UniToR: Towards Universal Token-level Routing for Efficient Reasoning in Large Language Models

Nguyen-Khang Le[1*], Dinh-Truong Do[1*], Nguyen Le Minh[1]

[1]Japan Advanced Institute of Science and Technology

Correspondence: {lnkhang, truongdo, nguyenml}@jaist.ac.jp

Abstract

Large Reasoning Models (LRMs) deliver strong performance on complex reasoning tasks, but their long chain-of-thought traces incur substantial inference latency and cost. To mitigate this overhead while preserving reasoning quality, token-level routing between a small language model (SLM) and a large language model (LLM) offers one of the best efficiency–performance trade-offs. However, existing token routers face two major practical limitations: they must be retrained for every new SLM–LLM pair, and they require both models to share a tokenizer. We present UniToR, a universal token-level router that removes both constraints. UniToR predicts per-token divergence using model-agnostic signals, enabling a single router to generalize across diverse SLM–LLM pairs without retraining. To support heterogeneous model pairs, we further introduce a token-alignment mechanism that eliminates the requirement for tokenizer compatibility, enabling seamless token-level switching across models with different tokenization schemes. Experimental results show that a single UniToR router generalizes across different model families (DeepSeek-R1, Qwen3, and Phi-4), including pairs with incompatible tokenizers. UniToR matches the accuracy of R1-32B at up to 6× lower cost and successfully operates on unseen model pairs where existing token-level routing methods fail.

1 Introduction

Large Language Models (LLMs) exhibit remarkable capabilities across diverse tasks. On challenging problems, recent work increasingly relies on test-time scaling of the largest and most powerful LLMs, often by generating extensive Chain-of-Thought (CoT) reasoning (Wei et al., 2022) before producing the final answer. While effective, this practice incurs substantial computational cost and latency (Jaech et al., 2024; Guo et al., 2025): the expense is driven not only by model size but also by the volume of intermediate "thinking" tokens. This creates a central deployment dilemma for interactive applications: how to retain strong reasoning while keeping inference compute manageable.

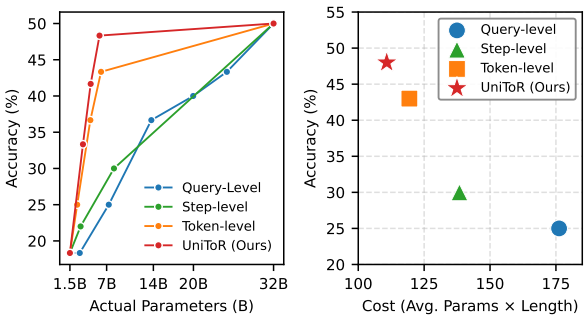


Figure 1: UniToR Pareto-dominates prior routing methods on both accuracy vs. effective parameters (left) and accuracy vs. cost (right).

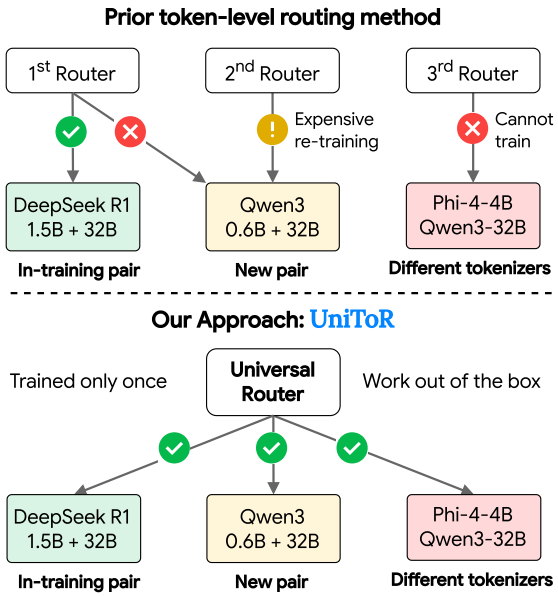


Figure 2: Prior token-level routers require retraining per SLM–LLM pair and fail on incompatible tokenizers. A single UniToR router, trained once, works for all pairs.

*These authors contributed equally to this work

Findings of the Association for Computational Linguistics: IJCNLP-AACL 2026, pages 4197–4215
November 6-10, 2026 ©2026 Association for Computational Linguistics

To resolve this tension, *model routing* has emerged as a critical efficiency strategy (Chen et al., 2024; Chuang et al., 2025). By orchestrating collaboration between cost-effective Small Language Models (SLMs) and capable but expensive LLMs, routing systems aim to invoke the heavy model only when necessary. The evolution of these systems is defined by their *granularity of control*.

Current routing approaches can be categorized into **query-level** routing (Ong et al., 2025; Feng et al., 2025) (Figure 3a), directing an entire user prompt to a strong or weak model. While effective for separating distinct complexity classes, this enforces a binary decision, causing mixed-difficulty queries to be either over-served by the LLM or under-served by the SLM. To address this, **step-level** methods (Pan et al., 2025; Fu et al., 2025b) (Figure 3b) decompose generation into intermediate reasoning steps, employing a "draft-then-verify" paradigm. However, this introduces a dependency on the verifier; if the SLM's draft is rejected, the system incurs the penalty of rollback and regeneration. The frontier of efficiency is currently defined by **token-level** routing (Fu et al., 2025a) (Figure 3c). The key insight is that SLMs can generate most low-risk syntax and simple-logic tokens, while LLM intervention is only needed at pivotal "branching points" in the reasoning path. By switching models at token level, these systems achieve a better efficiency–performance trade-off than coarser routing schemes.

However, state-of-the-art token routers (Fu et al., 2025a) suffer from a critical scalability limitation: **lack of transferability**. In particular, the router is typically tightly coupled to a specific SLM–LLM pair and assumes a shared tokenizer, making it difficult to reuse across model families (e.g., a router trained for Qwen3 can not transfer to DeepSeek-R1 or Phi4). This limitation stems from two issues. First, adapting to a new SLM–LLM pair typically requires an expensive data-generation pipeline and retraining, because the router is trained on model-specific representations (e.g., hidden states or token embeddings). Second, token-level switching requires the SLM and LLM to share a compatible tokenizer/vocabulary.

We introduce the **Universal Token-Level Router** (UniToR), which removes the need for pair-specific router training and eliminates the requirement that the SLM and LLM share a tokenizer. Our key idea is twofold. First, we predict divergence using *universal*, model-agnostic signals, including statistics of the SLM next-token distribution, token-identity cues, and other lightweight features (Section 3.2), enabling a single router to generalize across SLM–LLM pairs without retraining. Second, at inference time, we introduce a token-alignment mechanism that reconciles mismatched tokenizers by mapping recent generated tokens back to text and re-encoding under the other model's vocabulary, allowing token-level routing across heterogeneous vocabularies.

With UniToR, we achieve a "plug-and-play" capability: the router can be deployed on unseen model pairs without per-pair data generation or fine-tuning, and without constraints on tokenizer/vocabulary compatibility. We show that this universal approach retains the efficiency benefits of token-level routing and improves downstream performance (Figure 1, 2), while eliminating the overhead of per-pair training and the tokenizer-compatibility constraint. This paves the way for scalable and accessible test-time compute optimization. In summary, our contributions are:

- We propose a token-level router that predicts divergence using model-agnostic signals, enabling zero-shot transfer across model pairs.
- We introduce a token-alignment mechanism for inference that supports token-level switching between models with different tokenizers/vocabularies.
- We demonstrate empirically that our universal routing improves the accuracy–cost trade-off and transfers robustly across different SLM–LLM pairs.

2 Preliminaries on Token-Level Routing

We study token-level routing between a Small Language Model (SLM) and a Large Language Model (LLM) to balance generation quality and inference cost. Let M_s and M_l denote the SLM and LLM, parameterized by θ_s and θ_l, respectively. Both models are autoregressive models that generate tokens sequentially conditioned on the preceding context.

2.1 Divergent and Non-divergent Tokens

Prior work (Fu et al., 2025a) observes that SLMs and LLMs often agree on the next-token prediction when conditioned on the same context. In practice, however, large performance gaps arise less from isolated errors and more from error accumulation: once a critical mismatch appears in a

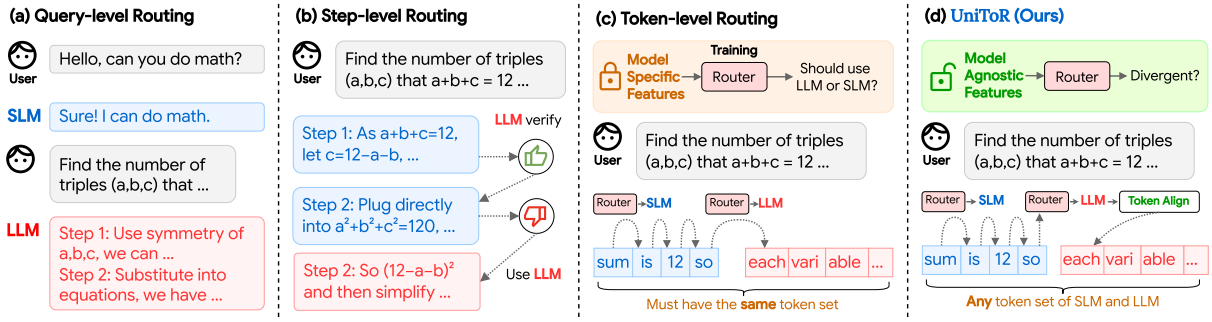


Figure 3: Overview of UniToR compared to other routing methods. (a) Query-level routing operates at the query level (Ong et al., 2025; Feng et al., 2025). (b) Step-level routing operates at the step level via a draft-and-verify paradigm (Pan et al., 2025; Fu et al., 2025b). (c) Token-level routing offers high efficiency but requires a heavy data pipeline and retraining for each new SLM, and assumes the SLM and LLM share the same token set. (d) UniToR achieves token-level efficiency without constraining the model pair or vocabulary.

partial response, the two models can follow increasingly different reasoning trajectories. Following Fu et al. (2025a), we categorize token-level differences between SLM and LLM outputs into two types. *Non-divergent* differences are surface-form variations (e.g., *don't* vs. *do not*, *because* vs. *since*) that preserve semantics and do not affect downstream reasoning. *Divergent* differences change meaning or logic, causing subsequent reasoning to deviate.

