

Inference Scaling of LLM Ensembling: Bridging Token Spaces with Token Translation

Tianxin Wei¹, Zhichen Zeng¹, Ruizhong Qiu¹, Zhining Liu¹, Xuying Ning¹, Xinrui He¹, Wenxuan Bao¹, Xiao Lin¹, Qi He², Xianfeng Tang², Hanghang Tong¹, Jingrui He¹

¹University of Illinois Urbana-Champaign ²Microsoft
{twei10, jingrui}@illinois.edu

Abstract

Large language models (LLMs) exhibit diverse strengths and weaknesses across tasks, motivating recent efforts to ensemble multiple models to harness their complementary capabilities, boosting test-time performance. While model diversity and capability are known to influence ensemble effectiveness, a persistent challenge in LLM ensembling arises from mismatched tokenizer vocabularies. Existing alignment strategies typically rely on token-level embeddings or string-level heuristics of tokens, overlooking the tokenizer priors embedded during LLM pretraining. Specifically, tokenizers such as Byte-Pair Encoding (BPE) and Unigram are constructed by statistically analyzing large pretraining corpora to identify frequent subword units, and they tokenize text using greedy or probabilistic algorithms that reflect these learned subword distributions. In this work, we propose a novel and remarkably simple *Token Translation* (ToT) method that explicitly leverages these tokenizer priors to bridge heterogeneous token spaces. Our method is lightweight, requiring only a few lines of code, pre-computable, and highly efficient at inference. To further enhance robustness, we incorporate token-level model uncertainty to dynamically reweight each model’s contribution during decoding. Extensive evaluations across diverse model combinations and tasks demonstrate that our method consistently outperforms existing ensembling baselines.

1 Introduction

Large language models (LLMs) (Touvron et al., 2023; OpenAI, 2023) have demonstrated strong performance across tasks such as question answering (Kamalloo et al., 2023), summarization (Zhang et al., 2024), and reasoning (Jaech et al., 2024; Guo et al., 2025). Owing to differences in data, architectures, and training objectives, LLMs exhibit complementary strengths and weaknesses (Jiang et al., 2023b; Yao et al., 2025). This motivates *test-time ensembling*, a practical approach to combine multiple pretrained LLMs for improved performance and adaptability without additional fine-tuning.

Test-time ensembling methods generally fall into two categories: response-level (Feng et al., 2024; Jiang et al., 2023b; Lu et al., 2023) and logit-level (or token-level) ensembling (Yao et al., 2025; Yu et al., 2024; Huang et al., 2024; Phan et al., 2025). Response-level approaches select among complete outputs or route inputs to a model, offering a simple but coarse way to combine LLMs. Logit-level approaches instead operate directly on token (or sub-token) distributions during generation, enabling finer-grained integration of model predictions. However, they face a key challenge: *tokenizer mismatch*, as models often adopt heterogeneous tokenization vocabularies, making the alignment of token-level predictions nontrivial.

LLMs adopt distinct tokenization schemes (e.g., BPE (Sennrich et al., 2015), Unigram (Kudo, 2018)), each trained on a separate pretraining corpus. These schemes differ in segmentation granularity, vocabulary, and subword decomposition, resulting in heterogeneous token spaces. Such mismatches prevent alignment of token-level distributions across models and can lead to sparse mappings, especially when vocabularies only partially overlap.

Table 1: Comparisons of LLM ensemble methods. **Left panel:** Comparison across five methodological dimensions. **Right panel:** Radar plot comparing ToT with the two base LLMs and the strongest ensemble baseline across six benchmarks.

Method	Mismatch Handling	Minimal Overhead	Tokenizer Priors	Uncertainty Aware	Bidirectional
LLM-Blender (Jiang et al., 2023b)	✗	✗	✗	✗	✗
GAC (Yu et al., 2024)	✗	✓	✗	✗	✗
DeepEn (Huang et al., 2024)	✓	✗	✗	✗	✗
CES (Phan et al., 2025)	✓	✗	✗	✗	✗
UniTE (Yao et al., 2025)	✓	✓	✗	✗	✗
ToT (Ours)	✓	✓	✓	✓	✓

Prior works attempted to align tokens across models by comparing token embedding similarities (Huang et al., 2024) or using string-level heuristics such as string distance (Wan et al., 2024a). They fail due to inconsistent token boundaries and lack of grounded semantic correspondence. However, these strategies often overlook a critical source of information: *tokenizer priors*, the statistical patterns captured during tokenizer training on large corpora. Widely deployed tokenizers like BPE and Unigram segment text by greedily selecting high-frequency subword units, and this behavior implicitly encodes valuable knowledge about how linguistic input is represented in each LLM.

To address the challenge of tokenizer mismatch in token-level LLM ensembling, we propose a simple and highly effective alignment strategy that leverages *tokenizer priors*, i.e., the statistical patterns implicitly captured by tokenizers during training on large corpora. Our key idea is to align heterogeneous token spaces through an operation we term ***token translation***: for any token from a source model, we first decode it into its original text form using the source tokenizer, and then re-encode this text using the target model’s tokenizer. This reveals how the target tokenizer interprets the same content and provides a grounded correspondence between token spaces. Intuitively, it approximates the token sequence the target model is likely to generate when expressing the same underlying content, capturing both interpretation and generation preferences. To enhance alignment quality, we further introduce a bidirectional translation scheme that integrates alignments in both directions to capture complementary structures. This approach is easy to implement, pre-computable for efficiency, and enables robust token-level fusion across models with diverse vocabularies. Detailed comparisons of existing methods are provided in Table 1, with the left panel showing the method-level characteristics and the right panel reporting quantitative performance. As highlighted, our proposed ToT method consistently checks all the desired properties and achieves the best overall results.

After aligning token predictions into a shared space, we address the challenge of aggregating outputs from models with varying reliability through an ***uncertainty-aware ensembling*** mechanism. In practice, different models may exhibit varying levels of confidence depending on the context, and treating them equally can dilute stronger signals. To address this, we dynamically reweigh each model’s token distribution based on prediction entropy: models with lower uncertainty get greater influence, while uncertain ones are downweighted. This adaptive weighting allows the ensemble to emphasize more reliable token-level signals.

We validate our method through extensive experiments on multiple model combinations and tasks. Results show that our approach consistently outperforms existing LLM ensembling methods, and further scales as more capable LLMs are included. Our main contributions are as follows:

- We highlight the untapped potential of tokenizer priors for bridging heterogeneous token spaces and addressing tokenizer mismatch, a key bottleneck in LLM ensembling.
- We introduce a bidirectional token alignment method ToT based on *token translation*, a simple yet effective operation that aligns tokens across models by leveraging their tokenizer behaviors.
- We propose an uncertainty-aware ensembling strategy that adaptively reweighs model contributions at each token step based on prediction confidence, enhancing quality.

- We demonstrate that our method consistently improves performance across diverse tasks and model combinations, achieving an average absolute gain of 5.95 points, offering a scalable, well-founded solution for training-free LLM ensembling.

2 Related Works

Ensembling LLMs is a promising way to improve accuracy, robustness, and calibration. Existing methods mainly fall into three categories: *response-level*, *logit-level*, and *training-time* ensembling. Our work focuses on token-level fusion and proposes a lightweight *token translation* strategy to handle tokenizer mismatch while preserving fine-grained generation.

Response-level ensembling aggregates complete outputs from different models. Early methods such as PairRanker (Jiang et al., 2023b) and routing-based ensembles (Feng et al., 2024; Lu et al., 2023) select the best response based on reranking or input-conditioned selection. More recent techniques like SweetSpan (Xu et al., 2024a) attempt to fuse spans across responses. While conceptually simple, these approaches ignore token-level information and often struggle to generalize across tasks and model combinations.

Logit-level ensembling operates at a finer granularity by aggregating token logits during generation. We use this term broadly to include variants that operate on tokens, sub-tokens (bytes), or token spans. UniTE (Yao et al., 2025) uses top- k token union for alignment, while GAC (Yu et al., 2024) and DeepEn (Huang et al., 2024) rely on dense mapping matrices between vocabularies to perform projection-based fusion. Span-level ensembling (Xu et al., 2025) merges consecutive tokens into larger segments, but substantially increases computation cost. Sub-token-level ensembling (Gu et al., 2024; Phan et al., 2025) such as CES, instead decomposes tokens into smaller units, but suffers from a lack of tokenizer priors, often requiring expensive search. Our method avoids these issues by leveraging tokenizer priors to construct sparse, grounded token mappings via a translation mechanism.

