using TRIVIAQA and OPENBOOKQA. TRIVIAQA measures open-domain question answering, while OPENBOOKQA evaluates scientific knowledge reasoning. We report *accuracy* for both benchmarks.
- **Science Reasoning.** We include ARC-C to evaluate grade-school science reasoning. The evaluation metric is *accuracy*.

## A.3 IMPLEMENTATION DETAILS

We evaluate TokenFlow on three representative open-source LLMs: LLaMA-3.1-8B, Qwen-2-7B, and OLMo-2-7B. Following the main experimental settings, we apply Low-Rank Adaptation (LoRA) Hu et al. (2022) for parameter-efficient fine-tuning. The maximum learning rate is set to $2 \times 10^{-5}$ with a linear decay schedule, and the batch size is set to 8. We fix the maximum input sequence length to 2080 tokens. All models are trained for 1 epoch on 4 A800 GPUs.

For TokenFlow-specific configurations, we set the token selection ratio $\rho = 0.6$ in all main experiments. During training, TokenFlow performs token selection and diversity-preserving optimization within the standard SFT pipeline. Specifically, given an instruction-response pair, TokenFlow first obtains intrinsic signals from the standard forward pass, including response-to-instruction attention patterns and token hidden representations. These signals are then used by **MINT** (**M**aximizing **IN**struction-aware **T**oken utility) to construct an instruction-aware token subset utility.

Table 4: Hyperparameter settings of TokenFlow.

| Hyperparameter | Default Value | Description |
|---|---|---|
| Learning rate | $2 \times 10^{-5}$ | Maximum learning rate with linear decay |
| Batch size | 8 | Training batch size |
| Sequence length | 2080 | Maximum input sequence length |
| Training epochs | 1 | Number of training epochs |
| Token selection ratio $\rho$ | 0.6 | Fraction of selected response tokens |
| DPP quality scaling $\gamma$ | 1.0 | Controls the contribution of instruction-aware token utility |
| TCE target weight $\alpha$ | 1.0 | Weight of the annotated target token |
| TCE alternative weight $\beta$ | 0.5 | Weight of plausible alternative tokens |
| Candidate size $K$ | 5 | Number of alternative tokens considered in TCE |

Different from independently ranking tokens by individual scores, MINT formulates token selection as a subset utility maximization problem. It jointly considers instruction-aware token relevance and pairwise token similarity, and selects a diverse and informative response-token subset under a fixed token budget. The selected tokens are subsequently optimized using **Diversity-preserving Tempered Cross-Entropy (TCE)**. TCE preserves the correctness supervision from the annotated token while incorporating plausible alternative tokens from the model's predictive distribution to maintain response diversity.

All required signals in TokenFlow are obtained from the current model during the standard forward pass without external reference models or additional inference procedures. Therefore, TokenFlow can be directly integrated into existing SFT pipelines without modifying the optimizer or training framework.

### A.4 Hyperparameter Analysis

We analyze the sensitivity of the key hyperparameters in TokenFlow. The token selection ratio $\rho$ determines the optimization budget allocated to response tokens. As shown in Section **??**, TokenFlow achieves the best performance with $\rho = 0.6$, indicating that an appropriate selection budget is important for balancing token informativeness and coverage. A smaller ratio may remove complementary supervision signals, while a larger ratio reduces the benefit of token-level selection by retaining less informative tokens.

The DPP quality scaling coefficient $\gamma$ controls the contribution of instruction-aware relevance in the subset utility. A larger $\gamma$ emphasizes individually relevant tokens, while a smaller value places relatively more emphasis on token diversity. Together with the diversity term, $\gamma$ balances selecting informative tokens and maintaining complementary supervision signals.

For the diversity-preserving optimization objective, the TCE coefficients $\alpha$ and $\beta$ determine the relative contribution of the annotated target and plausible alternatives. The target weight $\alpha$ maintains the correctness constraint from the original supervision, while the alternative weight $\beta$ controls the degree of distribution smoothing. The candidate size $K$ determines the number of alternative tokens incorporated into the calibrated supervision signal.

## B Detail of TokenFlow

The Training Workflow of TokenFlow is shown in Algorithm 1

### B.1 Overall Training Procedure of TokenFlow

Algorithm 1 summarizes the overall training procedure of TokenFlow. The training process follows the standard SFT pipeline while introducing token-level optimization in four steps.