Motivated by this distinction, token-level routing decides online whether an SLM-proposed token is divergent or non-divergent. If a token is classified as divergent, it is discarded and regenerated by the LLM; otherwise, it is accepted. A router makes this per-token divergence decision during generation.

2.2 Token-Level Routing

Token-level routing enables dynamic model selection at each decoding step. Given a query q with token representation $\mathbf{x} = (x_1, \ldots, x_n)$, decoding proceeds autoregressively. At step t, the routing function selects the model based on the output of the SLM.

$$m_t = \mathcal{R}_{\text{token}}(\mathbf{x}, y_{<t}; \theta_s) \in \{M_s, M_l\},$$

where $y_{<t} = (y_1, \ldots, y_{t-1})$ denotes the previously generated tokens. The selected model m_t then generates the next token via greedy decoding:

$$y_t = \arg\max_y P_{m_t}(y \mid \mathbf{x}, y_{<t}).$$

This fine-grained routing leverages the SLM for low-risk tokens while invoking the LLM only when necessary. Other routing strategies (e.g., query-level, step-level) are provided in Appendix E.

3 UniToR: Universal Token Routing

We present UniToR, a universal token-level routing framework that achieves model-agnostic training and inference, enabling seamless deployment with any SLM-LLM pair while maintaining high efficiency and performance. Prior token-level routing methods (Fu et al., 2025a) face two key limitations that prevent generalization across model pairs: (1) the router design and training process rely on model-specific features (e.g., hidden states, token embeddings), tightly coupling the router to a particular SLM-LLM combination; and (2) the inference procedure assumes both models share the same vocabulary, which rarely holds in practice. UniToR addresses both limitations.

For training, we design the router to operate on model-agnostic features extracted from the SLM's output distribution, rather than internal representations. We develop an efficient data generation pipeline that produces training samples applicable to any model pair. For inference, we introduce a token alignment mechanism that handles vocabulary mismatches, enabling seamless switching between models with different tokenizers.

3.1 Training of Universal Token Routing

The router model is the core component of token-level routing, responsible for determining whether the SLM or LLM should generate each token. Formally, the router learns a binary classification function: $\mathcal{R} : \mathbf{f}_t \rightarrow \{0, 1\}$, where $\mathbf{f}_t$ represents the features extracted at token position t, and the output indicates whether the token is *divergent* (1) or *non-divergent* (0). A token is considered divergent if using the SLM at that position would lead to semantically different output compared to the LLM.

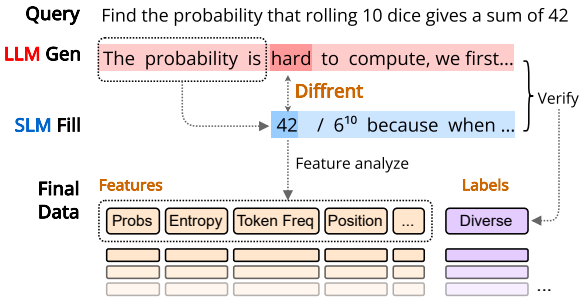


Figure 4: Illustration of the per-token data generation process. For each token position in the LLM-generated reference, we collect SLM features and determine whether the token is divergent.

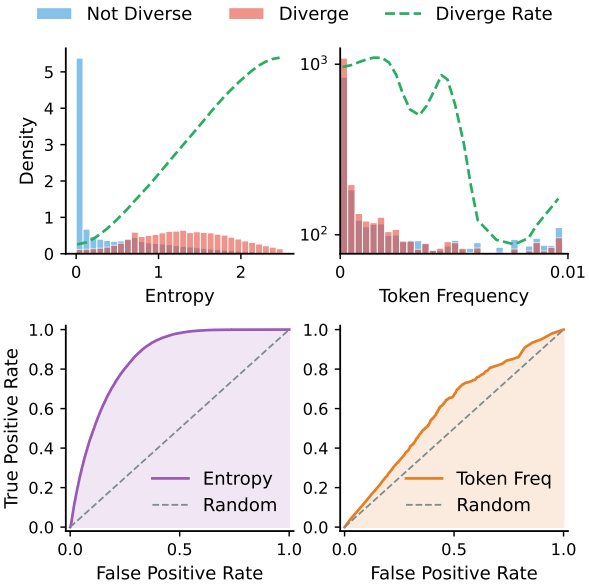


Figure 5: Analysis of model-agnostic features: entropy (left) and token frequency (right). Data distribution and divergence rate are shown on the top, and the ROC curve is shown on the bottom.

When a token is predicted as divergent, the LLM is invoked; otherwise, the SLM generates the token.

Prior work trains the router using hidden states or token embeddings from a specific SLM, which tightly couples the router to a particular model pair. To enable generalization, we propose a data generation pipeline that produces model-agnostic training data. Figure 4 illustrates our data generation process. Given a query q, we first use the LLM to generate a reference response $\mathbf{y}^{\text{ref}} = (y_1^{\text{ref}}, \ldots, y_T^{\text{ref}})$. For each token position t, we prompt the SLM to generate a token $\hat{y}_t$ conditioned on the query and the preceding reference tokens:

$$\hat{y}_t = \arg\max_y P_{M_s}(y \mid q, y_{<t}^{\text{ref}}).$$

If $\hat{y}_t \neq y_t^{\text{ref}}$, we continue generation with the SLM to complete the response and use a verifier model to assess whether this substitution leads to semantic divergence from the LLM output. This provides the supervision signal: the token is labeled as *divergent* if the SLM continuation is semantically inconsistent with the reference, and *non-divergent* otherwise. For each position, we extract model-agnostic features $\mathbf{f}_t$ from the SLM's output distribution and pair them with the divergence label to form a training instance $(\mathbf{f}_t, d_t)$, where $d_t \in \{0, 1\}$. Using this pipeline, we generate 15 million training samples from queries drawn from (Labs, 2025).

3.2 Model-Agnostic Features

While prior work (Fu et al., 2025a) relies on model-specific signals (e.g., hidden states and token embeddings), we find that *model-agnostic* features are highly predictive of token-level divergence. We group these features into (i) distribution-based uncertainty features from the SLM next-token probabilities, (ii) token-identity features, and (iii) other lightweight features (Appendix D).

Distribution features. Since models differ in vocabulary and distribution shape, we make these features model-agnostic by extracting the top-k SLM next-token probabilities, renormalizing over the truncated support, and computing summary statistics; throughout, we set $k = 100$. Entropy is particularly informative:

$$H_t = -\sum_{v \in \mathcal{V}_S} P_{M_S}(v \mid q, y_{<t}) \log P_{M_S}(v \mid q, y_{<t}),$$

where higher H_t indicates greater uncertainty and correlates with higher divergence rates (Figure 5). Other statistics derived from the normalized top-k distribution are also effective (Appendix D.1).

Token-identity features. We include features that depend only on the selected token, such as *token frequency*. To remain model-agnostic, we estimate frequencies from external corpora (e.g., Wikipedia or FineWeb (Penedo et al., 2024)). Higher-frequency tokens are less likely to be divergent (Figure 5). Token type (digits, punctuation) and normalized vocabulary position provide additional signal (Appendix D.2).

Other features. We also include lightweight structural features (e.g., relative position in the sequence); details are provided in Appendix D.

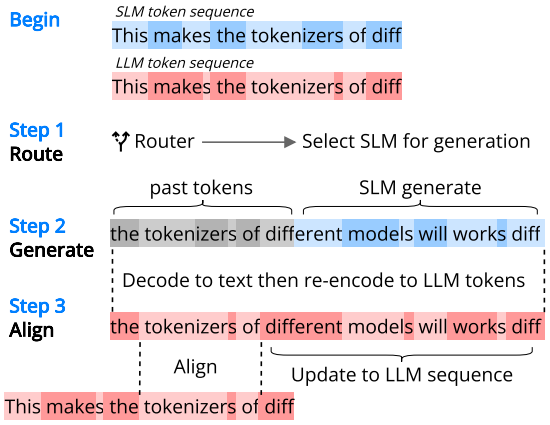


Figure 6: SLM/LLM token alignment process. Red text indicates tokens generated by the LLM; Blue text indicates tokens generated by the SLM; different shades of text separate tokens.