Training-time ensembling focuses on distilling multiple model outputs into a student model. FuseLLM (Wan et al., 2024a) and FuseChat (Wan et al., 2024b) perform logit-based distillation, while EVA (Xu et al., 2024b) and EnsW2S (Agrawal et al., 2024) learn vocabulary projection layers to integrate predictions. Ent (Ruan et al., 2022) also explore weighted ensembling, but their weights are learned during training, unlike our test-time aggregation of frozen heterogeneous LLMs. While effective for model compression, these approaches require model retraining and full access to parameters, making them inapplicable in test-time scenarios where models are frozen or proprietary.

3 Methodology

In this section, we present our test-time ensembling framework for token-level fusion across LLMs with heterogeneous tokenizers. We first define the *token alignment* problem arising from vocabulary mismatch, then introduce *token translation* to align token spaces, along with a *bidirectional* variant for added robustness. Finally, we describe an *uncertainty-aware* ensembling strategy that reweights model outputs by confidence. An overview is shown in Figure 1.

3.1 Problem Formulation: Token Alignment Across Heterogeneous Tokenizers

Token-level ensembling across LLMs requires the ability to align token distributions across models with heterogeneous tokenizers. We follow the setting adopted by prior work such as UniTE (Yao et al., 2025), GAC (Yu et al., 2024), and DeepEn (Huang et al., 2024), where a designated *main model* serves as the reference, and one or more *assist models* provide auxiliary signals. At each decoding step, each model outputs a probability distribution over its own vocabulary: $\mathbf{p}^{(\text{main})} \in \Delta^{|\mathcal{V}_{\text{main}}|}$ of tokenizer Tok_{main} and $\mathbf{p}^{(\text{assist})} \in \Delta^{|\mathcal{V}_{\text{assist}}|}$ of tokenizer $\text{Tok}_{\text{assist}}$, where $\Delta^{|\mathcal{V}|}$ denotes the probability simplex, i.e., the set of non-negative vectors in $\mathbb{R}^{|\mathcal{V}|}$ that sum to one.

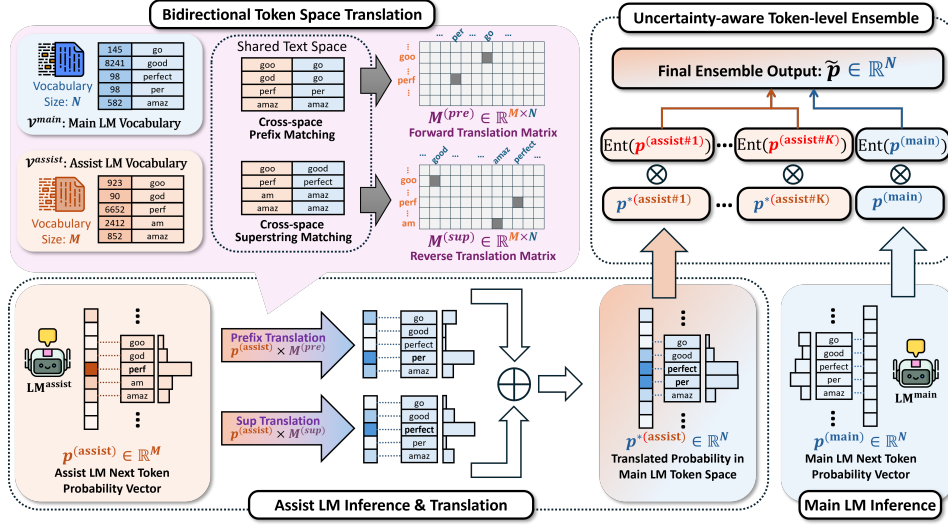


Figure 1: Illustration of our **ToT** framework for token-level ensembling across heterogeneous LLMs. Assist model predictions are translated into the main model’s token space via bidirectional token translation using tokenizer priors. Final predictions are fused using uncertainty-aware weighting for robust ensemble decoding.

The goal is to construct a *mapping matrix* $M \in \mathbb{R}^{|\mathcal{V}_{\text{assist}}| \times |\mathcal{V}_{\text{main}}|}$ for each assist model, where $M_{i,j}$ indicates the alignment between assist token $v_i^{(\text{assist})}$ and main token $v_j^{(\text{main})}$. By performing row-normalization on M , we obtain the transition matrix $T \in \mathbb{R}^{|\mathcal{V}_{\text{assist}}| \times |\mathcal{V}_{\text{main}}|}$, where $T_{i,j} = \Pr(v_j^{(\text{main})} | v_i^{(\text{assist})})$ is the probability of transiting assist token $v_i^{(\text{assist})}$ to main token $v_j^{(\text{main})}$. Using T , the assist model’s probability is projected into the main vocabulary space via $T^\top p^{(\text{assist})}$. Finally, these aligned distributions are ensembled by their linear combination.

3.2 Token Translation via Decode $\rightarrow$ Encode

Example: Failure of Similarity-based Alignment

Suppose the assist tokenizer produces the token *good*, while the main tokenizer includes *go*, *goo*, and *nice*. The intended continuation is *goods*. A surface-similarity heuristic might align *good* to *goo* or even *nice* due to lexical or semantic resemblance. However, this ignores how the main tokenizer actually segments and generates: in practice it may begin with *go*, then continue with *ods* (e.g., *go+od+s*). Mapping *good* to *goo* or *nice* biases decoding toward irrelevant continuations such as *goose* or *nicely*, deviating from the desired *goods*. By respecting tokenizer priors, token translation aligns *good* to the main prefix *go*, preserving the ideal generative priors.

Given the problem setting above, the key challenge lies in constructing a meaningful and efficient transition matrix T , as the vocabularies $\mathcal{V}_{\text{main}}$ and $\mathcal{V}_{\text{assist}}$ from different LLM families are often only partially overlapping. Traditional approaches based on string matching or embedding similarity frequently fail due to inconsistent token boundaries and the absence of semantic correspondence, often producing misleading alignments as shown in the example above. To address this, we propose an alignment strategy based on *tokenizer priors*, which leverages the statistical segmentation behavior inherently learned during tokenizer training, and further introduce a simple yet effective mechanism, *token translation*, that uses these priors to construct T and establish semantic correspondences across heterogeneous vocabularies.

Tokenizers such as BPE and Unigram are trained on large corpora to segment text into frequent and meaningful subword units. These statistical segmentation behaviors encode valuable priors about how each tokenizer decomposes natural language. Unlike model weights that are optimized for downstream objectives, tokenizer behavior is fixed and deterministic once trained. Therefore, it provides a stable and semantically grounded signal for reasoning about how different models represent and generate language. Our key insight is that tokenizer segmentation behavior has already encoded a deterministic causal structure for next-token generation, and leveraging these priors leads to more reliable cross-model alignment. A formal justification and analysis are provided in Appendix D.

Method. To account for such discrepancies, we propose to align tokens through their tokenizer-internal decoding preferences. Specifically, for assist token id i , we **decode** it into its raw text form $v_i^{(\text{assist})}$ using the assist tokenizer. We then re-**encode** $v_i^{(\text{assist})}$ using the main tokenizer to obtain a sequence $[t_{i,1}^{(\text{main})}, \dots, t_{i,m}^{(\text{main})}]$ of token ids from $\mathcal{V}_{\text{main}}$. We map the assist token id i to the first token id $t_{i,1}^{(\text{main})}$ in this sequence:

$$M_{i,j} = \begin{cases} 1 & \text{if } t_{i,1}^{(\text{main})} = j, \\ 0 & \text{otherwise.} \end{cases}$$

Intuitively, this token captures the initial prefix the main model is most likely to generate for the given content, providing a generation-compatible alignment.

This procedure is repeated for all tokens in $\mathcal{V}_{\text{assist}}$, yielding a sparse mapping matrix $M \in \mathbb{R}^{|\mathcal{V}_{\text{assist}}| \times |\mathcal{V}_{\text{main}}|}$. Each row has a single nonzero entry: $M_{i,j} = 1$ if $v_i^{(\text{assist})}$ aligns with $v_j^{(\text{main})}$. This sparsity makes the alignment efficient to store and apply, and it reflects a semantically informed, generation-aware projection from the assist to the main vocabulary spaces.

3.3 Bidirectional Translation for Complementary Alignment

While unidirectional token translation offers a simple and effective means of token alignment, it might be limited by the asymmetry. In particular, it may favor shorter prefixes or overlook longer and semantically richer tokenizations that arise in the reverse direction. To enhance alignment robustness and capture complementary structures, we propose a *bidirectional translation* scheme.

The key idea is to consider both translation directions between the assist and main vocabularies and combine the outcomes. Specifically, in addition to the forward mapping $M^{(\text{pre})}$ where each assist token is mapped to the first token generated by the main tokenizer (as done in Section 3.2), we also obtain a reverse mapping $M^{(\text{sup})}$ where each main token is decoded and then re-tokenized using the assist tokenizer. Intuitively, $M^{(\text{pre})}$ maps the assist token to its *prefix* main token, while $M^{(\text{sup})}$ maps the assist token to its *superstring* main tokens. These complementary mappings are further fused to yield a more robust bidirectional alignment.

