TokenTiming: A Dynamic Alignment Method for Universal Speculative Decoding Model Pairs

Sibo Xiao[♠], **Jinyuan Fu**[♠], **Zhongle Xie**[♠✉], **Lidan Shou**[♠♣],
[♠]Zhejiang University
[♣]Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security
{xiaosibo_email, 3220100587, xiezl, should}@zju.edu.cn

Abstract

Accelerating the inference of large language models (LLMs) has been a critical challenge in generative AI. Speculative decoding (SD) substantially improves LLM inference efficiency. However, its utility is limited by a fundamental constraint: the draft and target models must share the same vocabulary, thus limiting the herd of available draft models and often necessitating the training of a new model from scratch. Inspired by Dynamic Time Warping (DTW), a classic algorithm for aligning time series, we propose the algorithm TokenTiming for universal speculative decoding. It operates by re-encoding the draft token sequence to get a new target token sequence, and then uses DTW to build a mapping to transfer the probability distributions for speculative sampling. Benefiting from this, our method accommodates mismatched vocabularies and works with any off-the-shelf models without re-training and modification. We conduct comprehensive experiments on various tasks, demonstrating 1.57× speedup. This work enables a universal approach for draft model selection, making SD a more versatile and practical tool for LLM acceleration. The code is available at the link.

1 Introduction

Speculative decoding (SD) accelerates LLM inference using a small draft model to propose tokens that are then verified by the larger target model (Leviathan et al., 2023; Chen et al., 2023). The effectiveness of SD depends on a draft model that is both fast and accurate in approximating the target distribution (Timor et al., 2025b; Chen et al., 2024). However, a fundamental assumption in current verification methods—the requirement of a shared vocabulary between the draft model and target model— prevents the widespread adoption of SD (Miao et al., 2024; Sun et al., 2024). This single constraint leads to two significant practical barriers:

Limited Selection of Draft Models. SD requires that the target and draft models share the same vocabulary. Many target models are deployed independently, possessing unique vocabularies that fundamentally preclude the choice of draft models. On the other hand, even within the same model family (e.g., GPT-OSS-120B/20B) (OpenAI, 2025), the smallest variants often remain too large to provide a substantial draft acceleration. An effective draft model must possess a parameter scale significantly smaller than the target model to ensure high inference efficiency. This dual, stringent constraint on both scale and vocabulary compatibility makes finding the optimal draft model a major practical hurdle.

Costly and Inflexible Training. For a selected target model, obtaining a draft model with complete word alignment usually requires starting from the pre/post-training stage, e.g., Medusa (Cai et al., 2024), EAGLE (Li et al., 2024a). Furthermore, if switched to a new target model, the previously trained model will no longer align. This is extremely inflexible in the current era where there are numerous model types and model iterations occur rapidly.

To address this, several alignment algorithms for universal speculative decoding (Timor et al., 2025a) have been proposed. These methods resolve the heterogeneity of model vocabularies by operating at the linguistic level. SLEM (String-level Exact Match) cannot perform probabilistic sampling, while the performance of TLI (Token-level Intersection) is constrained by the size of the vocabulary intersection between the draft and target models. In conclusion, these methods have only partially alleviated the problem, but they cannot fully meet all the requirements for implementing lossless speculative decoding.

To overcome these challenges, we present **TokenTiming**, a novel universal speculative decoding

✉Corresponding Author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 42798–42812