3.3 Inference with Token Alignment

A key challenge for universal token-level routing is vocabulary mismatch: the SLM and LLM typically use different tokenizers with distinct vocabularies $\mathcal{V}_S$ and $\mathcal{V}_L$. As a result, tokens produced by one model cannot be directly appended to the other model's token sequence, because the same text may be segmented differently. To address this challenge, we introduce a token-alignment mechanism that maintains consistent text while allowing seamless model switching. We maintain two token sequences $\mathbf{s} = (s_i)_{i=1}^{t} \in \mathcal{V}_S^t$ and $\mathbf{l} = (l_j)_{j=1}^{u} \in \mathcal{V}_L^u$ representing the same text.

where t and u may differ due to tokenizer granularity. At each generation step, the router $\mathcal{R}$ selects a model based on the current features: $m = \mathcal{R}(\mathbf{f}) \in \{M_s, M_l\}$. Alignment is triggered only when the selected model changes (i.e., when switching between models).

We describe the procedure for switching from SLM to LLM; the reverse direction is handled symmetrically. The key idea is to decode a window of recent SLM tokens to text, then re-encode this text with the LLM tokenizer. Using a window of size W stabilizes token boundaries during re-encoding. We then align the re-encoded tokens with the existing LLM sequence by finding the longest overlap and append only the unmatched suffix to update the LLM sequence. Algorithm 1 provides pseudocode.

4 Experiments

In this section, we begin by describing the experimental setup in Section 4.1. We then compare the performance and efficiency of our proposed UniToR with current state-of-the-art methods in Section 4.2. Finally, we conduct a series of ablation studies in Section 4.3 to assess the contribution of each component of our method.

4.1 Experimental Setup

Models. Each routing method requires a small language model (SLM) and a large language model (LLM). We evaluate UniToR using three LLMs—DeepSeek-R1-Qwen-32B, Qwen3-32B, and Phi4-Reasoning-14B—and three SLMs—DeepSeek-R1-Qwen-1.5B, Qwen3-0.6B, and Phi-4-Mini-Reasoning-3.8B. We evaluate all heterogeneous SLM–LLM combinations across families, requiring the router to generalize without shared architectures or tokenizers between the paired models.

Baselines. We compare against state-of-the-art baselines at each routing granularity. (1) **Query-level routing:** HybridLLM (Ding et al., 2024), GraphRouter (Feng et al., 2025), and methods from the RouteLLM framework (Ong et al., 2025), including a BERT-based classifier (*QR-BERT*), matrix factorization (*QR-MF*), an LLM-based classifier (*QR-LLM*), and a weighted Elo-based routing strategy (*QR-SW*). (2) **Step-level routing:** *SpecReason* (Pan et al., 2025), which routes between the SLM and LLM at the level of reasoning steps. (3) **Token-level routing:** *R2R* (Fu et al., 2025a), a pair-specific token router that serves as a strong token-level inference method. Since R2R is not transferable across model pairs, we only report results using the official released weights for the DeepSeek-R1-Qwen-1.5B → 32B pair. (4) **Speculative decoding:** EAGLE-2 (Li et al., 2024) and EAGLE-3 (Li et al., 2025).

Benchmarks. We evaluate mathematical reasoning on AIME (2024–2025) (Veeraboina, 2023), scientific and general knowledge reasoning on GPQA (Rein et al., 2024), and code generation on LiveCodeBench (2024–2025) (Jain et al., 2025). Unless otherwise noted, we set the maximum generation length to 32K tokens.

Metrics. Following prior work (Fu et al., 2025a), we report **Accuracy (Acc.)**. For computation, we use **Average Activated Parameters** P_{avg} (in billions), which accounts for the average parameters used per generated token, including the SLM, router, and the LLM weighted by its activation frequency. We also report **Total Cost** C, defined as $C = P_{\text{avg}} \times \#\text{output tokens}$, which captures total

inference computation per query in a hardware-agnostic manner. To assess practical efficiency, we additionally report **Latency**, measured as wall-clock end-to-end decoding time per query under a fixed hardware and implementation setting.

Implementation details. UNITOR uses a lightweight FFN router with 35M parameters, trained on 15M token-level routing labels. We keep the router fixed across evaluations. To meet a user-specified budget, we choose the routing threshold on a held-out calibration split to match a target P_{avg} of 6B. Additional implementation details for baselines, our method, and the label-generation procedure are provided in Appendix A.

4.2 Performance & Efficiency

Universal routing across model pairs. Table 1 reports AIME results on both within-family and cross-family SLM–LLM pairs. Across all settings, UNITOR achieves the best accuracy–cost trade-off, consistently outperforming query-level and step-level routing while using fewer activated parameters and lower inference cost. These results highlight the effectiveness of token-level routing with a universal routing strategy.

UNITOR also exceeds the performance of the specialized token-level router R2R (Fu et al., 2025a) on the DeepSeek pair, despite R2R being trained using model-specific features such as hidden states or input embeddings. In contrast, UNITOR relies solely on universal features, including normalized logits and token-frequency statistics. More importantly, UNITOR generalizes to unseen SLM–LLM pairs without any per-pair data generation or retraining. This robustness is enabled by (i) routing based on universal uncertainty signals rather than model-dependent representations, and (ii) our tokenizer alignment mechanism, which supports heterogeneous vocabularies without additional supervision.

Generality across benchmarks. To evaluate generalization across domains, Figure 7 reports results on three generation benchmarks—AIME, LiveCodeBench, and GPQA—using the DeepSeek-R1 model family. In addition to routing methods, we include the standalone SLM (R1-1.5B), the full LLM (R1-32B), and intermediate distilled models (R1-7B and R1-14B) for reference. Across all benchmarks, UNITOR consistently outperforms query-level and step-level routing methods at comparable or lower average activated parameter budgets, and remains competitive with the pair-specific token router R2R (Fu et al., 2025a) while requiring no per-pair router training. Remarkably, UNITOR also surpasses the R1-14B model while maintaining an average activated parameter size of only 5.9–6.4B, demonstrating robust generalization across domains.

Scaling behavior. Figure 8 plots accuracy as a function of the average activated parameters on AIME and GPQA across multiple DeepSeek-R1-based model pair configurations. Distilled models (R1-7B, R1-14B) achieve strong performance through extensive training, reaching approximately 88–90% of the 32B model's accuracy while using only about half of the parameters at 14B. In contrast, by routing only divergent tokens to the LLM, UNITOR achieves 92% of the LLM's accuracy while activating only 17% of its parameters, establishing a new and more favorable Pareto frontier. Moreover, the adjustable routing threshold in UNITOR enables post-training control over the accuracy–compute trade-off, allowing users to target arbitrary average activated parameter budgets while consistently outperforming baseline routing methods at comparable budgets.

Latency and throughput. Table 2 reports wall-clock latency and throughput on AIME for DeepSeek-R1 (1.5B $\rightarrow$ 32B) and Qwen3 (0.6B $\rightarrow$ 32B). All methods are implemented in SGLang (Zheng et al., 2024) and evaluated with tensor parallelism on three NVIDIA A6000 (48GB) GPUs. UNITOR uses the same routing thresholds as Table 1, while query-level methods use the 7B routed option for comparable activated parameters. Across both model pairs, UNITOR achieves major speedups over LLM-only decoding, delivering $4.47\times$ and $4.86\times$ higher throughput than R1-32B and Qwen3-32B, respectively. Compared to query-level and step-level routing, UNITOR consistently reduces latency and improves throughput. It also outperforms optimized speculative decoding methods such as EAGLE-2/3 (Li et al., 2024, 2025). Although the pair-specific router R2R achieves competitive throughput, UNITOR attains lower latency by generating fewer tokens. Overall, these results show that UNITOR provides strong end-to-end efficiency while remaining practical for deployment.

4.3 Analysis

Impact of router features. Our router avoids model-specific representations (e.g., hidden states

Small Model \ Target Model		DeepSeek-R1-Distill-Qwen-32B			Qwen3-32B			Phi4-Reasoning-14B		
		Acc.	Param.	Cost	Acc.	Param.	Cost	Acc.	Param.	Cost
DeepSeek-R1-Distill Qwen-1.5B	QR-BERT	20%	6.7	174.0	38%	7.2	156.2	18%	6.8	155.4
	QR-MF	25%	7.3	176.2	33%	7.3	169.2	12%	7.3	180.3
	QR-SW	23%	7.0	169.8	37%	6.9	151.7	15%	7.2	175.2
	QR-LLM	23%	6.8	163.5	32%	7.0	165.3	17%	7.2	166.4
	SpecReason	30%	8.1	138.4	37%	6.8	102.2	45%	7.2	114.2
	R2R	43%	6.2	119.4	–	–	–	–	–	–
	UniToR (Ours)	**48%**	**5.9**	**110.8**	**43%**	**5.5**	**97.8**	**45%**	**4.2**	**88.7**
Qwen3 0.6B	QR-BERT	13%	7.2	170.5	33%	7.1	138.9	12%	7.4	152.4
	QR-MF	25%	6.9	136.3	32%	7.1	136.9	12%	7.4	150.8
	QR-SW	17%	6.8	157.7	30%	7.0	142.7	10%	6.9	156.2
	QR-LLM	20%	6.7	142.8	25%	7.1	153.2	13%	7.0	147.1
	SpecReason	22%	10.2	129.3	18%	9.8	123.9	15%	7.2	78.4
	UniToR (Ours)	**30%**	**6.1**	**118.5**	**53%**	**6.7**	**111.5**	**35%**	**3.5**	**63.9**
Phi-4-Mini Reasoning-3.8B	QR-BERT	37%	7.0	208.8	52%	7.1	186.2	28%	6.9	191.3
	QR-MF	38%	7.1	198.8	48%	6.9	182.0	25%	6.8	189.6
	QR-SW	37%	6.9	200.2	50%	6.7	177.5	28%	7.0	208.6
	QR-LLM	38%	6.9	207.3	50%	7.3	207.9	28%	7.1	199.5
	SpecReason	38%	9.6	120.9	37%	9.7	139.1	47%	7.8	99.6
	UniToR (Ours)	**38%**	**6.7**	**96.6**	**55%**	**6.7**	**121.4**	**53%**	**5.4**	**58.2**

Table 1: Performance of UniToR and baselines on within-family and cross-family SLM–LLM pairs on AIME. *Param.* denotes average activated parameters per token (billions), and *Cost* is the average output length (thousands of tokens) multiplied by average activated parameters (billions). Entries marked "–" indicate methods that are not applicable to cross-family pairs.