Formally, we use $\text{Pre}(v_i^{(\text{assist})})$ to denote the mapped prefix of $v_i^{(\text{assist})}$ and $\text{Sup}(v_i^{(\text{assist})})$ to denote the set of mapped superstrings of $v_i^{(\text{assist})}$. Define the forward mapping matrix $M^{(\text{pre})}$ such that $M_{i,j}^{(\text{pre})} = 1$ if $v_j^{(\text{main})} = \text{Pre}(v_i^{(\text{assist})})$ under token translation. Similarly, define the reverse mapping matrix $M^{(\text{sup})}$ such that $M_{i,j}^{(\text{sup})} = 1$ if $v_j^{(\text{main})} \in \text{Sup}(v_i^{(\text{assist})})$ under reverse token translation. Instead of taking the maximum of the two, we apply a weighted combination:

$$M^{(\text{assist} \rightarrow \text{main})} = M^{(\text{pre})} + \alpha \cdot M^{(\text{sup})},$$

where $\alpha \in [0, 1]$ is a tunable weight that controls the influence of reverse translation. We set $\alpha < 1$ to prioritize prefix translation, ensuring alignment remains consistent with the main model’s generative behavior, while still allowing reverse direction signals to contribute additional coverage for longer or nested token patterns. The algorithm is shown in Alg. 1.

This strategy reflects the intuition that forward translation maintains the semantics most compatible with the main model’s own vocabulary, while reverse translation can introduce complementary patterns that improve the recall. This is particularly useful when the reverse tokenizer favors generating longer subwords that align with semantic units absent in the forward match.

To generate a valid transition matrix, the fused mapping matrix is further row-normalized to produce a transition matrix $T^{(\text{assist} \rightarrow \text{main})}$, that is

$$T^{(\text{assist} \rightarrow \text{main})} = \text{diag}(M^{(\text{assist} \rightarrow \text{main})} \mathbf{1})^{-1} M^{(\text{assist} \rightarrow \text{main})}.$$

Based on the translation matrix transition matrix $T^{(\text{assist} \rightarrow \text{main})}$, we can decompose the token translation process into two parts, including translating to prefix and translating to substring, as follows

$$\begin{aligned} p^{(\text{assist} \rightarrow \text{main})}(i) &= \left(T^{(\text{assist} \rightarrow \text{main})^\top} p^{(\text{assist})} \right)(i) \\ &= \sum_{v_j^{(\text{assist})}} \Pr(\text{Pre}) \Pr(v_i^{(\text{main})} | v_j^{(\text{assist})}, \text{Pre}) \Pr(v_j^{(\text{assist})}) + \sum_{v_j^{(\text{assist})}} \Pr(\text{Sup}) \Pr(v_i^{(\text{main})} | v_j^{(\text{assist})}, \text{Sup}) \Pr(v_j^{(\text{assist})}) \\ &= \sum_j \left\{ \underbrace{\frac{1}{1+\alpha N}}_{\Pr(\text{Pre})} \underbrace{\mathbb{1}(v_i^{(\text{main})} = \text{Pre}(v_j^{(\text{assist})}))}_{\Pr(v_i^{(\text{main})} | v_j^{(\text{assist})}, \text{Pre})} \underbrace{p^{(\text{assist})}(j)}_{\Pr(v_j^{(\text{assist})})} + \underbrace{\frac{\alpha N}{1+\alpha N}}_{\Pr(\text{Sup})} \underbrace{\text{Unif}(\mathbb{1}(v_i^{(\text{main})} \in \text{Sup}(v_j^{(\text{assist})})))}_{\Pr(v_i^{(\text{main})} | v_j^{(\text{assist})}, \text{Sup})} \underbrace{p^{(\text{assist})}(j)}_{\Pr(v_j^{(\text{assist})})} \right\}, \end{aligned}$$

where $N = |\mathbb{1}(v_i^{(\text{main})} \in \text{Sup}(v_j^{(\text{assist})}))|$ is the number of target tokens that are the prefix of $v_i^{(\text{main})}$. The above equation provides insights on the translation process: with probability $\Pr(\text{Pre}) = \frac{1}{1+\alpha N}$, the probability mass on assist token is transited to the mapped prefix main token, and with probability $\Pr(\text{Sup}) = \frac{\alpha N}{1+\alpha N}$, the probability mass on assist token is uniformly transited to the mapped superstring main tokens.

3.4 Uncertainty-Aware Token-Level Ensembling

After establishing a shared alignment space, we ensemble token-level predictions from multiple LLMs with an *uncertainty-aware fusion strategy*.

Different models exhibit varying levels of confidence depending on the input. Treating all models equally may amplify noise or disproportionately emphasize low-confidence predictions. To address this, we use entropy-based weighting, following observations from UniTE (Yao et al., 2025) that low-confidence tokens tend to introduce noise. To improve robustness, we only compute entropy over the top- k tokens in each model’s predicted distribution (Ma et al., 2025), filtering out uninformative tails.

Specifically, define $\text{Top}_k(p^{(\text{assist})})$ as the top- k highest scoring assist tokens. The uncertainty-based weight $w_{\text{ent}}^{(\text{assist})}$ is computed as the inverse entropy of the Top- k tokens, that is

$$w_{\text{ent}}^{(\text{assist})} = \frac{1}{H(p^{(\text{assist})})}, \text{ where } H(p) = - \sum_{v \in \text{Top}_k(p)} (p(v)/Z) \log(p(v)/Z)$$

where $Z = \sum_{v \in \text{Top}_k(p)} p(v)$. A similar procedure can be applied to the main model, denoted $w_{\text{ent}}^{(\text{main})}$. We then normalize all weights across the main model and all assist models so that they sum to one. The final ensembled distribution over the main model’s vocabulary is:

$$\tilde{p} = w^{(\text{main})} \cdot p^{(\text{main})} + \sum_{\kappa=1}^K w^{(\text{assist}\#\kappa)} \cdot T^{(\text{assist}\#\kappa)\top} p^{(\text{assist}\#\kappa)},$$

where $[w^{(\text{main})}, w^{(\text{assist}\#1)}, \dots, w^{(\text{assist}\#K)}] \in \Delta^{K+1}$ are the normalized uncertainty-based weights that control the contribution of the main model and assist models.

All token-translation matrices are precomputed and cached as highly sparse matrices. Each column (corresponding to a single assist-model token) has only one or a small number of

non-zero entries that point to its top-ranked aligned tokens in the main model’s vocabulary. This sparsity enables fast lookup and efficient memory usage.

4 Experiments

In this section, we design experiments to answer the following research questions:

- **RQ1:** Does our ensemble framework consistently improve the performance of base models across diverse tasks?
- **RQ2:** How do different components, such as uncertainty modeling and prefix/superstring-based token mappings, and alignment strategies contribute to ensemble effectiveness?
- **RQ3:** How efficient is our method in terms of preprocessing overhead, inference latency, and GPU memory compared to existing ensemble baselines?
- **RQ4:** How does performance scale with the number and capacity of involved models?

4.1 Experimental Setup

Models. We conduct all experiments using a diverse set of open-source chat and instruct-tuned models, including LLaMA-3.1-8B-Instruct (Grattafiori et al., 2024), InternLM-2.5-7B/20B-Chat (Team, 2023), GLM-4-9B-Chat (GLM et al., 2024), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023a), Qwen2.5-7B-Instruct (Yang et al., 2024), Yi-1.5-9B-Chat (Young et al., 2024), and Gemma-2-9B-IT (Team et al., 2024). These models reflect a range of widely adopted architectures spanning practical model sizes and are among the most recent publicly available versions at the time of the experiments.

Baselines. We compare against four representative test-time ensembling methods. **LLM-Blender** (Jiang et al., 2023b) uses a reward model (PairRanker) and a fusion model (GenFuser) to rerank and merge responses. Due to over-generation issues with GenFuser, we use only the reward-based selection. **GAC** (Yu et al., 2024) maps token probabilities into a unified space using a learned matrix and performs token-level aggregation. **CES** (Phan et al., 2025) ensembles probabilities at the sub-token (byte) level for fill-in-the-middle tasks. **DeepEn** (Huang et al., 2024) projects model outputs into a shared latent space based on overlapping vocabularies, using relative representation theory. **UniTe** (Yao et al., 2025) uses a top- k union strategy to align and filter tokens.