**Step 1: Standard SFT Forward Pass.** Given a mini-batch of instruction-response pairs, TokenFlow first performs a standard forward pass with the current model parameters. The forward process produces token-level predictive distributions, attention representations, and hidden states: $\{\mathbf{P}^{(k)}, \mathbf{A}^{(k)}, \mathbf{H}^{(k)}\}$. These intermediate outputs are directly obtained from the normal SFT forward

**Algorithm 1** Overall Training Procedure of TokenFlow

**Require:** Dataset $\mathcal{D}$, model parameters $\theta$, selection budget $\rho$, learning rate $\eta$, training steps $K_{\max}$
**Ensure:** Fine-tuned model $\theta^*$

```
Initialize θ^(0) ← θ
for k = 0 to K_max − 1 do
    Sample a mini-batch B ∼ D
    Step 1: Standard SFT Forward Pass
    {P^(k), A^(k), H^(k)} ← Forward(B; θ^(k))
    Step 2: Token Utility Estimation and Selection
    u ← UtilityEstimator(P^(k), A^(k), H^(k))
    m_ρ ← TokenSelector(u, ρ)
    Step 3: Token-level Supervision Calibration
    L_token ← Calibration(P^(k), m_ρ)
    L_TokenFlow ← (Σ_i m_i L_i) / (Σ_i m_i)
    Step 4: Model Update
    θ^(k+1) ← θ^(k) − η ∇_θ L_TokenFlow
end for
return θ^(K_max)
```

computation and serve as the basis for subsequent token utility estimation, without requiring any additional inference or external models.

**Step 2: Token Utility Estimation and Selection.** Based on the intermediate representations obtained from Step 1, TokenFlow estimates the utility of each response token through the utility estimation module. The estimated utilities are then used by the token selection module to identify a compact subset of informative and complementary tokens under the given selection budget $\rho$. The selected tokens are represented by a binary mask $\mathbf{m}_\rho$, where only tokens with $m_i = 1$ are retained for optimization.

**Step 3: Token-level Supervision Calibration.** After determining which tokens should be optimized, TokenFlow further adjusts the supervision signal for the selected tokens. Instead of directly applying the original one-hot supervision from standard cross-entropy, the calibration module constructs a token-level objective that preserves the annotated target while incorporating additional supervision from plausible alternatives. The final token loss is computed according to the selected token mask, which allows the model to focus optimization on high-value tokens while maintaining a richer supervision signal.

**Step 4: Model Update.** Finally, TokenFlow updates the model parameters using the calibrated token-level objective through standard gradient-based optimization. The updated model is then used in the next training iteration to dynamically estimate token utilities based on the latest model state.

Overall, TokenFlow only modifies the token-level optimization process while keeping the standard SFT training pipeline unchanged, making it compatible with existing LLM fine-tuning frameworks.

### B.2 Token Subset Optimization

This section provides the detailed formulation and optimization procedure of the token selection module in TokenFlow. The objective of this module is to identify a compact subset of response tokens that provides the largest optimization benefit under a limited token budget.

Unlike methods that independently rank tokens according to individual scores, TokenFlow formulates token selection as a subset optimization problem. Specifically, the selection objective considers both the utility of individual tokens and the interactions among selected tokens, enabling the model to allocate optimization resources to informative and complementary token subsets.

#### B.2.1 Token Subset Optimization Objective

Given a set of response tokens $\mathcal{Z} = \{y_1, \ldots, y_{L_r}\}$, we aim to select a subset $\mathcal{S} \subseteq \mathcal{Z}$ with a fixed budget $\rho$:

$$\mathcal{S}^* = \arg \max_{\mathcal{S} \subseteq \mathcal{Z}, |\mathcal{S}| = \rho} f(\mathcal{S}), \tag{16}$$

where $f(\mathcal{S})$ denotes the overall utility of the selected token subset.

To model the interaction among selected tokens, we instantiate the subset utility function using a determinantal point process (DPP):

$$f(\mathcal{S}) = \det(\mathbf{L}_{\mathcal{S}}), \tag{17}$$

where $\mathbf{L}_{\mathcal{S}}$ is the submatrix of the subset utility kernel $\mathbf{L}$ corresponding to the selected tokens.

The kernel matrix is constructed as:

$$\mathbf{L} = \mathbf{Q}\mathbf{K}\mathbf{Q}, \tag{18}$$

where $\mathbf{Q} = \mathrm{diag}(q_1, \ldots, q_{L_r})$ contains token quality scores and $\mathbf{K}$ captures pairwise token interactions.