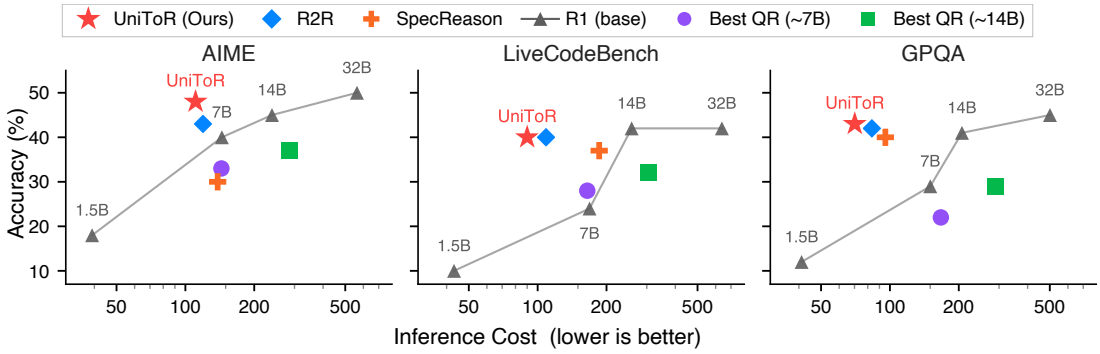


Figure 7: Performance on DeepSeek-R1-Distill-Qwen (1.5B → 32B) across AIME, LiveCodeBench, GPQA. Best QR denotes the best-performing result among the four RouteLLM frameworks (Ong et al., 2025).

or embeddings) and instead uses universal signals from the SLM next-token distribution and token properties. We compare the full feature set (*distribution*, *token-identity*, and lightweight auxiliary features) with ablated variants in Table 3. Consistent with Section 3.2, distribution-based uncertainty is the strongest signal: using only distribution features retains about 90% of the full model's accuracy. Adding token-identity features further improves performance, while the remaining lightweight features provide smaller but consistent gains by complementing uncertainty signals and improving divergence detection.

Impact of training data size. We study how routing supervision scale affects a single universal router by training the same architecture with 5M, 10M, and 15M token-level divergence labels. Table 4 shows a clear scaling trend: more training data improves accuracy while reducing LLM usage and overall cost. With only 5M labels, the router over-routes to the LLM, leading to higher activation rates and cost. Scaling to 10M labels already reduces both cost and latency. Using the full 15M-

Router Features	Acc.	#Token (K)	Param. (B)	Cost	Lat. (s)
Dist. + Tok-ID + Other	**48%**	**18.7**	**5.9**	**110.8**	**220**
Dist. + Tok-ID	47%	20.5	5.9	120.9	246
Dist. only	42%	19.0	6.1	119.7	243

Table 3: Feature ablation on the AIME benchmark. *Dist.* denotes distribution-based uncertainty features derived from logits; *Tok-ID* denotes token-identity features; *Other* denotes other input features (Appendix A).

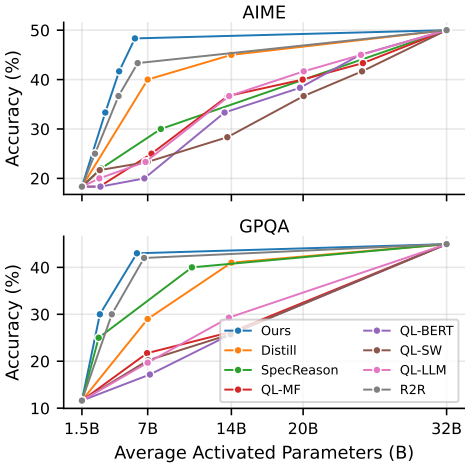


Figure 8: Accuracy as a function of average activated parameters on AIME and GPQA.

Method	DeepSeek-R1-1.5B + R1-32B			Qwen3-0.6B + Qwen3-32B		
	#Token (K)	Lat. (s)	Spd (tok/s)	#Token (K)	Lat. (s)	Spd (tok/s)
SLM Only	$26.0_{\pm 11.8}$	$193.7_{\pm 89.0}$	134.4	$22.7_{\pm 10.8}$	$114.9_{\pm 64.7}$	197.3
LLM Only	$17.6_{\pm 12.8}$	$924.1_{\pm 685.0}$	19.0	$15.6_{\pm 10.0}$	$827.9_{\pm 542.4}$	18.9
QR-SW	$24.3_{\pm 12.6}$	$379.0_{\pm 506.2}$	64.0	$20.4_{\pm 10.5}$	$278.0_{\pm 353.3}$	73.3
QR-LLM	$24.0_{\pm 12.6}$	$367.5_{\pm 476.5}$	65.2	$21.6_{\pm 11.3}$	$300.8_{\pm 476.4}$	71.7
QR-BERT	$25.9_{\pm 11.7}$	$394.1_{\pm 513.2}$	65.7	$19.6_{\pm 10.7}$	$269.4_{\pm 333.7}$	72.7
QR-MF	$24.1_{\pm 12.6}$	$386.4_{\pm 477.4}$	62.3	$19.3_{\pm 11.4}$	$266.4_{\pm 390.1}$	72.3
EAGLE-2	$15.7_{\pm 12.3}$	$364.0_{\pm 303.8}$	43.1	–	–	–
EAGLE-3	–	–	–	$13.9_{\pm 8.4}$	$430.9_{\pm 299.4}$	31.0
SpecReason	$17.1_{\pm 8.3}$	$369.3_{\pm 322.1}$	46.3	$12.6_{\pm 7.7}$	$262.5_{\pm 219.4}$	48.0
R2R	$19.4_{\pm 13.1}$	$232.3_{\pm 151.4}$	83.5	–	–	–
UniToR (Ours)	$18.7_{\pm 13.4}$	**$220.0_{\pm 157.6}$**	**85.0**	$18.7_{\pm 10.9}$	**$203.5_{\pm 118.6}$**	**91.9**

Table 2: Latency and decoding speed on AIME. We report the average generated tokens (#Token), wall-clock latency, and throughput. Values are mean ± standard deviation. "–" indicates unavailable baselines due to missing drafter weights for the corresponding model pair.

Training Size	Acc.	#Token (K)	Param. (B)	Cost	Lat. (s)
15M	**48%**	**18.7**	**5.9**	**110.8**	**220**
10M	43%	18.5	6.1	112.9	231
5M	42%	17.9	7.9	141.4	263

Table 4: Training-label scaling on AIME. More labels improve calibration, reducing LLM usage and cost while increasing accuracy.

label set achieves the best trade-off, reaching 48% accuracy with the lowest cost and latency. These results suggest that larger supervision improves calibration, enabling the router to better detect divergent tokens and avoid unnecessary LLM calls.

Additional analyses. Additional analyses and extended results are provided in Appendix B.

5 Related Work

Test-time scaling improves LLM performance but often requires long Chain-of-Thought (CoT) traces, increasing latency and inference cost (Jaech et al., 2024; Guo et al., 2025; Wang and Zhou, 2024). To mitigate this overhead, speculative decoding (Le et al., 2025; Miao et al., 2024) and model routing (Srivatsa et al., 2024; Ong et al., 2025) have received significant attention.

Speculative Decoding. Speculative Decoding (SD) uses a smaller "draft" model (SLM) to propose tokens that are then verified by the target LLM (Sun et al., 2023; Leviathan et al., 2023; Le et al., 2025). EAGLE-2/3 (Li et al., 2024, 2025) improve acceptance via stronger feature integration and dynamic draft trees, and HASS (Zhang et al., 2025) aligns draft and target representations. However, classical SD enforces strict distribution matching by rejecting any token that differs from the LLM output, which can discard valid paraphrases and limit speedups in open-ended reasoning.

Model Routing. Model routing assigns computation based on input complexity to avoid strict token-level verification. Early methods use *query-level* routing (Feng et al., 2025; Lu et al., 2024; Wang et al., 2025), sending each prompt to a strong or weak model (e.g., RouteLLM (Ong et al., 2025)). However, this coarse choice can be suboptimal for mixed-difficulty prompts (Fu et al., 2025a). *Step-level* methods, such as SpecReason (Pan et al., 2025), route over intermediate steps but often rely on delimiters/heuristics and can be costly when steps are rejected.