Experimental Setup. We evaluate nine benchmarks across four task categories: **comprehensive exams, reasoning, knowledge-intensive question answering, and summarization**. Dataset details can be found in Appendix A. We evaluate both two-model and multi-model ensembling across the aforementioned benchmarks. For the two-model setting, we select LLaMA-3.1-8B-Instruct and InternLM-2.5-7B-Chat as the main pair, chosen for their strong and comparable performance. For multi-model settings, we include Qwen2.5-7B-Instruct and additional models to assess scalability. More ablations on diverse model combinations are provided in Section C.2. Unless otherwise specified, we set $\alpha = 0.5$ and $k = 50$, and run all experiments on NVIDIA A100 80GB GPUs.

4.2 Main Results (RQ1)

As shown in Table 2, our method (ToT) achieves the best performance across all benchmarks, with an average improvement of +5.95 points over the main model (LLaMA-3.1-8B-Instruct). It delivers consistent gains across tasks (e.g., +4.79 on NQ, +7.01 on TriviaQA, +7.02 on ARC-C), and remains robust even when the main model underperforms, for example, surpassing both base models on ARC-C (85.74 vs. 78.72 and 85.13).

We also report a *roofline accuracy*, the percentage of examples answered correctly by at least one base model, as an empirical upper bound for training-free ensembles. ToT closely tracks this upper bound across tasks, especially on reasoning-oriented benchmarks (GSM8K and ARC-c), indicating that it effectively aggregates complementary reasoning signals

Table 2: Evaluation results across datasets for base models and LLM ensemble methods in two-model setting. Ensembling uses **LLaMA-3.1-8B-Instruct** (light blue) as the main model and **InternLM-2.5-7B-Chat** (light pink) as the assist model. Best results are in **bold**, and second-best are underlined.

Model	NQ	TriviaQA	ARC-c	MMLU	GSM8K	PIQA	Avg
<i>Base Models</i>							
Gemma-2-9B-IT	13.49	52.53	49.57	41.56	78.23	61.96	49.22
GLM-4-9B-Chat	24.27	58.94	88.06	64.68	76.41	82.58	65.82
InternLM-2.5-7B-Chat	26.62	63.23	85.13	70.57	83.93	87.47	69.49
LLaMA-3.1-8B-Instruct	27.51	65.57	78.72	67.38	82.87	82.86	67.82
Mistral-7B-Instruct	25.37	66.61	73.42	58.76	61.74	69.55	59.91
Qwen2.5-7B-Instruct	15.04	53.23	87.35	70.46	78.67	85.25	65.67
Yi-1.5-9B-Chat	14.54	51.74	80.64	66.27	64.69	80.64	59.75
<i>LLM Ensemble Methods</i>							
LLM-Blender	25.10	61.12	76.01	65.89	80.15	80.43	64.78
GAC	22.86	65.45	70.83	68.92	75.62	78.90	63.10
CES	30.54	66.20	81.80	68.10	81.10	87.72	69.24
DeepEn	28.63	66.32	75.49	68.23	81.33	79.24	66.21
Unite	29.11	67.45	83.23	69.56	84.43	86.63	70.07
ToT (Ours)	32.30	72.58	85.74	71.89	87.41	88.69	73.77
<i>Improve over Base LLM</i>	(+4.79)	(+7.01)	(+7.02)	(+4.51)	(+4.54)	(+5.83)	(+5.95)
Oracle (Roofline)	39.11	78.53	89.06	80.48	91.20	94.90	78.88

at inference. The remaining headroom is concentrated in knowledge-intensive settings, suggesting that integrating external knowledge or task-specific priors could further narrow the gap. Compared to prior methods, LLM-Blender relies on coarse-grained output ranking and shows limited improvement. Unite uses top- k token selection but lacks generation-awareness. Other methods like GAC and DeepEn suffer from poor token alignment across vocabularies. These results highlight the value of tokenizer-aware alignment and uncertainty modeling in test-time LLM ensembling.

4.3 Ablation Study (RQ2)

We perform ablation experiments on the NQ benchmark to assess the contribution of each module and the effectiveness of alternative alignment strategies (Table 3). Our two-model ToT ensemble improves the main model by +4.79 points (32.30), and adding a third model (Mistral-7B-Instruct) further boosts performance to 32.94, demonstrating scalability. Among core components, removing uncertainty weighting yields a minor drop (32.11), indicating stability benefits. In contrast, removing the prefix-based alignment matrix M^{pre} significantly reduces performance (30.75), confirming the importance of generation-aware alignment. Omitting the reverse mapping M^{sup} (31.86) or the clipping factor β (32.12) also leads to degradation, highlighting the utility of bidirectional alignment and confidence balancing. We also compare ToT with alternative alignment strategies, including semantic similarity (27.63), lexical matching (28.55), and edit-distance-based alignment (Wan et al., 2024a) (29.12). All underperform our method, underscoring the benefit of leveraging tokenizer priors over surface heuristics.

Table 3: Ablation study on NQ. We compare ToT with different variants and token mapping strategies.

Setting	NQ Acc.
<i>Baselines</i>	
InternLM-2.5-7B-Chat	26.62
LLaMA-3.1-8B-Instruct	27.51
ToT (2 models)	32.30
+Mistral-7B (3 models)	32.94
<i>Ablations</i>	
w/o Uncertainty	32.11
w/o M^{pre}	30.75
w/o M^{sup}	31.86
w/o β	32.12
<i>Alternative Mapping Strategies</i>	
with Semantic	27.63
with Lexical	28.55
with MinED (Wan et al., 2024a)	29.12

Hyperparameter studies for $\alpha - k$ (Appendix C.1) show stable performance across wide ranges, with fixed defaults near-optimal and consistently outperforming baselines. Results across diverse model combinations (Appendix C.2) demonstrate gains that persist when swapping the *main* and *assist* roles; the improvement magnitude correlates with the assist model’s strength. Token-alignment visualizations (Appendix C.3) reveal sparse, semantically grounded mappings and cases where the fused distribution corrects errors even

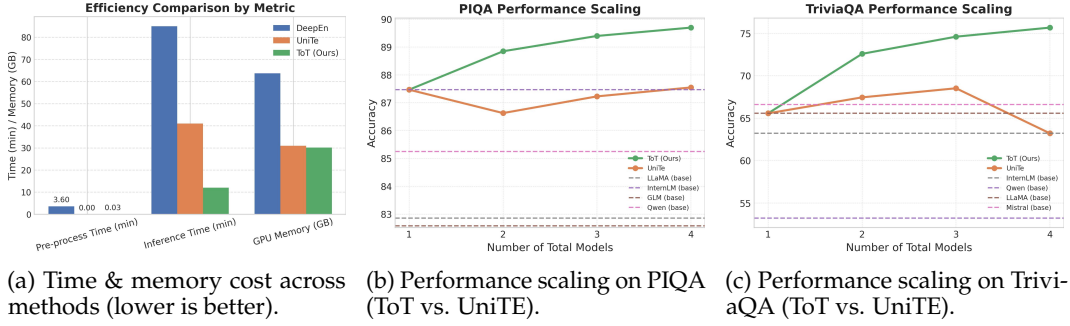


Figure 2: Efficiency and scaling overview. **Left:** wall-clock time and peak memory across ensembling methods. **Middle/Right:** scaling behavior of ToT vs. UniTE across base models.

when both base models’ predictions are wrong. Appendix C.4 presents empirical studies on tokenizer alignment properties, scalability across models and languages, and the effect of alternative uncertainty measures that further confirm our approach’s effectiveness.

4.4 Efficiency Analysis (RQ3)

Figure 2a compares the time and memory costs of different ensembling methods. ToT achieves clear efficiency advantages in both preprocessing and inference. Unlike DeepEn, which requires costly offline alignment, ToT relies only on lightweight decode-encode operations to construct a sparse mapping (0.03 min vs. 3.6 min). At inference, ToT performs a single sparse matrix multiplication without altering model architecture or decoding logic, yielding substantially lower time (12 min vs. 85 min) and memory use (30.2 GB vs. 63.7 GB). Compared to UniTE, ToT is also more efficient due to its sparse and fixed-tokenizer design. Overall, ToT delivers strong accuracy (Table 2) while maintaining practical speed and memory scalability.

4.5 Scalability Across Model Sizes and Quantities (RQ4)

To evaluate scalability, we consider two settings: (1) ensembling models of different scales, and (2) increasing the number of ensembled models.

As shown in Table 4, ToT achieves substantial gains when applied to large-scale models. By ensembling InternLM2.5-20B-Chat with LLaMA-3.1-8B-Instruct, ToT consistently outperforms both base models across all four tasks (e.g., +10.51 on HumanEval, +7.31 on NQ). This highlights the flexibility of our approach in handling model pairs with heterogeneous capacities and confirms its effectiveness in large-model regimes.

Table 4: Ensemble performance across model scales on four tasks.