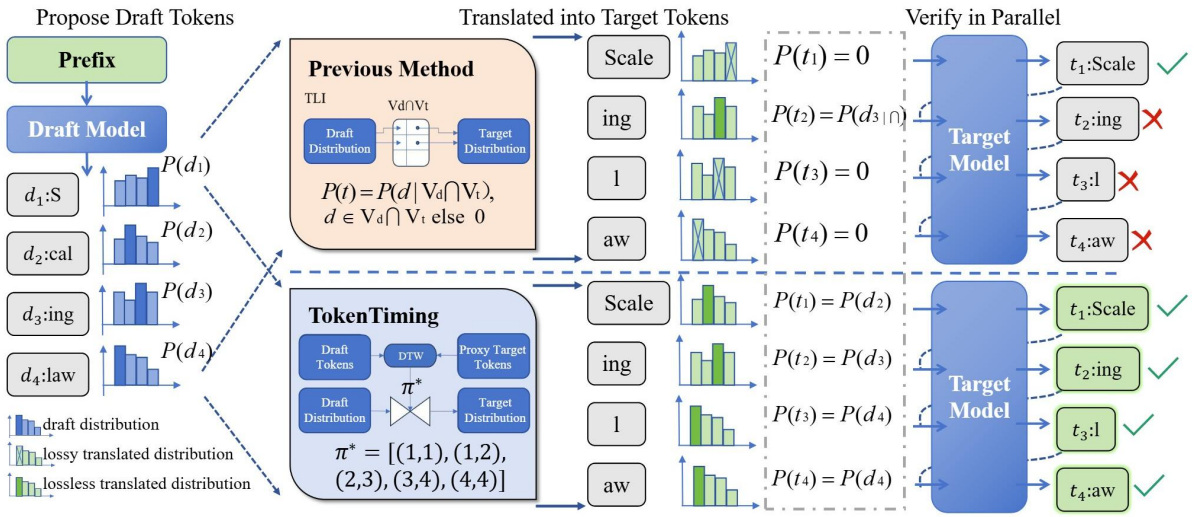

Figure 1: Comparison of core points of TokenTiming with previous work. Previous methods (e.g., TLI) carry the transfer of probability distribution in the vocabulary space of $\mathcal{V}_d \cap \mathcal{V}_t$. If the draft tokens fall outside the intersection, they will be accepted unconditionally, which violates the integrity of losslessness. TokenTiming constructs intact token mapping via DTW, ensuring lossless speculative sampling.

framework that enables lossless acceleration across heterogeneous vocabularies. At the core of TokenTiming is **Dynamic Token Warping (DTW)**, a lightweight alignment mechanism inspired by Dynamic Time Warping (Sakoe and Chiba, 1978) from time series analysis. Given a sequence of draft tokens, TokenTiming first converts it into a string and then re-tokenizes it using the target tokenizer to obtain a proxy target token sequence. DTW is then applied to construct a many-to-many alignment between the draft and proxy target token sequence, enabling accurate transfer of probability distributions from the draft vocabulary to the target vocabulary. This alignment is performed *on-the-fly* during each decoding step, without requiring any re-training or model modification.

Our Contributions:

- ***Universal Compatibility Without Shared Vocabularies:*** TokenTiming allows any off-the-shelf draft model to be plugged in without a strict vocabulary match.
- ***Strong Empirical Performance Across Tasks:*** On summarization, translation, code, and math, TokenTiming attains up to 1.57× speedup over autoregressive baselines and surpasses universal SD rivals.
- ***Approaching Homogeneous-Vocabulary SD SOTA Performance:*** On 7B/33B Models, TokenTiming yields 2.27× speedup, closing in on Medusa and EAGLE-1/2 while retaining model flexibility.

2 Related Works

SD with Homogeneous Vocabularies Speculative Decoding (SD) (Leviathan et al., 2023; Chen et al., 2023) reduces inference latency by using a fast draft model to propose tokens that are then verified in parallel by a larger target model. There are different types of speculative decoding approaches. Draft-head methods like Medusa (Cai et al., 2024), Hydra (Ankner et al., 2024), and EAGLE (Li et al., 2024b,a, 2025) integrate auxiliary heads into the target model to propose sequences. In contrast, Jacobi-based approaches such as Lookahead Decoding (Fu et al., 2024) and CLLM (Kou et al., 2024) enable parallel n-gram generation without draft models. System-level efforts (Miao et al., 2024; Liu et al., 2024) further optimize SD's runtime efficiency in serving systems.

SD with Pruned Vocabularies Zhao et al. (2025) proposed a vocabulary pruning strategy to enhance the efficiency of speculative decoding draft models. The core rationale is the pronounced long-tailed structure of token frequency distributions. An empirical analysis of Llama-3-8B on the SlimPajama dataset confirms this, showing that a vast majority of the vocabulary (75%) accounts for a small fraction (less than 5%) of token occurrences (Grattafiori et al., 2024; Soboleva et al., 2023). The proposed method operates by calculating token frequencies on a dataset S and constructing a reduced vocabulary comprising only the most frequent to-