Token-Level Routing. The finest control granularity is *token-level* routing, where the system decides model usage for every token generated. Approaches like CITER (Zheng et al., 2025) and RSD (Liao et al., 2025) utilize reinforcement learning or reward models to govern these transitions. Most notably, R2R (Fu et al., 2025a) established a new state-of-the-art by training a lightweight router to detect "semantic divergence" points where the SLM's reasoning path drifts from the LLM's intended logic. However, these routers are typically pair-specific and trained on model-internal states, making them hard to transfer to new SLM–LLM pairs. UniToR targets this limitation.

6 Conclusion

We presented UniToR, a universal token-level routing framework that eliminates pair-specific router training and supports heterogeneous SLM–

LLM vocabularies via token alignment. By routing based on model-agnostic uncertainty features and token-identity cues, UNITOR generalizes across model pairs and delivers strong accuracy–efficiency trade-offs on different benchmarks.

Limitations

While UNITOR demonstrates strong universality and efficiency across diverse SLM–LLM pairs, several limitations remain that we discuss explicitly to guide future work.

Reliance on SLM probability access. UNITOR requires access to the SLM's top-k next-token logits to compute its distribution-based features, and token-frequency features require a one-time tokenization of an external corpus per SLM vocabulary. This makes the method directly applicable to open-weight SLMs but precludes deployments where the small model is a black-box API that exposes only generated text.

Assumption of a meaningful capability gap. Token-level routing is beneficial only when the LLM is materially stronger than the SLM on at least a subset of tokens; otherwise, the router degenerates toward using a single model. UNITOR inherits this assumption and is not designed for pairs with near-identical capability or for settings where the SLM is selectively stronger than the LLM (e.g., domain-specialized small models).

Language and domain coverage of training data. Our routing labels are generated from Bespoke-Stratos-17k, and our token-frequency tables are built from FineWeb (Penedo et al., 2024). Both corpora are predominantly English. Although the underlying distribution and token-identity features are in principle language-agnostic, we do not empirically evaluate UNITOR on non-English or code-switched generation, and divergence rates may differ in those settings. A broader multilingual study is left to future work.

References

Lingjiao Chen, Matei Zaharia, and James Zou. 2024. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*. Featured Certification.

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. 2025. Learning to route LLMs with confidence tokens. In *Forty-second International Conference on Machine Learning*.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. 2024. Hybrid LLM: Cost-efficient and quality-aware query routing. In *The Twelfth International Conference on Learning Representations*.

Tao Feng, Yanzhen Shen, and Jiaxuan You. 2025. Graphrouter: A graph-based router for LLM selections. In *The Thirteenth International Conference on Learning Representations*.

Tianyu Fu, Yi Ge, Yichen You, Enshu Liu, Zhihang Yuan, Guohao Dai, Shengen Yan, Huazhong Yang, and Yu Wang. 2025a. R2r: Efficiently navigating divergent reasoning paths with small-large model token routing. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Yichao Fu, Rui Ge, Zelei Shao, Zhijie Deng, and Hao Zhang. 2025b. Scaling speculative decoding with lookahead reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2025. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*.

Bespoke Labs. 2025. Bespoke-stratos: The unreasonable effectiveness of reasoning distillation. www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation. Accessed: 2025-01-22.

Nguyen-Khang Le, Truong Dinh Do, and Le-Minh Nguyen. 2025. SPECTRA: Faster large language model inference with optimized internal and external speculation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14015–14034, Vienna, Austria. Association for Computational Linguistics.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7421–7432, Miami, Florida, USA. Association for Computational Linguistics.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2025. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. 2025. Reward-guided speculative decoding for efficient LLM reasoning. In *Forty-second International Conference on Machine Learning*.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2024. Routing to the expert: Efficient reward-guided ensemble of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1964–1974, Mexico City, Mexico. Association for Computational Linguistics.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, and 1 others. 2024. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pages 932–949.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2025. RouteLLM: Learning to route LLMs from preference data. In *The Thirteenth International Conference on Learning Representations*.

Rui Pan, Yinwei Dai, Zhihao Zhang, Gabriele Oliaro, Zhihao Jia, and Ravi Netravali. 2025. Specreason: Fast and accurate inference-time compute via speculative reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Kv Aditya Srivatsa, Kaushal Maurya, and Ekaterina Kochmar. 2024. Harnessing the power of multiple minds: Lessons learned from llm routing. In *Proceedings of the Fifth Workshop on Insights from Negative Results in NLP*, pages 124–134.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. 2023. Spectr: Fast speculative decoding via optimal transport. *Advances in Neural Information Processing Systems*, 36:30222–30242.

Hemish Veeraboina. 2023. AIME Problem Set 1983–2024. https://www.kaggle.com/datasets/hemishveeraboina/aime-problem-set-1983-2024. Kaggle dataset.

Xinyuan Wang, Yanchi Liu, Wei Cheng, Xujiang Zhao, Zhengzhang Chen, Wenchao Yu, Yanjie Fu, and Haifeng Chen. 2025. MixLLM: Dynamic routing in mixed large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 10912–10922, Albuquerque, New Mexico. Association for Computational Linguistics.

Xuezhi Wang and Denny Zhou. 2024. Chain-of-thought reasoning without prompting. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. 2025. Learning harmonized representations for speculative sampling. In *International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark

Barrett, and Ying Sheng. 2024. SGLang: Efficient execution of structured language model programs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Wenhao Zheng, Yixiao Chen, Weitong Zhang, Souvik Kundu, Yun Li, Zhengzhong Liu, Eric P. Xing, Hongyi Wang, and Huaxiu Yao. 2025. CITER: Collaborative inference for efficient large language model decoding with token-level routing.

A Additional Experimental Setup

A.1 Router Architecture

Input Feature. Detailed feature construction is described in Appendix D.5. We concatenate the distribution features, the token-frequency signal, and lightweight structural features (e.g., token position). The resulting feature vector is passed through a linear projection to match the input dimensionality of the router backbone.

Router neural network backbone. The router itself is implemented as a lightweight feed-forward neural network with four stacked blocks and residual connections. Each block follows a standard expand–contract design, consisting of an input LayerNorm, a linear projection that expands the hidden dimension by a factor of four, a GELU nonlinearity, and a second linear projection that restores the original dimension. Dropout is applied to both linear layers for regularization. Residual connections link consecutive blocks to stabilize optimization. The hidden dimension is set to 1024 throughout the network. After the final block, an additional LayerNorm and a linear layer map the hidden representation to a scalar score, which is passed through a sigmoid function to produce a normalized routing probability. A fixed threshold $p_{\text{th}} \in (0, 1)$ is used to convert this probability into a binary routing decision: values above p_{th} indicate that the current token is likely to diverge and should be generated by the LLM.

A.2 Training Data

Training set. We construct router-training examples from three task families—mathematics, code, and question answering (QA)—using prompts sampled from the Bespoke-Stratos-17k collection (Labs, 2025). For each prompt, we generate a reference response with DEEPSEEK-R1-DISTILL-QWEN-32B using greedy decoding (temperature $= 0$) and a maximum generation length of 32,768 tokens. We retain only generations that produce an explicit end-of-sequence (EOS) token within this limit.

To create token-level routing labels, we use DEEPSEEK-R1-DISTILL-QWEN-1.5B as the SLM. At each token position, we compare the SLM's greedy next-token prediction against the reference token. When the prediction matches, the token is treated as non-divergent. When the prediction differs, we continue the rollout with the SLM and use QWEN2.5-72B-INSTRUCT as a verifier to determine whether the resulting continuation is semantically consistent with the reference (neutral) or represents a meaningful divergence. These verifier judgments provide the binary supervision used in the subsequent stages of our data-generation pipeline.

Validation set. We build the validation set using the same procedure and model settings as above, but with disjoint prompts. Specifically, we use the 229 Bespoke-Stratos-17k examples (across math, QA, and code) that are excluded from the router-training split. A summary of the training and validation splits is provided in Table 5.

Type	#Query	#Token	#Divergent	Div. Rate
Training	2710	15M	735K	4.9%
Validation	229	619K	32K	5.2%

Table 5: Statistics of tokens difference and divergence across the training and validation sets.

A.3 Training Scheme

Loss function. The divergence labels are imbalanced (non-divergent tokens are substantially more frequent). To account for this, we optimize a weighted binary cross-entropy loss with logits (`BCEWithLogitsLoss`). Class weights are set inversely proportional to empirical class frequencies so that mistakes on the minority class receive larger penalties.

Optimization and hyperparameters. We train the router with AdamW (β_1=0.9, β_2=0.999) using a learning rate of 1×10^{-5}, dropout rate 0.2, and weight decay 5×10^{-4}. Training is performed in float32 precision with batch size 4096. We run training for at most 50 epochs and apply early stopping with patience 10 based on validation performance (evaluated once per epoch). We select the checkpoint from the best validation epoch as the final router.

Threshold selection. After training, we calibrate the routing threshold p_{th} using the validation set. We run the router once on the pre-collected inputs (features derived from SLM logits, token-frequency statistics, and token-position features) and record the predicted divergence probabilities. We then sweep $p_{\text{th}} \in [0, 1]$ to map out the trade-off between LLM activation rate and average activated parameters (Figure 9). Since the inputs are fixed, calibration consists of a single forward pass and an inexpensive threshold sweep. At inference time, for any user-specified compute budget (e.g., a target P_{avg}), we choose p_{th} to meet the corresponding budget.