Model	HumanEval	SAMSum	NQ	PIQA
LLaMA-3.1-8B-Instruct	73.17	31.70	27.51	82.86
InternLM2.5-20B-Chat	79.73	32.29	28.53	88.19
UniTE	78.43	32.45	31.65	88.96
ToT (Ours)	90.24	32.92	34.82	89.79

Furthermore, Figure 2b-2c illustrates performance scaling trends as more models are added to the ensemble. For both TriviaQA and PIQA, ToT demonstrates a clear upward trajectory, achieving steady improvements from 2 to 4 models. In contrast, UniTE’s performance plateaus or even degrades when additional models are introduced, likely due to its less robust token selection strategy and lack of generation preference modeling. This further confirms ToT’s robustness and generality when scaling across both model diversity and ensemble size.

5 Conclusion

We propose a lightweight and effective framework ToT for token-level ensembling of LLMs, addressing the core challenge of tokenizer mismatch. Our method aligns heterogeneous token spaces via a simple token translation based on tokenizer priors, enhanced by bidirectional alignment and uncertainty-aware fusion. Extensive experiments across tasks, model sizes, and ensemble settings show that ToT consistently outperforms existing baselines with high efficiency. Looking ahead, selecting complementary model combinations is a key challenge, and adaptive strategies for model pairing will be essential as the LLM landscape grows. Extending ToT to closed-source models that expose top-k log-probabilities is also a promising direction for expanding its applicability.

Acknowledgements

This work is supported by NSF (2316233). This work is also supported by National Science Foundation under Award No. IIS-2117902. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the government.

Ethics Statement. This work focuses on advancing algorithmic methods for test-time ensembling of large language models. No human subjects or sensitive personal data are involved. All datasets used in our experiments (e.g., MMLU, ARC-C, GSM8K, TriviaQA, NaturalQuestions, SAMSum, HumanEval, PIQA) are standard public benchmarks widely adopted in prior research. We have taken care to follow the COLM Code of Ethics by ensuring appropriate dataset usage, acknowledging limitations, and avoiding any harmful applications. The proposed method is purely methodological and does not pose direct risks related to privacy, security, or fairness.

References

- Aakriti Agrawal, Mucong Ding, Zora Che, Chenghao Deng, Anirudh Satheesh, John Langford, and Furong Huang. Ensemw2s: Can an ensemble of llms be leveraged to obtain a stronger llm? *arXiv preprint arXiv:2410.04571*, 2024.
- Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.
- 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.
- Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for llm selections. *arXiv preprint arXiv:2410.03834*, 2024.
- Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *arXiv preprint arXiv:1911.12237*, 2019.
- Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.

- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- Kevin Gu, Eva Tuecke, Dmitriy Katz, Raya Horesh, David Alvarez-Melis, and Mikhail Yurochkin. Chared: Character-wise ensemble decoding for large language models. *arXiv preprint arXiv:2407.11009*, 2024.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.
- Yichong Huang, Xiaocheng Feng, Baohang Li, Yang Xiang, Hui Wang, Ting Liu, and Bing Qin. Ensemble learning for heterogeneous large language models with deep parallel collaboration. *Advances in Neural Information Processing Systems*, 37:119838–119860, 2024.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L  lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. Mistral 7b. *CoRR*, abs/2310.06825, 2023a. doi: 10.48550/ARXIV.2310.06825. URL <https://doi.org/10.48550/arXiv.2310.06825>.
- Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14165–14178, 2023b.
- Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.
- Ehsan Kamalloo, Nouha Dziri, Charles Clarke, and Davood Rafiei. Evaluating open-domain question answering in the era of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5591–5606, 2023.
- Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*, 2018.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.
- Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models. *arXiv preprint arXiv:2311.08692*, 2023.
- Huan Ma, Jingdong Chen, Guangyu Wang, and Changqing Zhang. Estimating llm uncertainty with logits. *arXiv preprint arXiv:2502.00290*, 2025.
- OpenAI. Gpt-4 technical report. 2023. URL <https://api.semanticscholar.org/CorpusID:257532815>.

- Buu Phan, Brandon Amos, Itai Gat, Marton Havasi, Matthew J Muckley, and Karen Ullrich. Exact byte-level probabilities from tokenized language models for fim-tasks and model ensembles. In *The Thirteenth International Conference on Learning Representations*, 2025.
- Yangjun Ruan, Saurabh Singh, Warren Morningstar, Alexander A Alemi, Sergey Ioffe, Ian Fischer, and Joshua V Dillon. Weighted ensemble self-supervised learning. *arXiv preprint arXiv:2211.09981*, 2022.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.
- Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.
- InternLM Team. Internlm: A multilingual language model with progressively enhanced capabilities, 2023.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971, 2023. URL <https://api.semanticscholar.org/CorpusID:257219404>.
- Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. Knowledge fusion of large language models. In *ICLR*, 2024a.
- Fanqi Wan, Longguang Zhong, Ziyi Yang, Ruijun Chen, and Xiaojun Quan. Fusechat: Knowledge fusion of chat models. *arXiv preprint arXiv:2408.07990*, 2024b.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.
- Yangyifan Xu, Jianghao Chen, Junhong Wu, and Jiajun Zhang. Hit the sweet spot! span-level ensemble for large language models. *arXiv preprint arXiv:2409.18583*, 2024a.
- Yangyifan Xu, Jinliang Lu, and Jiajun Zhang. Bridging the gap between different vocabularies for llm ensemble. 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. 7133–7145, 2024b.
- Yangyifan Xu, Jianghao Chen, Junhong Wu, and Jiajun Zhang. Hit the sweet spot! span-level ensemble for large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 8314–8325, 2025.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.
- Yuxuan Yao, Han Wu, Sichun Luo, Xiongwei Han, Jie Liu, Zhijiang Guo, Linqi Song, et al. Determine-then-ensemble: Necessity of top-k union for large language model ensembling. In *The Thirteenth International Conference on Learning Representations*, 2025.
- Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Guoyin Wang, Heng Li, Jiangcheng Zhu, Jianqun Chen, et al. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*, 2024.

Yao-Ching Yu, Chun-Chih Kuo, Ziqi Ye, Yu-Cheng Chang, and Yueh-Se Li. Breaking the ceiling of the llm community by treating token generation as a classification for ensembling. *arXiv preprint arXiv:2406.12585*, 2024.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57, 2024.

Appendix

Contents

A Dataset Statistics	15
B Token Translation Algorithm	16
C Further Ablation Study and Visualization	16
C.1 Hyper-parameter Study	16
C.2 Ablation Study on Various Model Ensembling	16
C.3 Demonstration of Alignment and Ensembling	17
C.4 Empirical Analysis of Tokenizers and Performance	21
D Formal Analysis of Token Translation	22
D.1 Problem Setup	23
D.2 Tokenizer Assumptions	23
D.3 Causal Alignment	23
D.4 ToT Recovers the Exact Alignment	23
D.5 Causal Validity	24
E Use of Large Language Models	24
F Limitations	24

A Dataset Statistics

Algorithm 1 Token Translation (ToT)

Input: Tokenizers $\text{Tok}_{\text{assist}}, \text{Tok}_{\text{main}}$; Vocabularies $\mathcal{V}_{\text{assist}}, \mathcal{V}_{\text{main}}$; weight α

Initialize: $M^{(\text{pre})} \leftarrow \mathbf{0}_{|\mathcal{V}_{\text{assist}}| \times |\mathcal{V}_{\text{main}}|}$, $M^{(\text{sup})} \leftarrow \mathbf{0}_{|\mathcal{V}_{\text{assist}}| \times |\mathcal{V}_{\text{main}}|}$

for $i = 1, 2, \dots, |\mathcal{V}_{\text{assist}}|$ **do**

$v_i^{(\text{assist})} \leftarrow \text{Tok}_{\text{assist}}.\text{decode}(i)$

$[t_1, \dots, t_m] \leftarrow \text{Tok}_{\text{main}}.\text{encode}(v_i^{(\text{assist})})$

$M_{i, t_1}^{(\text{pre})} \leftarrow 1$

end for

for $j = 1, 2, \dots, |\mathcal{V}_{\text{main}}|$ **do**

$v_j^{(\text{main})} \leftarrow \text{Tok}_{\text{main}}.\text{decode}(j)$

$[t'_1, \dots, t'_n] \leftarrow \text{Tok}_{\text{assist}}.\text{encode}(v_j^{(\text{main})})$

$M_{t'_1, j}^{(\text{sup})} \leftarrow 1$

end for

$M^{(\text{assist} \rightarrow \text{main})} \leftarrow M^{(\text{pre})} + \alpha \cdot M^{(\text{sup})}$

Return: $M^{(\text{assist} \rightarrow \text{main})}$

We evaluate nine benchmark datasets across four representative task categories to assess language model performance under diverse reasoning and comprehension requirements.