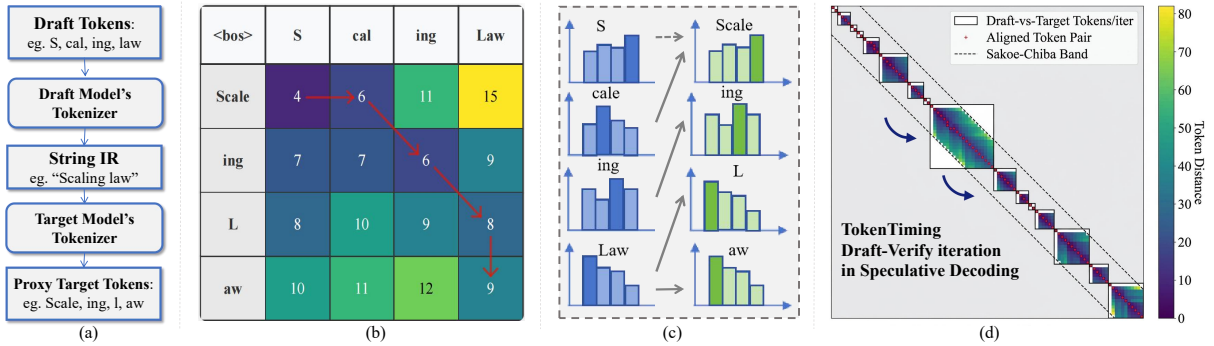

Figure 2: **Phase illustrations of TokenTiming. (a)** illustrates the re-tokenization of Draft Tokens into Proxy Target Tokens, which are used to construct the mapping in the DTW. **(b)** DTW calculation process (token distance matrix) and aligned token mapping $\pi^* = [(S, Scale), (cal, Scale), (ing, ing), (Law, L), (Law, aw)]$. The calculation rules for this mapping are presented in Alg. 1. **(c)** Probability distribution of draft tokens is transferred based on the mapping π^*. **(d)** Dynamic alignment in every Draft-Verify iteration of Speculative Decoding.

kens.

SD with Heterogeneous Vocabularies Token-Level Intersection (TLI) for heterogeneous draft models has been recently introduced by Timor et al. (2025a), which normalizes the intersected distribution by zeroing out all out-of-vocabulary mass. Alternatively, Redistributing draft model Kernels (RDK) (Timor et al., 2025c) uses a row-stochastic matrix M to convert the draft distribution d to a new distribution t' via the operation $t' = M^T d$. The resulting distribution t' and the target distribution t are then used for native speculative sampling.

3 Preliminaries

Speculative Decoding (SD) Let $\mathcal{V}_d$, $\mathcal{V}_t$ denote the vocabularies of draft model M_d and target model M_t respectively. SD accelerates autoregressive generation by letting M_d propose K tokens $\mathbf{D} = (d_1, \ldots, d_K)$ in one shot; M_t then verifies them in a *single* forward pass and rolls back at the first rejection. The expected speed-up is $\mathbb{E}[\gamma + 1]$ where $\gamma \in [0, K]$ is the number of accepted tokens. Crucially, standard verification assumes $\mathcal{V}_d = \mathcal{V}_t$ so that each draft token d_i can be directly verified by M_t.

Standard Speculative Sampling Speculative sampling enables *lossless* acceleration by verifying multiple tokens in parallel. Let $p(t)$ denote the probability assigned by the draft model to token t, and $q(t)$ the corresponding probability under the target model. A proposed token t is accepted with probability $\min\left(1, \frac{q(t)}{p(t)}\right)$, otherwise it is rejected and a new token is sampled from the adjusted target distribution. This acceptance rule guarantees that the final output distribution remains identical to the target model's distribution.

Vocabulary Mismatch When $\mathcal{V}_d \neq \mathcal{V}_t$[1], the draft token sequence $\mathbf{D}$ cannot be directly interpreted by M_t: the same surface form may be tokenised differently (e.g. **"Scaling"** $\rightarrow$ one token in $\mathcal{V}_t$ but **"Scal"+"ing"** in $\mathcal{V}_d$), and normalisation rules make re-encoding non-invertible, so the proxy target token sequence $\mathbf{t}$ can diverge sharply from $\mathbf{D}$, collapsing the accept rate.

4 Method

To address the fundamental challenges of vocabulary mismatch in universal speculative decoding, we propose **TokenTiming**, a novel algorithm built upon our core integration of **Dynamic Token Warping (DTW)**. The complete algorithmic procedure is formally presented in Fig. 2.