The quality score $q_i$ is obtained from the estimated token utility:

$$q_i = \mathrm{Normalize}(u_i), \tag{19}$$

where $u_i$ is produced by the token utility estimation module.

The interaction matrix is computed based on token representations:

$$K_{ij} = \mathrm{sim}(\mathbf{h}_i, \mathbf{h}_j), \tag{20}$$

where $\mathbf{h}_i$ denotes the hidden representation of token $y_i$.

The determinant objective in Eq. 17 measures the quality of a token subset as a whole. Tokens with high utility increase the subset objective through the quality term, while redundant tokens provide limited additional gains due to their strong interactions with previously selected tokens. Therefore, the optimization objective favors subsets containing informative and complementary response tokens.

### B.2.2 Greedy MAP Inference

Directly solving Eq. 16 requires searching over all possible token subsets and is computationally infeasible. Therefore, TokenFlow adopts a greedy MAP inference algorithm to efficiently approximate the optimal subset.

At each iteration, given the currently selected subset $\mathcal{S}$, the algorithm selects the token with the largest marginal gain:

$$j^* = \arg \max_{j \in \mathcal{Z} \setminus \mathcal{S}} \Delta(j|\mathcal{S}), \tag{21}$$

where $\Delta(j|\mathcal{S})$ denotes the increase of the determinant objective after adding token $j$.

To efficiently compute the marginal gain, we maintain an incremental Cholesky decomposition of the kernel matrix. Let $\mathbf{C}$ denote the accumulated Cholesky factors. The residual gain of each candidate token is computed as:

$$d_i^2 = L_{ii} - \sum_{t=1}^{|\mathcal{S}|} C_{ti}^2. \tag{22}$$

**Algorithm 2** Greedy MAP Inference for Token Subset Optimization

**Require:** Kernel matrix $\mathbf{L}$, candidate token set $\mathcal{Z}$, selection budget $\rho$
**Ensure:** Selected token subset $\mathcal{S}$
$\mathbf{C}_i \leftarrow [], \quad d_i^2 \leftarrow L_{ii}, \forall i \in \mathcal{Z}$
$j \leftarrow \arg\max_{i \in \mathcal{Z}} d_i^2, \mathcal{S} \leftarrow \{j\}$
**while** $|\mathcal{S}| < \rho$ **do**
  **for** $i \in \mathcal{Z} \setminus \mathcal{S}$ **do**
    $e_i \leftarrow \frac{L_{ji} - \langle \mathbf{C}_j, \mathbf{C}_i \rangle}{\sqrt{d_j^2}}$
    $\mathbf{C}_i \leftarrow [\mathbf{C}_i, e_i], \quad d_i^2 \leftarrow d_i^2 - e_i^2$
  **end for**
  $j \leftarrow \arg\max_{i \in \mathcal{Z} \setminus \mathcal{S}} d_i^2$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{j\}$
**end while**
**return** $\mathcal{S}$

The residual value $d_i^2$ represents the remaining contribution of token $i$ to the determinant objective. Thus, selecting the token with the largest residual value is equivalent to the greedy marginal gain maximization:

$$j^* = \arg\max_{j \in \mathcal{Z} \setminus \mathcal{S}} d_j^2. \tag{23}$$

After selecting token $j^*$, the Cholesky factors are updated by:

$$e_i = \frac{L_{j^* i} - \langle \mathbf{C}_{j^*}, \mathbf{C}_i \rangle}{\sqrt{d_{j^*}^2}}, \tag{24}$$

and the residual gains are updated as:

$$d_i^2 \leftarrow d_i^2 - e_i^2. \tag{25}$$

The procedure continues until the selected subset reaches the budget $\rho$. The complete greedy MAP inference procedure is summarized in Algorithm 2.

# C Additional Experiment

## C.1 Detailed Analysis of Selection and Training Objective.

We further disentangle the contributions of the token selection strategy and the training objective in Figure 8 and Table 2.

**Effect of TCE.** Figure 8(a) first examines whether TCE is tied to a particular token selection strategy. We replace the standard CE objective with TCE while keeping different selection methods unchanged. TCE consistently improves all evaluated strategies, yielding gains of $+1.42$ to $+2.17$ points across Full Token, Random, RHO-1, TokenCleaning, DFT, EFT, and our MINT selector. In particular, TCE improves MINT by $+2.01$ points, from $56.46$ to $58.47$. This consistent improvement suggests that TCE is largely orthogonal to the token selection mechanism: rather than changing *which* response tokens are selected, it improves *how* the selected tokens are optimized by preserving plausible alternative supervision while retaining the annotated target. The result also demonstrates that TCE can serve as a plug-and-play training objective for different token selection methods.