(1) Comprehensive Exams: MMLU (5-shot, 5,000 test examples) (Hendrycks et al., 2020) is a multiple-choice benchmark that spans 57 diverse subjects across STEM, humanities, social sciences, and more. Each question contains four answer choices, and models are evaluated using in-context few-shot prompting and accuracy as the metric. ARC-Challenge (0-shot, 1,172 examples) (Clark et al., 2018) is a science reasoning benchmark targeting difficult grade-school level questions with multiple-choice formats. Models must directly select the correct answer without in-context examples.

(2) Reasoning: GSM8K (4-shot with chain-of-thought prompting (Wei et al., 2022), 1,319 examples) (Cobbe et al., 2021) consists of grade-school math word problems. The model is prompted with four examples that explicitly demonstrate step-by-step intermediate reasoning, and the final prediction is considered correct only if the numeric answer matches the gold label exactly. PIQA (0-shot, 1,838 examples) (Bisk et al., 2020) evaluates physical commonsense reasoning: models are given a naturalistic question involving physical interactions and must choose between two plausible outcomes. HumanEval (0-shot, 164 examples) (Chen et al., 2021) is a code generation benchmark where each input specifies a function signature and docstring in Python, and the model is expected to synthesize correct code. Performance is evaluated using pass@1, indicating the percentage of problems solved correctly in a single attempt.

(3) Knowledge-Intensive QA: TriviaQA (5-shot, 11,313 examples) (Joshi et al., 2017) contains open-domain factoid questions curated from trivia websites, with Wikipedia-based ground truth answers. Each input includes five QA exemplars followed by a new question; accuracy is measured by exact match. NaturalQuestions (NQ) (5-shot, 3,610 examples) (Kwiatkowski et al., 2019) consists of real anonymized Google search queries, paired with short answers derived from Wikipedia articles. We follow prior work in using short-form extractive answer strings and compute exact match as the evaluation metric.

(4) Summarization: SAMSum (0-shot, 818 examples) (Gliwa et al., 2019) is a single-document summarization task where the input is a multi-turn dialogue between fictitious participants, and the model must produce a concise and coherent summary. No demonstrations are provided, and performance is measured using ROUGE-L.

Following prior work (Yao et al., 2025; Huang et al., 2024), we construct the prompt by providing the question followed by the answer format, optionally including in-context examples and chain-of-thought prompting using the phrase “let’s think step-by-step” (Wei et al., 2022).

B Token Translation Algorithm

Algorithm 1 summarizes the proposed token translation procedure. It first constructs a forward mapping by decoding each assist token and re-tokenizing it with the main tokenizer, assigning the translation to the first resulting main token to preserve the main model’s generative prefix prior. It then builds a reverse mapping by decoding each main token and re-tokenizing it with the assist tokenizer, capturing complementary superstring relations. The final alignment matrix is obtained by combining these two mappings with weight α , yielding a lightweight yet effective bidirectional translation scheme.

C Further Ablation Study and Visualization

In this section, we first perform a comprehensive hyperparameter study to demonstrate the robustness of our method under various settings. We then conduct an ablation study using different pairs of large language models (LLMs) in the ensemble to further verify its robustness across diverse model combinations. Finally, we present representative output examples and token alignment visualizations to provide qualitative insights into the behavior and effectiveness of our approach.

C.1 Hyper-parameter Study

Figures 3, 4, and 5 demonstrate robustness across the key hyperparameters α , β , and k . Here, α controls the blend between prefix and superstring mappings when aligning token sequences; β controls the *strength of the main model* by enforcing a minimum contribution—implemented as clipping the normalized weight so that $w^{(\text{main})} \leftarrow \max(\beta, w^{(\text{main})})$ and renormalizing the remaining mass across assist models; and k sets how many (top) tokens are used to compute model uncertainty. Despite their different roles, all three are easy to configure and yield stable performance over wide ranges. Across settings, our method consistently outperforms the base models, making it practical without tuning; thus we fix these values uniformly in all experiments.

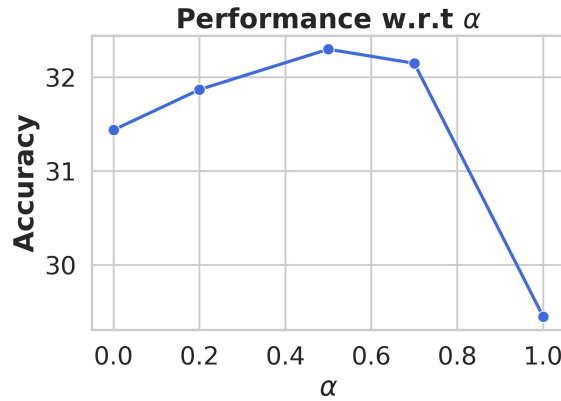


Figure 3: Performance with respect to α , representing the ratio between prefix and superstring mappings on the NQ dataset with InternLM and LLaMA models.

C.2 Ablation Study on Various Model Ensembling

Figure 6 provides a comprehensive view of the performance gains achieved when ensembling different main and assist model pairs. Two key observations emerge.

First, our method is highly robust, as it consistently delivers substantial performance improvements across all combinations of main and assist models. No matter which model

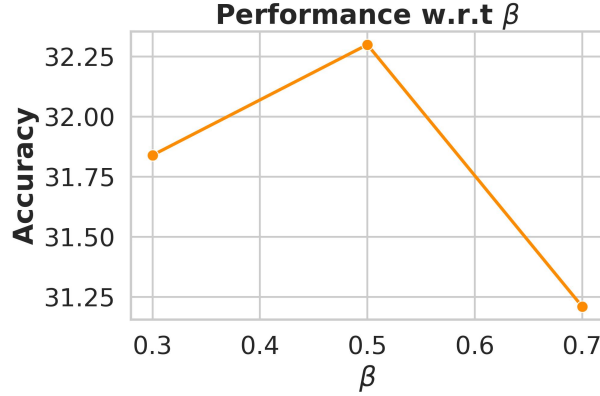


Figure 4: Performance with respect to β , denoting the minimum contribution of the main model to the final prediction, evaluated on the NQ dataset with InternLM and LLaMA.

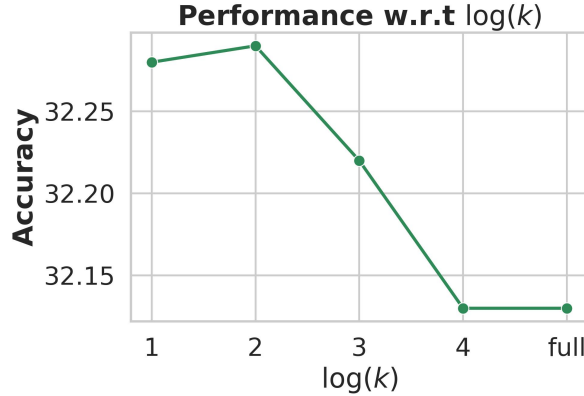


Figure 5: Performance with respect to $\log k$, where k is the number of tokens used to compute uncertainty, evaluated on the NQ dataset using InternLM and LLaMA.

is used as the main model, GLM, InternLM, LLaMA, or Mistral, the introduction of an assist model yields a notable gain over the base performance. This demonstrates that our ensembling strategy generalizes well and is not overly sensitive to the choice of models involved.

Second, the magnitude of the gain tends to correlate with the strength of the assist model. Specifically, using stronger models like InternLM and LLaMA as assist models leads to the largest improvements, with performance gains exceeding 5 points in some cases. For example, pairing InternLM as the assist model with GLM as the main model yields the highest observed gain. This suggests that stronger assist models provide more informative or complementary signals during ensembling, thus boosting the overall prediction quality.

Together, these results validate the versatility and effectiveness of our method, showing it works well in diverse model settings and particularly excels when high-capacity assist models are available.

C.3 Demonstration of Alignment and Ensembling

Table 5 compares three token mapping methods: string similarity, embedding similarity, and our proposed Token Translation. String similarity is based on edit distance at the string level,

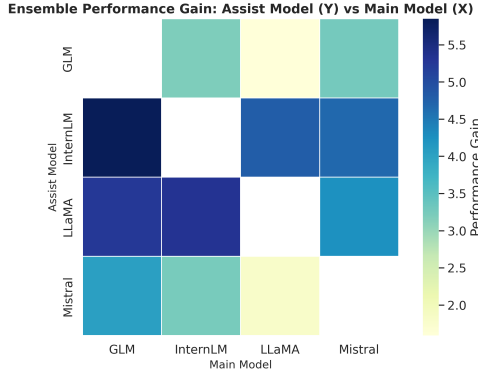


Figure 6: Performance gain with different main and assist models.

while embedding similarity measures cosine similarity between token embeddings. In contrast, Token Translation (using prefix-based mapping) leverages both the model’s generation preferences and tokenizer priors to identify mappings that align more closely with actual generation behavior. As evidenced by the performance results in Table 3, both string- and embedding-based methods often produce suboptimal or irrelevant mappings—for example, mapping “Papa” to “Luna” or “Administrator” to a Chinese token—because they overlook how tokens function during decoding. In contrast, Token Translation consistently produces compact, semantically coherent segments such as “Admin” and “P” that better match the tokenizer’s segmentation and generation dynamics. This demonstrates the effectiveness of our approach in generating mappings that are not only linguistically and structurally aligned, but also more suitable for downstream ensemble generation.