4.1 Core Component

Dynamic Token Warping (DTW) algorithm serves as the core alignment component in our framework, addressing the fundamental challenge of vocabulary mismatch between heterogeneous tokenizers. As formally presented in Alg. 1, DTW establishes an optimal many-to-many mapping between draft token sequence $D = (d_1, \ldots, d_k)$ and proxy target token sequence $T = (t_1, \ldots, t_m)$ through dynamic programming.

The algorithm operates by constructing a cumulative cost matrix $C \in \mathbb{R}^{k \times m}$, where each entry $C_{i,j}$ represents the minimum cumulative distance to align the first i draft tokens with the first j target

[1]Tab. 1 demonstrates detailed statistics of Vocabulary Mismatch by intersection ratio.

Algorithm 1 Dynamic Token Warping (DTW)

Require: Token sequences $\mathbf{X} = (x_1, \ldots, x_m)$, $\mathbf{Y} = (y_1, \ldots, y_n)$, edit distance metric d, window size w.

Ensure: Optimal path π^* and total cost $\mathbf{C}^*$.

Initialize $C_{0..m,0..n}$: $C_{0,0} \leftarrow 0$; $C_{i,0}, C_{0,j} \leftarrow \infty$ for $i, j > 0$.
for $i \leftarrow 1$ **to** m **do**
  for $j \leftarrow \max(1, i-w)$ **to** $\min(n, i+w)$ **do**
    $C_{i,j} \leftarrow d(x_i, y_j) +$
      $\min\{C_{i-1,j}, C_{i,j-1}, C_{i-1,j-1}\}$
  end for
end for
$\mathbf{C}^* \leftarrow C_{m,n}$
$\pi^* \leftarrow []$, $(i, j) \leftarrow (m, n)$
while $(i, j) \neq (0, 0)$ **do**
  Prepend (i, j) to π^*
  $(i, j) \leftarrow \underset{(i',j') \in \{(i-1,j),(i,j-1),(i-1,j-1)\}}{\operatorname{argmin}} C_{i',j'}$
end while
return π^*

tokens. The local dissimilarity $d(x_i, y_j)$ between tokens is computed using an appropriate distance metric, with Levenshtein distance serving as our primary choice due to its effectiveness in capturing token-level edits.

To ensure computational efficiency while maintaining alignment quality, we implement the **Sakoe-Chiba Band** constraint: $W = \{(i, j) \mid |i - j| \leq w\}$ This restriction limits the search space to a diagonal band of width w, reducing computational complexity from $O(k \cdot m)$ to $O(w \cdot \max(k, m))$. The window size w is carefully selected to balance alignment accuracy and computational overhead, with the additional constraint that $\Delta pos < w \wedge \Delta pos < k$, where Δpos denotes the sequence position deviation between matched tokens. The optimal alignment path $\pi^* = [(i_1, j_1), \ldots, (i_L, j_L)]$ is obtained through backtracking from $C_{k,m}$ to $C_{1,1}$, following the minimal cost path through the accumulated distance matrix. This path establishes the crucial correspondence that enables subsequent probability transfer between token sequences.

4.2 Pipeline

TokenTiming, as formally presented in Alg. 2, integrates DTW alignment into a cohesive speculative decoding pipeline, enabling efficient cross-tokenizer acceleration while preserving output distribution losslessness.

4.2.1 Draft Token Calculation

At each decoding iteration, the draft model M_d autoregressively generates a token sequence $D = (d_1, \ldots, d_k)$ from the current prefix $\mathbf{P}$. To bridge the **heterogeneous tokenizer** gap, we employ a dual-conversion pipeline: D is first encoded into an intermediate string representation $\mathbf{S}$ using Tokenizer_d, then decoded into the target vocabulary space using Tokenizer_t to yield **Proxy Target Tokens** $\mathbf{T} = (t_1, \ldots, t_m)$. Notably, the cardinality mismatch $m \neq k$ frequently arises due to fundamental tokenizer disparities, necessitating our DTW alignment mechanism.