A.4 Routing System Implementation Details

Model initialization. The routing system consists of three modules: the SLM (DeepSeek-R1-Distill-Qwen-1.5B), the LLM (DeepSeek-R1-Distill-Qwen-32B), and the router. We place the SLM on a single GPU via the SGLang scheduler (`mem_fraction_static`=0.15). The LLM is served with tensor-parallel inference across two GPUs using SGLang schedulers (`mem_fraction_static`=0.7). The router runs on GPU 0 as a standalone PyTorch model, independent of the SGLang runtime.

Inference workflow. At each decoding step, the SLM proposes the next token and returns its output logits. The router converts these SLM outputs into a divergence probability. If the probability exceeds the threshold p_{th}, we route the step to the LLM: we enqueue a new request to the LLM built from the current input token IDs, initialize a scheduling batch in `EXTEND` mode, and allocate the required KV-cache memory. The system maintains prefix indices to track which tokens have been consumed by each model and to keep both model states synchronized. When the LLM produces a token, it is returned through an output queue and replaces the SLM-proposed token. A token manager tracks active sequences and termination conditions (EOS or maximum length). This loop repeats token-by-token until completion.

B Additional Experiment Results

B.1 Impact of the routing threshold

Figure 9 shows the relationship between the routing threshold p_{th} and the cost of inference. Specifically, p_{th} controls the "positive rate"—the percentage of tokens sent to the LLM—which determines the average activated parameters (P_{avg}). We tune this threshold on validation data to satisfy a compute budget (e.g., $P_{\text{avg}} \approx 6$B). While P_{avg} can vary if the test workload is significantly harder or longer, we find this deviation is negligible when validation and test data are from the same distribution. For new workloads, p_{th} can be re-calibrated using a small sample dataset.

We also examine the router's ability to catch divergence events. Lowering the threshold sends more tokens to the LLM, which improves the recall of divergent tokens. Figure 9 demonstrates that recall rises sharply with only a small increase in compute (P_{avg}). This suggests the router successfully identifies the most critical tokens first, optimizing the trade-off between compute and accuracy.

B.2 UniToR with Sampling Decoding

In this appendix, we report additional results for UniToR under *sampling* decoding. We use temperature $T = 1.0$ for both the SLM and LLM and keep all other settings identical to the main experiments in Section 4. As shown in Table 6, UniToR remains competitive under sampling: it outperforms query-level routing baselines and achieves approximately 90% of the R1-32B accuracy while using only 10.0B average activated parameters. These results indicate that UniToR's token-level routing remains effective beyond greedy decoding and transfers to stochastic generation settings.

Method	Acc.	Param. (B)	Cost (KB)
R1-1.5B	28%	1.5	17.5
R1-32B	55%	32.0	364.9
QR-SW	42%	11.1	132.0
QR-LLM	43%	11.4	127.7
QR-BERT	37%	11.3	134.5
QR-MF	38%	11.1	118.8
UniToR (ours)	50%	10.0	120.8

Table 6: AIME results under sampling decoding (temperature $T = 1.0$) for the DeepSeek-R1 model pair.

B.3 Inference Latency Breakdown

To quantify the runtime overhead introduced by UniToR, we instrument the serving stack and provide a fine-grained breakdown of end-to-end decoding latency. The routing logic is integrated directly into the SGLang serving pipeline, and all measurements are conducted on the AIME-2024 benchmark, averaged over 30 questions using three NVIDIA A6000 GPUs.

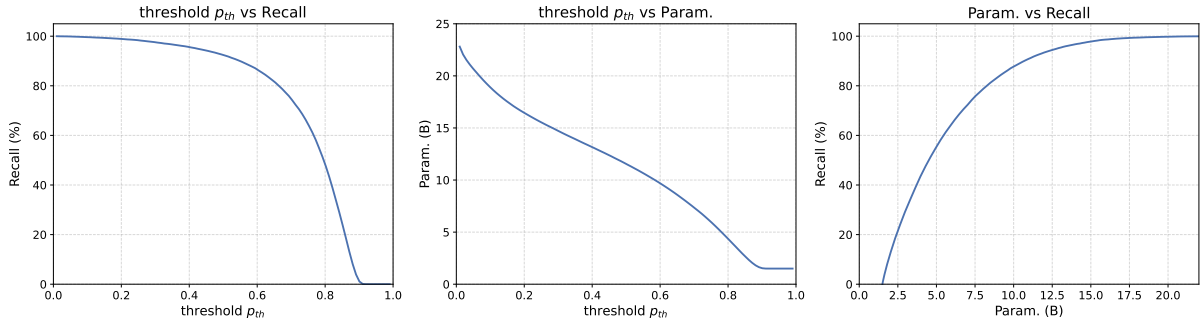


Figure 9: Relationship between the routing threshold, recall for divergent tokens, and average parameter size.

Component	#Calls	Time (s)	Share
LLM (Qwen3-32B)	34,760	2,072.70	52.43%
SLM (R1-1.5B)	222,688	1,477.42	37.37%
Router inference	222,688	201.96	5.11%
Token alignment (Alg. 1)	34,760	105.25	2.66%
Feature extraction	222,688	60.43	1.53%
Others	–	35.69	0.90%

Table 7: Component-level latency breakdown on AIME-2024 using the R1-1.5B + QWEN3-32B pair ($W = 16$, 3×A6000). Routing-specific components account for less than 10% of total runtime, confirming that routing overhead is small relative to model inference cost.

Table 7 presents the component-wise execution time. The results show that the vast majority of decoding time is spent in model forward passes, with the LLM and SLM accounting for 52.43% and 37.37% of total runtime, respectively. In contrast, the routing-specific overhead introduced by UNITOR remains minimal. Router inference contributes only 5.11% of total execution time, while token alignment accounts for 2.66%. Feature extraction incurs an additional 1.53%, yielding a combined routing-related overhead of less than 10%.

These findings confirm that the computational cost of routing decisions is negligible relative to the underlying model inference cost. Although UNITOR introduces additional control logic at each decoding step, this overhead is substantially outweighed by the reduction in expensive LLM invocations enabled by adaptive token-level switching. This demonstrates that the practical efficiency gains of UNITOR are not offset by routing complexity, making it suitable for deployment in latency-sensitive inference settings.

B.4 Resource Consumption and Efficiency Analysis

To provide a comprehensive assessment of practical deployment efficiency, we report aggregate resource consumption metrics averaged across AIME, LIVECODEBENCH, and GPQA. In addition to task accuracy, we evaluate each method in terms of parameter footprint, inference cost, computational demand measured in TFLOPs per query, and hardware requirements in terms of GPU count.

The results in Table 8 demonstrate that UNITOR achieves the most favorable accuracy–efficiency trade-off among all evaluated approaches. Compared to the pair-specific router R2R, UNITOR improves average accuracy by 2 percentage points while simultaneously reducing computational cost by approximately 6% (187 vs. 199 TFLOPs/query). This highlights that universality does not come at the expense of efficiency; instead, the proposed model-agnostic routing strategy yields both stronger performance and lower resource consumption.

B.5 Open-Ended Generation (MT-Bench)

A common concern surrounding token-level routing is that its effectiveness may be limited to verifiable reasoning tasks, where divergence labels can be derived from objective correctness signals. To evaluate whether UNITOR generalizes beyond this setting, we assess the same router—without retraining or task-specific adaptation—on MT-BENCH (Zheng et al., 2023), a benchmark of open-ended multi-turn dialogue spanning creative writing, roleplay, coding, reasoning, and subjective conversational tasks.

We evaluate an unseen cross-tokenizer model pair, R1-1.5B + QWEN3-32B, using GPT-4o as the automatic judge. As shown in Table 9, UNITOR achieves an overall score of 7.26, retaining approximately 93% of the full QWEN3-32B performance (7.81) while activating only $\sim$28% of the large model's parameters on average. This demonstrates that adaptive token-level routing remains

Method	Acc (%)	Param (B)	Cost	TFLOPs/q	#GPUs
R1-1.5B	13	1.5	41.1	82	1
R1-7B	31	7.0	154.4	309	1
R1-32B	46	32.0	567.4	1135	2
R2R	42	6.5	99.5	199	2
UNITOR	**44**	**6.1**	**93.3**	**187**	**2**

Table 8: Average resource consumption and accuracy across reasoning benchmarks (A6000 GPUs).

Model	Writing	Roleplay	Reason.	Math	Coding	Extract.	STEM	Human.	Overall
R1-1.5B	3.30	3.55	3.80	8.00	2.55	4.45	4.30	3.95	4.24
Qwen3-32B	7.60	7.30	7.20	8.25	7.50	7.85	8.20	8.55	7.81
UNITOR ($\bar{P}=6$B)	6.80	6.90	6.70	8.85	7.20	6.45	7.50	7.65	7.26

Table 9: MT-Bench results (GPT-4o judge) using the R1-1.5B + QWEN3-32B pair with a 4K maximum generation length. UNITOR retains 93% of the full LLM's performance across diverse open-ended generation tasks while using only $\sim$28% of its parameters on average. Notably, it surpasses QWEN3-32B on mathematical reasoning by selectively allocating more LLM tokens to high-divergence regions.

highly effective even in open-ended generation scenarios where correctness is inherently subjective. Performance remains consistently strong across most categories, with UNITOR staying within 0.5–1.0 points of the full LLM on writing, coding, STEM, and human preference tasks. Most notably, it surpasses QWEN3-32B on mathematical reasoning (8.85 vs. 8.25), suggesting that the router successfully identifies moments requiring stronger reasoning capacity and allocates LLM computation precisely where needed.