**Effect of MINT.** We next isolate the contribution of the selection strategy by fixing the training objective to standard CE in Figure 8(b). Under the same training objective and token budget, MINT achieves an average score of $56.46$, outperforming Full Token ($54.80$), Random ($54.50$), RHO-1 ($54.86$), TokenCleaning ($55.69$), DFT ($54.90$), and EFT ($54.41$). Notably, MINT surpasses the

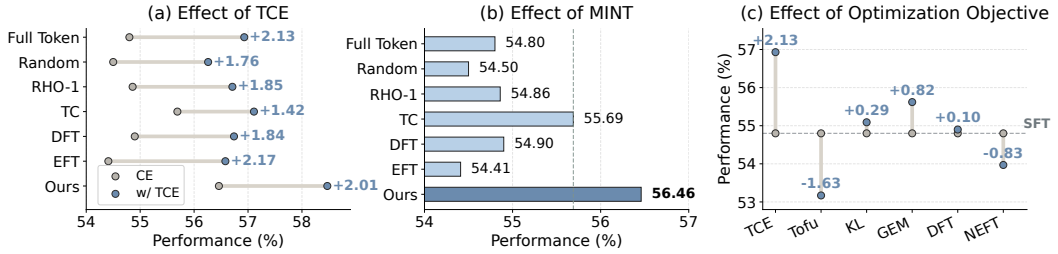



Figure 8: Analysis of Selection Strategy and Training Objective.

strongest competing selector, TokenCleaning, by 0.77 points, while also exceeding full-token training by 1.66 points. These results indicate that simply retaining more tokens or independently ranking tokens by individual scores is not sufficient. Instead, explicitly optimizing subset-level utility allows MINT to allocate the limited optimization budget to tokens that are both instruction-relevant and complementary.

The component-level ablation in Table 2 further supports this observation. Using only the relevance component reduces the average performance from 56.46 to 55.65, while using only the diversity-aware DPP component yields 55.98. Combining the two signals achieves the best performance, showing that instruction relevance and inter-token diversity provide complementary information for subset construction. The relevance term favors tokens that are strongly grounded in the instruction, whereas the diversity term discourages the selected subset from being dominated by redundant response tokens.

**Comparison of Training Objectives.** Finally, Figure 8(c) compares TCE with several alternative objectives under the same training setting. Relative to standard SFT, TCE provides the largest improvement of $+2.13$ points. In comparison, KL regularization, GEM, and DFT yield only $+0.29$, $+0.82$, and $+0.10$ points, respectively, while ToFu and MEFT decrease performance by 1.63 and 0.83 points. This contrast highlights an important distinction between TCE and generic regularization or target-modification objectives. Aggressively redistributing or suppressing the original supervision signal can introduce incorrect alternatives or weaken the learning signal. TCE instead retains the annotated token as the primary supervision target while selectively assigning probability mass to plausible alternatives from the model prediction distribution. Consequently, it improves response diversity without sacrificing the correctness signal required for effective SFT.

Overall, these results validate the two complementary design choices of TokenFlow: MINT determines *which tokens to optimize* by maximizing instruction-aware subset utility, while TCE determines *how the selected tokens are optimized* by calibrating their supervision targets. Their improvements are individually observable and remain complementary when combined, leading to the best overall performance of 58.47.

## C.2 The Generalization of TokenFlow.

### C.2.1 Generalization across Diverse Domains.

To evaluate the robustness of TokenFlow under diverse data distributions, we further conduct experiments on the multi-domain instruction-following dataset constructed by Designer, which covers 75 different domains. As shown in Table 5, TokenFlow consistently outperforms all token selection baselines, achieving the highest average performance of 42.58%. Compared with the strongest baseline TokenCleaning, TokenFlow improves the average score by 1.41 points, demonstrating that our token-level utility modeling remains effective beyond the original training distribution.

Moreover, TokenFlow achieves consistent gains across different reasoning-oriented benchmarks, including LogiQA2.0, GPQA, and MMLU-Pro. Notably, TokenFlow improves GPQA and MMLU-Pro by 0.54 and 1.10 points over TokenCleaning, respectively, suggesting that selecting instruction-aware and informative tokens is particularly beneficial for challenging knowledge and reasoning tasks. These results demonstrate that TokenFlow is not tailored to a specific instruction dataset, but provides a general token-efficient fine-tuning strategy applicable across heterogeneous domains.