4.2.2 Verification via Speculative Sampling

Leveraging the mapped probability distribution $\{p(t_j)\}$ derived from DTW alignment, the target model M_t performs parallel verification through a single forward pass over the proposed sequence T, computing the true conditional probabilities $\{q(t_j)\}$. Tokens are sequentially accepted according to the speculative decoding criterion: Accept t_j if $r < \min\left(1, \frac{q(t_j)}{p(t_j)}\right)$ where $r \sim U(0, 1)$. This mechanism guarantees that the output distribution exactly matches that of the target model, preserving generative quality while enabling acceleration. The verification process terminates at the first rejection instance, yielding γ accepted tokens where $0 \leq \gamma \leq m$.

4.2.3 Prefix Update

Upon accepting γ tokens $T^* = (t_1, \ldots, t_\gamma)$, the target model generates the subsequent token distribution $P_{M_t}(\cdot \mid \mathbf{P}, t_1, \ldots, t_\gamma)$. We sample $t_{\gamma+1}$ from this distribution and update the decoding prefix as $\mathbf{P} \leftarrow \mathbf{P} \oplus (t_1, \ldots, t_\gamma, t_{\gamma+1})$. This completes one decoding iteration, preparing the system for the next draft generation and verification cycles.

5 Experiments

5.1 Experiment Settings

LLM Backbones To demonstrate our method's effectiveness, we conducted experiments on a diverse set of LLM model pairs, detailed in Tab. 4. Our selection spans various architectures, sizes, and specializations to ensure a comprehensive evaluation. The target models include common dense (e.g., Meta-Llama-3.1-70B, Phi-4)(Dubey et al., 2024; Abdin et al., 2024), distilled (DeepSeek-R1-Distill-Llama-70B, Deepseek-

Algorithm 2 TokenTiming

Require: Prefix $\mathbf{P}$, draft M_d, target M_t, length k
Ensure: Full sequence $\mathbf{X}$

$\mathbf{X} \leftarrow \mathbf{P}$
while last token of $\mathbf{X} \neq$ <EOS> **do**
  $D \leftarrow M_{\mathrm{d}}.\mathrm{generate}(\mathbf{X}, k)$
  $\mathbf{S} \leftarrow \mathrm{Tokenizer}_d.\mathrm{encode}(D)$
  $T \leftarrow \mathrm{Tokenizer}_t.\mathrm{decode}(\mathbf{S})$
  $\pi^* \leftarrow \mathrm{DTW}(D, T)$
  $\gamma \leftarrow 0$
  for $j = 1$ to $|T|$ in parallel **do**
    $q(t_j) \leftarrow P_{M_t}(t_j \mid \mathbf{X} \oplus T_{1:j-1})$
    $p(t_j) \leftarrow \mathrm{MapProbabilities}(p(T), \pi^*, j)$
    $r \sim U(0, 1)$
    if $r < \min(1, q(t_j)/p(t_j))$ **then**
      $\gamma \leftarrow \gamma + 1$
    else
      break
    end if
  end for
  $\mathbf{X} \leftarrow \mathbf{X} \oplus T_{1:\gamma}$ ▷ Accept γ tokens
  $t_{\gamma+1} \leftarrow \mathrm{Sample}(P_{M_t}(\cdot \mid \mathbf{X}))$
  $\mathbf{X} \leftarrow \mathbf{X} \oplus t_{\gamma+1}$
end while
return $\mathbf{X}$

R1-Distill-Qwen-1.5B)(DeepSeek-AI et al., 2025), and Mixture-of-Experts (Qwen3-30B-A3B)(Yang et al., 2025) architectures, with several being optimized for reasoning. Ranging from 14B to 70B parameters, these models confirm our method's scalability. For draft models, we showcase the framework's portability by using small, heterogeneous, off-the-shelf models. These include inchoate pre-trained (OPT-350M)(Zhang et al., 2022), instruction-tuned (Qwen2.5-0.5B), and even an extremely compact fine-tuned model (Vicuna-68M)(Zheng et al., 2023). The vast size disparity underscores the strong efficiency of our approach, as draft models as small as 68M can effectively accelerate 70B targets.

Generation Settings To evaluate our algorithm, we employ Spec-Bench (Xia et al., 2024), a comprehensive benchmark designed for assessing Speculative Decoding across diverse scenarios, including translation, summarization, question answering, reasoning, and coding. Spec-Bench integrates CNN/Daily Mail (Nallapati et al., 2016), WMT14 DE-EN, Natural Questions (Kwiatkowski et al., 2019), and GSM8K (Cobbe et al., 2021) as the primary datasets for these scenarios. Using this dataset, we perform 480 generations for twenty-five model pairs. All model hyperparameter settings (such as temperature, top_p, etc.) adopt the default settings from the Hugging Face model library. The detailed hyperparameters for the models used in this experiment are shown in Tab. 5.