These results provide strong evidence that the model-agnostic routing signals learned by UNITOR capture general patterns of token-level uncertainty rather than task-specific heuristics. This allows a single universal router to transfer effectively from verifiable reasoning benchmarks to open-ended conversational generation, further supporting the practicality of UNITOR as a general-purpose inference acceleration framework.

B.6 Label Quality Validation

Since UNITOR relies on verifier-generated token-level divergence labels for supervision, the reliability of these labels is critical to the effectiveness of the routing policy. Following the human evaluation protocol introduced by Fu et al. (2025a), we conduct a manual validation study to assess the quality of our labeling pipeline. We randomly sample 500 token-level disagreement cases and recruit three graduate researchers to independently annotate each instance as either *neutral* or *divergent*. Ground-truth labels are determined by majority vote across the three annotators. A fourth independent annotator is used as the human performance baseline, enabling direct comparison between verifier predictions and human-level agreement.

Verifier	Acc.	Recall	Prec.
Human (4th annot.)	0.82	0.82	0.61
Qwen2.5-72B (**ours**)	**0.85±0.01**	**0.86±0.02**	**0.65±0.02**
Qwen2.5-7B	0.80±0.01	0.83±0.02	0.56±0.02
Qwen2.5-3B	0.75±0.01	0.72±0.02	0.50±0.02

Table 10: Verifier agreement with human-annotated token-level divergence labels on 500 sampled disagreements. QWEN2.5-72B matches or exceeds human-level agreement, validating the quality of our automated labeling pipeline.

Table 10 compares the performance of our verifier (QWEN2.5-72B) against both the human baseline and smaller QWEN2.5 variants. The results show that QWEN2.5-72B achieves 0.85 accuracy and 0.86 recall, slightly exceeding human-level agreement (0.82 accuracy and recall). It also consistently outperforms smaller verifier models across all evaluation metrics.

These results validate the reliability of our verifier-based annotation pipeline and suggest that high-capacity LLMs can provide token-level divergence supervision at a quality comparable to, or exceeding, expert human annotation. This supports the use of automated verifier labeling as a scalable alternative to expensive manual annotation.

B.7 Sensitivity to Alignment Window Size

The token-alignment mechanism in UNITOR uses a sliding context window of size W to reconcile token boundaries across heterogeneous tokeniz-

Window	Acc.	Param. (B)	Align. time (s)
$W = 16$ (ours)	**36.67**	6.31	**105.25**
$W = 32$	33.33	6.21	115.71
$W = 64$	30.00	6.13	122.66
$W = 128$	30.00	6.38	128.13
Full context	33.33	6.34	135.56

Table 11: Sensitivity analysis of alignment window size W on AIME-2024 (30 problems, R1-1.5B + QWEN3-32B). Smaller windows achieve both higher accuracy and lower alignment overhead, with $W = 16$ selected as the default.

ers. The choice of W introduces a trade-off between alignment context and computational overhead: larger windows provide more context for matching but increase alignment latency during decoding. To evaluate this trade-off, we sweep $W \in \{16, 32, 64, 128\}$ as well as a full-context alignment setting on AIME-2024 using the R1-1.5B + QWEN3-32B pair over 30 problems. Table 11 reports accuracy, effective parameter usage, and alignment latency.

The results show that routing performance is robust across a broad range of window sizes, indicating that UNITOR does not require extensive tuning of this hyperparameter. However, smaller windows consistently provide better efficiency. In particular, $W = 16$ achieves the highest accuracy (36.67) while also yielding the lowest alignment latency (105.25s), offering the best overall accuracy–efficiency trade-off.

Interestingly, increasing the window size does not improve routing quality. In several cases, larger windows slightly degrade performance despite incurring higher computational overhead. We hypothesize that this is because token divergence decisions are predominantly determined by local contextual information, making additional long-range alignment context unnecessary and potentially introducing noise into token matching. Based on these findings, we adopt $W = 16$ as the default setting throughout all experiments.

C Prompts and Response Examples

C.1 Verifier Prompt

As described in Appendix A.2, we use a verifier model (Qwen2.5-72B-Instruct) to judge whether a mismatch between the reference continuation and the SLM continuation is semantically *neutral* or constitutes a meaningful *divergence*. Our verifier prompt explicitly marks the disagreement span and asks the model to decide whether the difference changes the meaning, reasoning steps, or correctness of the answer (Text 1). The prompt also requests a short justification and includes a few in-context examples to reduce ambiguity and encourage consistent labeling.

C.2 Qualitative Response Example

To provide an intuitive illustration of UNITOR routing, we include a representative AIME problem (Text 2) together with outputs from the SLM (R1-1.5B), the LLM (R1-32B), and UNITOR. Example generations are shown in Text 3–Text 5.

Text 2: Example Question (AIME)

Alice and Bob play the following game. A stack of n tokens lies before them. The players take turns with Alice going first. On each turn, the player removes either 1 token or 4 tokens from the stack. Whoever removes the last token wins. Find the number of positive integers n less than or equal to 2024 for which there exists a strategy for Bob that guarantees that Bob will win the game regardless of Alice's play.

In this example, the SLM and LLM arrive at different final answers due to a divergence in the intermediate reasoning. UNITOR detects the divergence point and routes the corresponding token(s) to the LLM, steering generation back to the higher-quality reasoning path and producing a final answer that matches the LLM.

D More on feature analysis

D.1 Distribution-based Features

- **Max Probability**: The probability assigned to the top-1 prediction, indicating model confidence.
- **Probability Margin**: $p_1 - p_2$, the difference between top-1 and top-2 probabilities.

D.2 Token-identity-based features

- **Normalized Vocabulary Position**: The token ID divided by vocabulary size, where lower values correspond to more common tokens in the base vocabulary.
- **Token Character Length**: The number of characters in the decoded token.

Text 1: Verifier Prompt

```
Task:
Determine if the divergence between Sentence 1 and Sentence 2 affects the
    meaning, reasoning, logic, or conclusion derived from them.

Instructions:
- The marker "*" indicates where the sentences diverge. It is not part of the
    original text.
- Assess whether this divergence changes the meaning, reasoning, logic, or
    conclusions, or if it introduces new information or contradictions.

Output 1 if:
- The divergence causes a change in meaning, reasoning, logic, or conclusions.
- It introduces new information, shifts focus, or contradicts prior facts.
- The sentences follow different reasoning paths or focus on different aspects.

Output 0 if:
- The divergence is superficial and does not affect meaning, reasoning, logic,
    or conclusions.
- Both sentences follow the same reasoning path or lead to the same conclusion.

Reasoning:
Provide a brief explanation of how the divergence impacts (or does not impact)
    meaning, reasoning, logic, or conclusions.

---
Example 1 (Same - 0):
Sentence 1:
"The ratio of adults to total people <is*> now 11/25."
Sentence 2:
"The ratio of adults to total people <chang*>ed from 5/12 to 11/25 after adding
    50 people."
Output: 0
Reasoning: The change from "is" to "changed" does not affect the overall
    meaning, reasoning, logic, or conclusions.

Example 2 (Different - 1):
Sentence 1:
"Let's solve this using <integration> by parts."
Sentence 2:
"Let's solve this using <u->substitution."
Output: 1
Reasoning: The change in method (from integration by parts to substitution)
    alters the reasoning and approach to solving the problem.

---
Now complete the task:
Common Context:
{CONTEXT}

Sentence 1:
{SENTENCE1}

Sentence 2:
{SENTENCE2}

Answer (Output: <0 or 1>)
Reasoning:
```

Text 3: R1-1.5B Response

Okay, so Alice and Bob are playing this game where they take turns removing either 1 or 4 tokens from a stack. Alice goes first, and the person who takes the last token wins. I need to figure out how many positive integers $n \leq 2024$ allow Bob to have a winning strategy, no matter how Alice plays. Hmm, this sounds like a combinatorial game theory problem. I remember something about winning and losing positions in such games.

Let me start by understanding the rules better. Each player can remove 1 or 4 tokens on their turn. So, if there are n tokens, a player can take 1 or 4 tokens, and the next player does the same until

...

$n = 15$: Alice can take 1, leaving 14 for Bob. $n = 14$ is a winning position for Bob, so Bob can win. Alternatively, Alice can take 4, leaving 11 for Bob. $n = 11$ is a winning position for Bob, so Bob can win. So, $n = 15$ is a losing position for Alice, meaning Bob can win.

...

Final Answer

$\boxed{N/A}$

Text 4: R1-32B Response

Okay, so Alice and Bob are playing this game where they take turns removing either 1 or 4 tokens from a stack of n tokens. Alice goes first, and whoever takes the last token wins. I need to figure out how many positive integers $n \leq 2024$ make Bob have a winning strategy, meaning no matter how Alice plays, Bob can force a win.