Table 5: Comparison of Token Mapping Methods

Method	Source Token	Mapped Token
String sim	Administrator	Administr
Embedding sim	Administrator	Administrator in Chinese
Token Translation	Administrator	Admin
String sim	_CAT	_CA
Embedding sim	_CAT	_VAR
Token Translation	_CAT	_C
String sim	Papa	Pap
Embedding sim	Papa	Luna
Token Translation	Papa	P

Next, we present output examples of our ensemble method in Tables 6 and 7, which not only correct errors made by the base models but also outperform baseline methods on the GSM8K dataset. Correct answers are highlighted in green, while incorrect ones are marked in red. These cases clearly demonstrate the effectiveness of our method in enhancing reasoning accuracy and robustness by leveraging complementary strengths of different models.

Table 6: Case study of ToT and other ensembling methods on GSM8K.

Question	Blake and Kelly are having a contest to see who can run the most in 15 minutes. They decide to do it on a football field that is 100 yards long. Blake runs back and forth 15 times. Kelly runs back and forth once, and then decides that she doesn't want to run next to Blake, so she starts to run to the 40-yard line and back. She does this 34 times. How much farther does the winner run than the loser?
LLaMA	Blake runs 15 times back and forth on a 100-yard field, so he runs $15 \times 2 \times 100 = 3000$ yards. Kelly runs 34 times to the 40-yard line and back, so she runs $34 \times 2 \times 40 = 2720$ yards. Blake runs 3000 yards and Kelly runs 2720 yards, so Blake runs $3000 - 2720 = 280$ yards more than Kelly. The answer is 280
InternLM	First, let's calculate how far Blake runs. A football field is 100 yards long, and he runs back and forth 15 times. So, Blake runs $100 \text{ yards} \times 15 = 1500$ yards. Next, let's calculate how far Kelly runs. She runs back and forth once, which is $100 \text{ yards} \times 2 = 200$ yards. Then, she starts running to the 40-yard line and back, which is $40 \text{ yards} \times 2 = 80$ yards. She does this 34 times, so she runs an additional $80 \text{ yards} \times 34 = 2720$ yards. In total, Kelly runs $200 \text{ yards} + 2720 \text{ yards} = 2920$ yards. To find out how much farther the winner runs than the loser, we subtract the distance that Blake ran from the distance that Kelly ran: $2920 \text{ yards} - 1500 \text{ yards} = 1420$ yards. The answer is 1420
UniTE	Blake and Kelly are having a running contest on a 100-yard football field. Blake runs back and forth 15 times, and since each round trip is 100 yards, he runs a total of $15 \times 100 = 1500$ yards. Kelly first runs one full trip across the field and back, which is $2 \times 100 = 200$ yards. After that, she runs to the 40-yard line and back 34 times. Since the 40-yard line is not the full length, we compute her additional distance as $40 \times 34 = 1360$ yards. Adding this to her initial 200 yards, Kelly runs a total of 1560 yards. Comparing the two, Kelly runs $1560 - 1500 = 60$ yards more than Blake. The answer is 60
ToT (Ours)	Blake runs back and forth 15 times on a 100-yard field, so he runs $15 \times 2 \times 100 = 3000$ yards. Kelly runs back and forth once, which is $2 \times 100 = 200$ yards. Then she runs to the 40-yard line and back, which is $2 \times 40 = 80$ yards. She does this 34 times, so she runs $34 \times 80 = 2720$ yards. The total distance that Kelly runs is $200 + 2720 = 2920$ yards. The difference between the distances that Blake and Kelly run is $3000 - 2920 = 80$ yards. The answer is 80

Table 7: Case study of ToT and other ensembling methods on GSM8K.

Question	John buys milk for 2 dollars, eggs for 3 dollars, light bulbs for 3 dollars, cups for 3 dollars, and roach traps for 4 dollars. If there is a 10% tax on all nonfood items how much did everything cost?
LLaMA	John buys milk for 2 dollars, eggs for 3 dollars, and roach traps for 4 dollars, so the total cost of the food items is $2+3+4=9$ dollars. The nonfood items are the light bulbs, cups, and roach traps. The total cost of the nonfood items is $3+3+4=10$ dollars. There is a 10% tax on the nonfood items, so the tax is 10% of 10 dollars, which is $0.1*10=1$ dollar. The total cost of the nonfood items with tax is $10+1=11$ dollars. The total cost of everything is the cost of the food items plus the cost of the nonfood items with tax, so the total cost is $9+11=$ 20
InternLM	First, let's calculate the cost of the non-food items before tax. The non-food items are the light bulbs, cups, and roach traps. The cost of the light bulbs is \$3, the cups are also \$3, and the roach traps are \$4. So the total cost of the non-food items is $\$3 + \$3 + \$4 = \10 . The answer is 10
UniTE	John buys milk for 2 dollars, eggs for 3 dollars, and roach traps for 4 dollars, so the total for food items is $2 + 3 + 4 = 9$ dollars. The nonfood items include light bulbs, cups, and roach traps, totaling $3 + 3 + 4 = 10$ dollars. A 10% tax applies to nonfood items, so the tax is 10% of 10 dollars, which is $0.1 \times 10 = 1$ dollar. Adding the tax, the nonfood total becomes $10 + 1 = 11$ dollars. Combining the food and taxed nonfood totals, the final amount is $9 + 11 =$ 20
ToT (Ours)	First, let's calculate the total cost of the non-food items. The non-food items are the light bulbs, cups, and roach traps. The cost of the light bulbs is \$3, the cups are \$3, and the roach traps are \$4. So the total cost of the non-food items is $\$3 + \$3 + \$4 = \10 . Next, we need to calculate the tax on the non-food items. The tax rate is 10%, so we need to find 10% of \$10. To do this, we multiply \$10 by 0.10 (which is the decimal equivalent of 10%). So the tax on the non-food items is $\$10 * 0.10 = \1 . Now, we can add the tax to the cost of the non-food items to find the total cost of the non-food items. The total cost of the non-food items is $\$10 + \$1 = \$11$. Finally, we need to add the cost of the milk and eggs to the total cost of the non-food items to find the total cost of everything. The cost of the milk is \$2 and the cost of the eggs is \$3. So the total cost of everything is $\$2 + \$3 + \$11 =$ 16

C.4 Empirical Analysis of Tokenizers and Performance

To complement the formal analysis in Appendix D, we present an empirical study of tokenizer behavior and its effect on cross-model alignment. This analysis examines how heterogeneous vocabularies interact in practice and how these structural properties influence ensemble performance. We begin by measuring token alignment coverage across representative LLM families.

C.4.1 Token Alignment Coverage Across Models

We begin by quantifying token-level alignment coverage across five representative LLM families. Table 8 reports the percentage of tokens in one vocabulary that find direct alignment matches in another. Coverage varies substantially across model pairs (e.g., 85.43% for LLaMA3→Qwen2.5 vs. 36.30% for GLM4→Mistral), revealing the strong tokenizer-induced structural differences that motivate a deterministic translation mechanism such as ToT.

Table 8: Cross-token alignment coverage among representative LLM tokenizers.

Model Pair	internlm2 (92k)	Qwen2.5 (152k)	LLaMA3 (128k)	GLM4 (151k)	Mistral (32k)
internlm2 (92k)	–	61.51%	62.70%	54.75%	79.65%
Qwen2.5 (152k)	61.51%	–	85.43%	33.05%	75.57%
LLaMA3 (128k)	62.70%	85.43%	–	39.15%	75.79%
GLM4 (151k)	54.75%	33.05%	39.15%	–	36.30%
Mistral (32k)	79.65%	75.57%	75.79%	36.30%	–

C.4.2 Forward/Backward Mapping Precision

Next, we examine how many non-shared tokens can be mapped via ToT. Table 9 reports results when mapping several assist models into the internlm2 vocabulary. Because vocabularies differ greatly in size and segmentation granularity, forward (assist→main) and backward (main→assist) mapping coverage can be highly asymmetric—for example, Qwen2.5→internlm2 achieves 91.35% forward coverage but only 23.09% backward coverage. Smaller vocabularies (e.g., Mistral’s 32k) also show the opposite pattern. These results illustrate why alignment would be better to consider tokenizer directionality rather than assume symmetric similarity.