Table 5: Generalization performance on the multi-domain instruction-following dataset from Designer, covering 75 diverse domains.

| Method | LogiQA2.0 | FOLIO | GPQA | MMLU-Pro | Avg. |
|---|---|---|---|---|---|
| Base | 33.95 | 39.22 | 35.35 | 33.74 | 35.56 |
| Full Token | 36.41 | 36.76 | 41.09 | 46.94 | 40.30 |
| Random | 34.41 | 43.63 | 40.91 | 39.85 | 39.70 |
| Rho-1 | 37.02 | 42.16 | 35.28 | 46.82 | 40.32 |
| TokenCleaning | 33.49 | 43.63 | 40.76 | 46.82 | 41.17 |
| **TokenFlow (Ours)** | **37.02** | **43.63** | **41.63** | **48.04** | **42.58** |

Table 6: Generalization across different model scales. We evaluate TokenFlow on LLaMA models with different sizes.

| Model | Method | HellaSwag | BoolQ | GSM8K | TyDiQA | LogiQA | Avg. |
|---|---|---|---|---|---|---|---|
| LLaMA-3.2-3B | Base | 55.10 | 73.00 | 28.51 | 32.69 | 22.12 | 42.28 |
| | Full Token | 55.23 | 74.22 | 29.04 | 38.65 | 24.58 | 44.34 |
| | Random | 54.80 | 73.82 | 29.49 | 35.67 | 23.20 | 43.40 |
| | Rho-1 | 54.38 | 74.98 | 31.01 | 40.75 | 23.04 | 44.83 |
| | TokenCleaning | 54.78 | 75.08 | 30.48 | 42.79 | 23.96 | 45.42 |
| | **TokenFlow (Ours)** | **55.74** | **77.49** | **34.65** | **50.03** | **25.96** | **48.77** |
| LLaMA-2-13B | Base | 60.01 | 80.91 | 24.12 | 31.20 | 25.89 | 44.43 |
| | Full Token | 57.76 | 74.92 | 31.08 | 44.37 | 23.04 | 46.23 |
| | Random | 56.26 | 74.92 | 29.34 | 40.86 | 23.04 | 44.88 |
| | Rho-1 | 58.75 | 77.06 | 31.84 | 42.03 | 24.58 | 46.85 |
| | TokenCleaning | 59.75 | 78.90 | 31.46 | 46.12 | 26.11 | 48.47 |
| | **TokenFlow (Ours)** | **60.81** | **83.03** | **32.45** | 44.89 | **28.11** | **49.86** |
| LLaMA-3.1-8B | Base | 59.92 | 82.18 | 50.31 | 22.80 | 26.51 | 48.34 |
| | Full Token | 60.57 | 83.11 | 54.13 | 48.21 | 28.06 | 54.82 |
| | Random | 60.14 | 82.80 | 51.98 | 49.83 | 27.91 | 54.53 |
| | Rho-1 | 60.46 | 83.57 | 54.21 | 47.53 | 28.06 | 54.77 |
| | TokenCleaning | 61.92 | 82.71 | 54.13 | 54.86 | 27.60 | 56.24 |
| | DFT | 60.81 | 82.92 | 54.74 | 50.97 | 26.51 | 55.19 |
| | EFT | 60.21 | 83.17 | 55.05 | 49.33 | 28.06 | 55.16 |
| | **TokenFlow (Ours)** | **66.48** | **85.79** | **55.90** | **58.72** | **29.30** | **59.24** |

### C.2.2 Generalization across Model Scales

**Experimental Setting.** To investigate whether the effectiveness of TokenFlow depends on a specific model scale, we conduct additional experiments on LLaMA models with different parameter sizes, including LLaMA-3.2-3B, LLaMA-3.1-8B, and LLaMA-2-13B. For all models, we adopt the same supervised fine-tuning setting and compare TokenFlow with representative token selection baselines, including Full Token, Random, Rho-1, TokenCleaning, and other strong baselines when available. The models are evaluated on five representative benchmarks covering diverse capabilities, including commonsense reasoning (HellaSwag), natural language understanding (BoolQ), mathematical reasoning (GSM8K), multilingual question answering (TyDiQA), and logical reasoning (LogiQA). The results are summarized in Table 6.