Hmm, let me think. This sounds like a combinatorial game theory problem. I remember that in such games, positions can often be classified as either winning or losing positions. A winning position means the current player can force a win, while a losing position means the current player will lose if the opponent plays optimally.

So, in this case, if n is a losing position, then the player whose turn it is will lose if the opponent plays correctly.

...

$n = 15$: The player can take 1, leaving $n = 14$ (W), or take 4, leaving $n = 11$ (W). So, all moves lead to the opponent's W. Therefore, $P(15)$ is L.

...

Final Answer

$\boxed{809}$

Text 5: UniToR Response

Okay, so Alice and Bob are playing this game where they take turns removing either 1 or 4 tokens from a stack of n tokens. Alice goes first, and whoever takes the last token wins. I need to figure out how many positive integers $n \leq 2024$ make Bob have a winning strategy, meaning no matter how Alice plays, Bob can always win.

Hmm, let me think. This sounds like a combinatorial game theory problem. I remember that in such games, positions can often be classified as either winning or losing positions. A winning position means the current player can force a win no matter what the opponent does, while a losing position means the current player will lose if the opponent plays optimally.

...

- $P(15)$: The player can take 1, leaving $P(14)$, which is winning for the opponent, or take 4, leaving $P(11)$, which is winning for the opponent. So, both moves leave the opponent in a winning position. Therefore, $P(15)$ is losing.

...

Final Answer

$\boxed{809}$

- **Token Frequency**: The empirical frequency of the token in the training data.
- **Token Type Indicators**: Binary features for punctuation, digits, whitespace, and special tokens.

D.3 Position-based features

- **Sequence Position**: Normalized position of the token within the generation sequence.

D.4 Analysis

Figure 10 summarizes the predictive power of individual features. Distribution-based features exhibit strong discriminative ability, confirming that model uncertainty is the primary signal for predicting token-level divergence. Among the remaining features, token character length and normalized vocabulary position achieve moderate predictive performance.

Interestingly, we observe that vocabulary position (i.e., token ID) is a strong indicator of divergence: tokens with higher ID values tend to exhibit higher divergence rates. However, this effect is highly dependent on the tokenizer's vocabulary construction and is therefore unlikely to generalize reliably across different models. In contrast, sequence position features (e.g., log position and relative position) yield low AUC scores, indicating that a token's location within the generated sequence has a limited impact on its likelihood of divergence.

D.5 Feature Construction

Distribution features. At each decoding step, we extract a compact set of model-agnostic features from the SLM output distribution. Specifically, we select the top-100 logits, apply softmax to obtain probabilities, and use the resulting vector (and summary statistics) as distribution features.

Token-identity features. To capture token rarity, we incorporate token-frequency statistics. For each SLM, we build a frequency lookup table by tokenizing 5M samples from FineWeb (Penedo et al., 2024); this preprocessing is performed once per SLM and reused across experiments. During infer-

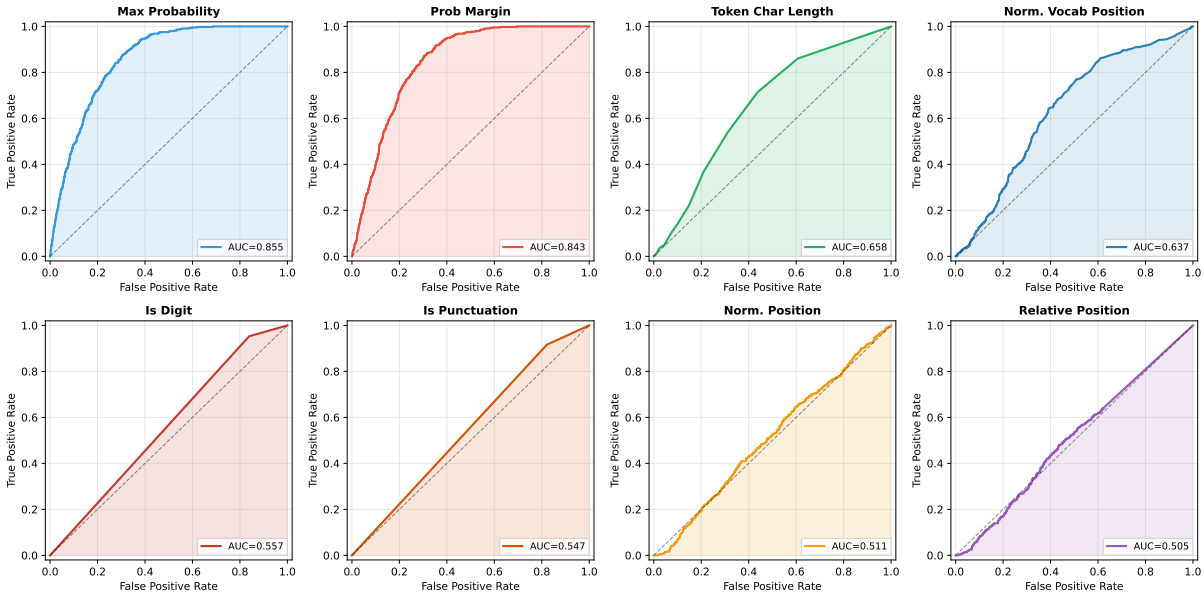


Figure 10: ROC curves and AUC of features.

ence, we retrieve the frequency of the generated token from this table. We also include token-type indicators (e.g., digit, punctuation, alphabetic), computed by decoding the token and applying simple rules.

Other lightweight features. We include additional features derived from generation metadata, such as the token's relative position in the context.

E Background on Routing Strategies

E.1 Query-level Routing

Let $\mathcal{Q}$ be the query space and $\mathcal{A}$ the output space. Query-level routing selects a single model to generate the entire response:

$$\mathcal{R}_{\text{query}} : \mathcal{Q} \to \{M_s, M_l\}.$$

Given a query $q \in \mathcal{Q}$, a routing function $\mathcal{R}_{\text{query}}$ assigns the query to either the SLM or LLM, and the selected model $M = \mathcal{R}_{\text{query}}(q)$ generates the complete response $a \in \mathcal{A}$.

E.2 Step-level Routing

Step-level routing assumes the output can be decomposed into a sequence of reasoning steps $\mathbf{a} = (r_1, \ldots, r_T)$. Given a query q, at step t, the SLM first proposes a candidate step $\hat{r}_t = M_s(q, r_{<t})$, where $r_{<t} = (r_1, \ldots, r_{t-1})$ denotes the preceding steps. The LLM evaluates this proposal using a scoring function $\mathcal{E}_{\theta_l}$, producing a confidence score $e_t = \mathcal{E}_{\theta_l}(q, r_{<t}, \hat{r}_t)$. Given a threshold τ, the step is accepted if $e_t \geq \tau$; otherwise, the LLM generates the step:

$$r_t = \begin{cases} \hat{r}_t, & \text{if } e_t \geq \tau, \\ M_l(q, r_{<t}), & \text{if } e_t < \tau. \end{cases}$$

E.3 Token-level Routing

Token-level routing enables dynamic model selection at each decoding step. Given a query q with token representation $\mathbf{x} = (x_1, \ldots, x_n)$, decoding proceeds autoregressively. At step t, a routing function first selects the model:

$$m_t = \mathcal{R}_{\text{token}}(\mathbf{x}, y_{<t}; \theta_s) \in \{M_s, M_l\},$$

where $y_{<t} = (y_1, \ldots, y_{t-1})$ denotes the previously generated tokens. The selected model m_t then generates the next token via greedy decoding:

$$y_t = \arg\max_y P_{m_t}(y \mid \mathbf{x}, y_{<t}).$$

This fine-grained routing allows the system to leverage the SLM for simpler tokens while invoking the LLM only when necessary.

F Algorithms

Algorithm 1 Token Alignment When Switching SLM/LLM

Require: SLM current sequence $\mathbf{s} = (s_1, \ldots, s_t)$
Require: LLM current sequence $\mathbf{l} = (l_1, \ldots, l_u)$
Require: window size W
{Extract window and decode to text}
$\mathbf{s}_{\text{win}} \leftarrow (s_{\max(1,t-W+1)}, \ldots, s_t)$
$x \leftarrow \text{Decode}_{\mathcal{V}_S}(\mathbf{s}_{\text{win}})$
{Re-encode under LLM vocabulary}
$\mathbf{l}_{\text{new}} \leftarrow \text{Encode}_{\mathcal{V}_L}(x)$
$m \leftarrow |\mathbf{l}_{\text{new}}|$
{Find longest suffix–substring overlap}
$k^* \leftarrow 0,\ j^* \leftarrow 0$
for $k = 1$ **to** $\min(u, m)$ **do**
    for $j = 1$ **to** $m - k + 1$ **do**
        if $\mathbf{l}[u - k + 1 : u] = \mathbf{l}_{\text{new}}[j : j + k - 1]$ **then**
            $k^* \leftarrow k,\ j^* \leftarrow j$
        end if
    end for
end for
{Append only unmatched continuation}
$\mathbf{l} \leftarrow (\mathbf{l}, \mathbf{l}_{\text{new}}[j^* + k^* : m])$
{Append unmatched suffix to LLM sequence}
$\mathbf{l} \leftarrow (\mathbf{l}, \mathbf{l}_{\text{new}}[k{+}1{:}])$
return Updated LLM token sequence $\mathbf{l}$