Table 9: Forward (assist→main) and backward (main→assist) mapping coverage into internlm2.

Model → internlm2	Direction	Mapped	Non-Shared	Coverage
Qwen2.5 (152k)	Forward	32,540	35,616	0.9135
	Backward	21,969	95,136	0.2309
LLaMA3 (128k)	Forward	17,004	34,521	0.4927
	Backward	6,283	70,233	0.0894
GLM4 (151k)	Forward	37,705	41,876	0.9006
	Backward	32,371	100,884	0.3210
Mistral (32k)	Forward	1,061	66,443	0.0159
	Backward	5,719	6,667	0.8576

C.4.3 Semantic Coherence Analysis

To evaluate whether ToT’s alignments are semantically meaningful, Table 10 reports the average BGE embedding (Xiao et al., 2023) similarity between paired tokens. Scores are consistently high (0.70–0.97), confirming semantic coherence while remaining below the theoretical maximum, indicating that alignment is still governed by tokenizer, induced structural decomposition rather than semantic nearest neighbors.

Table 10: Semantic similarity of aligned tokens using BGE embeddings.

Model $\rightarrow$ internlm2	Direction	Avg Sim	Max Avg Sim
Qwen2.5 (152k)	Forward	0.7778	0.8347
	Backward	0.9407	0.9904
LLaMA3 (128k)	Forward	0.6999	0.7969
	Backward	0.8863	0.9880
GLM4 (151k)	Forward	0.7818	0.8398
	Backward	0.9669	0.9959
Mistral (32k)	Forward	0.7142	0.7679
	Backward	0.8411	0.9508

C.4.4 Scaling to Larger Backbones

To assess scalability, we evaluate ToT on Mixtral-8 $\times$ 7B and Qwen2.5-32B-Instruct. Table 11 shows that ToT consistently improves over both individual models and the UniTe baseline, demonstrating that the alignment benefits persist even for strong dense and MoE backbones.

Table 11: Scalability evaluation on larger backbones.

Model	SAMSum	NQ	Avg
Mixtral-8 $\times$ 7B	32.8	30.9	31.9
Qwen2.5-32B-Instruct	34.1	33.7	33.9
UniTe (baseline)	34.0	34.5	34.3
ToT (Ours)	35.3	36.9	36.1

C.4.5 Multilingual Evaluation: FLORES En $\rightarrow$ De and En $\rightarrow$ Zh

We further test robustness on FLORES En $\rightarrow$ De and En $\rightarrow$ Zh translation. Table 12 shows that ToT improves over both single models and UniTe, particularly in En $\rightarrow$ Zh where tokenization differences are largest.

Table 12: Multilingual evaluation on FLORES subsets.

Dataset	Model	BLEU ($\uparrow$)
En $\rightarrow$ De	LLaMA	29.87
	InternLM	26.74
	UniTe	33.17
	Ours	36.13
En $\rightarrow$ Chinese	LLaMA	27.45
	InternLM	32.10
	UniTe	34.82
	Ours	37.56

C.4.6 Uncertainty Measure Comparison

Finally, we compare several uncertainty metrics within the same routing framework. As shown in Table 13, top- k entropy achieves the strongest average performance. This aligns with the observation that long-tail logits introduce noise, while the top- k region contains the meaningful decision mass for autoregressive decoding.

D Formal Analysis of Token Translation

In this section, we provide a rigorous justification for our Token Translation (ToT) framework. We show that ToT is not a heuristic approximation, but a deterministic implementation of *Exact Causal Alignment* between heterogeneous token spaces. Under standard tokenizer

Table 13: Comparison of uncertainty measures for routing.

Uncertainty Measure	Description	Avg Score ($\uparrow$)
Entropy	Full-distribution entropy	32.13
Variance	Variance of the distribution	31.10
Calibration	$1 - \max(p)$ confidence proxy	31.56
Disagreement	$\text{KL}(\text{main} \parallel \text{assist})$	31.28
Top-k Entropy (Ours)	Entropy over normalized top- k mass	32.30

assumptions, ToT recovers the ground-truth conditional probability required for cross-model alignment.

D.1 Problem Setup

Let $\mathcal{V}_{\text{assist}}$ and $\mathcal{V}_{\text{main}}$ denote the vocabularies of the assist and main models, respectively. We posit a shared, continuous text space $\mathcal{S}$ representing all possible raw text strings. The alignment objective is to compute

$$P(v^{(\text{main})} \mid v^{(\text{assist})}), \quad v^{(\text{assist})} \in \mathcal{V}_{\text{assist}}, v^{(\text{main})} \in \mathcal{V}_{\text{main}}.$$

The true conditional probability is

$$P(v^{(\text{main})} \mid v^{(\text{assist})}) = \sum_{s \in \mathcal{S}} \underbrace{P(v^{(\text{main})} \mid s)}_{\text{Main Prior}} \cdot \underbrace{P(s \mid v^{(\text{assist})})}_{\text{Assist Decoding}}. \quad (1)$$

D.2 Tokenizer Assumptions

Modern BPE and Unigram tokenizers satisfy the following assumptions.

Assumption 1 (Deterministic Decoding). Each assist token $v^{(\text{assist})}$ deterministically decodes to a unique raw string:

$$P(s \mid v^{(\text{assist})}) = \mathbb{1}[s = \text{Tok}_{\text{assist}}.\text{decode}(v^{(\text{assist})})].$$

Assumption 2 (Canonical Encoding Prior). For any string s , the main tokenizer produces a unique canonical sequence $\text{Tok}_{\text{main}}.\text{encode}(s) = [t_1, t_2, \dots]$. Due to autoregressive causality,

$$P(v^{(\text{main})} \mid s) = \mathbb{1}[v^{(\text{main})} = \text{First}(\text{Tok}_{\text{main}}.\text{encode}(s))].$$

D.3 Causal Alignment

Under Assumptions 1–2, the intractable sum of $P_{\text{true}}(v^{(\text{main})} \mid v^{(\text{assist})})$ in Eq. 1 collapses to the following:

$$\mathbb{1}[v^{(\text{main})} = \text{First}(\text{Tok}_{\text{main}}.\text{encode}(\text{Tok}_{\text{assist}}.\text{decode}(v^{(\text{assist})})))].$$

This is the exact causal mapping from assist-token space to main-token space.

D.4 ToT Recovers the Exact Alignment

ToT constructs a forward translation matrix $M^{(\text{pre})}$:

$$\hat{P}_{\text{ToT}}(v^{(\text{main})} \mid v^{(\text{assist})}) \triangleq M_{v^{(\text{assist})}, v^{(\text{main})}}^{(\text{pre})},$$

where

$$M_{i,j}^{(\text{pre})} = \mathbb{1}[j = \text{First}(\text{Tok}_{\text{main}}.\text{encode}(\text{Tok}_{\text{assist}}.\text{decode}(i)))].$$

Thus,

$$\mathbb{E}[\hat{P}_{\text{ToT}}] = P_{\text{true}},$$

showing that ToT exactly recovers the deterministic causal alignment induced by the tokenizers.

D.5 Causal Validity

Consider an assist token that expands to a multi-token main sequence $[t_1, t_2, \dots, t_n]$. Define:

- E_{seq} : the event that the main model generates the full sequence;
- E_{t_1} : the event that it generates t_1 .

Since

$$E_{\text{seq}} \subseteq E_{t_1} \implies P(E_{\text{seq}}) \leq P(E_{t_1}),$$

the first token t_1 is a necessary causal precursor for generating the entire sequence. If the assist model assigns probability p to $v^{(\text{assist})}$, then the main model must assign at least p to t_1 . Therefore,

$$\hat{P}(t_1) \leftarrow p$$

preserves probability mass and yields a deterministic causal mapping for next-token prediction.

As shown in Table 3, this prefix-only mapping already surpasses strong baselines. Moreover, the backward translation matrix $M^{(\text{sup})}$ restores the residual probability mass spread across longer sequences, providing improved robustness through bidirectional alignment.

E Use of Large Language Models

During the preparation of this paper, we made limited use of large language models (LLMs), specifically ChatGPT, as an auxiliary writing tool. The LLM was used exclusively for stylistic refinement, including improvements to fluency, grammar, and readability of text initially drafted by the authors. All scientific content, including problem formulation, methodology, experiments, analyses, and overall narrative, was entirely conceived and validated by the authors. Thus, the role of LLMs was restricted to text polishing and does not constitute authorship.

F Limitations

Our approach assumes access to model tokenizers and output probabilities, which may not be available for all proprietary APIs. Additionally, selecting optimal model combinations for ensembling remains an open challenge.