**Results Analysis.** As shown in Table 6, TokenFlow consistently achieves the best average performance across all evaluated model scales, demonstrating its robustness to different backbone capacities. Compared with the strongest token selection baseline, TokenFlow improves the average score by 3.35, 3.00, and 1.39 points on LLaMA-3.2-3B, LLaMA-3.1-8B, and LLaMA-2-13B, respectively. The consistent improvements across small, medium, and large models indicate that the effectiveness of TokenFlow is not tied to a particular model architecture or scale.

Furthermore, TokenFlow obtains larger gains on smaller models, especially on LLaMA-3.2-3B, where it improves the average performance from 45.42% of TokenCleaning to 48.77%. This sug-

Table 7: Generalization across Qwen model families. We evaluate TokenFlow on Qwen models with different scales.

| Model | Method | TyDiQA | HellaSwag | BoolQ | GSM8K | LogiQA | Avg. |
|---|---|---|---|---|---|---|---|
| Qwen3-4B | Base | 55.03 | 52.34 | 83.21 | 82.18 | 34.56 | 61.46 |
| | Full Token | 51.87 | 54.90 | 83.76 | 80.97 | 33.95 | 61.09 |
| | Random | 60.36 | 53.84 | 83.64 | 83.17 | 31.64 | 62.53 |
| | Rho-1 | 62.23 | 53.51 | 84.07 | 84.23 | 32.87 | 63.38 |
| | TokenCleaning | 62.46 | 53.28 | 83.64 | 81.50 | 32.26 | 62.63 |
| | **TokenFlow (Ours)** | **64.72** | **54.54** | **85.14** | **85.75** | **33.18** | **64.67** |
| Qwen-2-7B | Base | 30.72 | 59.38 | 76.72 | 50.20 | 50.03 | 53.41 |
| | Full Token | 31.16 | 58.83 | 78.10 | 51.07 | 56.43 | 55.12 |
| | Random | 31.32 | 57.02 | 76.88 | 51.85 | 52.81 | 53.98 |
| | Rho-1 | 32.56 | 57.02 | 75.57 | 49.57 | 54.01 | 53.75 |
| | TokenCleaning | 30.54 | 57.55 | 75.57 | 49.53 | 54.78 | 53.59 |
| | DFT | 29.77 | 57.23 | 76.65 | 56.38 | 53.72 | 54.75 |
| | EFT | 31.01 | 58.92 | 77.49 | 48.26 | 53.74 | 53.88 |
| | **TokenFlow (Ours)** | **33.33** | **64.19** | **77.95** | **65.14** | 55.50 | **59.22** |

gests that accurate token utility estimation is particularly beneficial for models with limited capacity, where inefficient optimization on redundant or less informative tokens may introduce stronger optimization interference. Meanwhile, TokenFlow also maintains stable improvements on larger models, such as LLaMA-3.1-8B and LLaMA-2-13B, demonstrating that selectively optimizing high-utility tokens remains effective even when the model has stronger representation capabilities.

### C.2.3 Generalization across Qwen Model Families

**Experimental Setting.** To further examine whether the effectiveness of TokenFlow generalizes beyond the LLaMA family, we conduct additional experiments on Qwen backbones with different model scales, including Qwen3-4B and Qwen-2-7B. For fair comparison, we report results on the five benchmarks shared by both models, including TyDiQA, HellaSwag, BoolQ, GSM8K, and LogiQA. For each backbone, we compare TokenFlow against representative token selection baselines, including Full Token, Random, Rho-1, and TokenCleaning, as well as DFT and EFT when available. The results are summarized in Table 7.

**Results Analysis.** As shown in Table 7, TokenFlow consistently achieves the best average performance across both Qwen backbones, demonstrating that its effectiveness generalizes beyond the originally evaluated model family. On Qwen3-4B, TokenFlow improves the average score from 63.38% of the strongest baseline to 64.67%, while on Qwen-2-7B, it improves the average performance from 55.12% to 59.22%. These results show that the advantages of TokenFlow are not tied to a specific architecture or scale.

More importantly, TokenFlow delivers consistent gains on challenging reasoning-oriented benchmarks such as TyDiQA and GSM8K. For example, on Qwen3-4B, TokenFlow achieves the best results on TyDiQA, BoolQ, and GSM8K, while on Qwen-2-7B, it substantially improves HellaSwag, TyDiQA, and GSM8K. This suggests that the proposed token utility estimation and diversity-aware optimization remain effective across different pretraining families and parameter scales, further validating the robustness of TokenFlow.