Metrics We assess generation efficiency and quality via: (1) **Tokens Per Second (TPS)**—tokens per second, averaged over the full sequence; (2) **Accept rate**—fraction of draft tokens accepted by the target model in one speculative step; (3) **Speedup**—wall-clock time of autoregressive (AR) decoding divided by that of speculative decoding; (4) **Time to First Token (TTFT)**—time from prompt availability to first token emission; (5) **Inter-Token Latency (ITL)**—average latency between consecutive tokens; (6) **Rep-N (Repetition-N)**—proportion of duplicate n-grams among all n-grams in the output, which measures diversity (Shao et al., 2019).

5.2 Experiment Results

5.2.1 Overall Results

Tab. 1 presents a comprehensive evaluation of our proposed method, TokenTiming, against autoregressive (AR) decoding and a recent speculative decoding algorithm, TLI (Timor et al., 2025a)[2], which is designed for heterogeneous vocabularies. The experiments span a diverse set of large-scale target models, including DeepSeek-R1-Distill-Llama-70B model, Llama-3.1-70B, Qwen3-30B, Qwen3-32B, and Phi-4, paired with various small draft models such as Deepseek-R1-Distill-Qwen-1.5B (abbreviated as DQwen-1.5B in Tab. 1). Performance is primarily measured by the speedup ratio relative to the AR baseline and the absolute TPS.

The results demonstrate the superior performance of our proposed TokenTiming algorithm. Across all tested target models, TokenTiming consistently achieves a higher speedup than both the AR baseline and the TLI method. For instance, when accelerating the Qwen3-32B model, TokenTiming achieves a remarkable speedup of up to 1.57× (using the Qwen3-0.6B draft model), which is a significant improvement over the maximum speedup of 1.33× achieved by TLI. Similarly, for the DeepSeek-R1-Distill-Llama-70B model, TokenTiming reaches a speedup of 1.45×, whereas

[2]RDK (Timor et al., 2025b) is not included in the baseline due to the inaccessible code source.

Target Model	Draft Model	$\frac{\mathcal{V}_d \cap \mathcal{V}_t}{\mathcal{V}_d \cup \mathcal{V}_t}$	$\frac{\mathcal{V}_d \cap \mathcal{V}_t}{\max\{\mathcal{V}_d, \mathcal{V}_t\}}$	TPS		Accept Rate		Speedup	
				TLI	Ours	TLI	Ours	TLI	Ours
DeepSeek-R1 -Distill-Llama-70B	Autoregressive	-	-	14.68		-	-	-	-
	Qwen2.5-0.5B	0.643	0.722	15.68	**19.26**	0.34	**0.40**	1.07	**1.31**
	Qwen3-0.6B	0.643	0.722	14.14	15.35	0.28	0.29	0.96	1.05
	DQwen-1.5B	0.643	0.722	17.76	19.08	0.29	0.37	1.21	1.30
	Vicuna-68M	0.064	0.075	14.79	15.43	0.23	0.26	1.01	1.05
	OPT-350M	0.319	0.337	16.03	**21.35**	0.19	**0.31**	1.09	**1.45**
Llama-3.1-70B	Autoregressive	-	-	13.55		-	-	-	-
	Qwen2.5-0.5B	0.643	0.722	16.35	14.38	0.34	0.31	1.21	1.06
	Qwen3-0.6B	0.643	0.722	14.18	15.03	0.23	0.33	1.05	1.11
	DQwen-1.5B	0.643	0.722	16.67	**18.25**	0.20	0.22	1.23	1.35
	Vicuna-68M	0.064	0.075	15.58	16.53	0.19	0.06	1.15	1.22
	OPT-350M	0.319	0.337	16.56	**17.84**	0.35	**0.25**	1.22	**1.32**
Qwen3-30B-A3B	Autoregressive	-	-	9.80		-	-	-	-
	Qwen2.5-0.5B	0.999	0.999	10.74	11.37	0.36	0.37	1.10	1.16
	Qwen3-0.6B	1.000	1.000	11.71	**11.90**	0.44	**0.45**	1.19	**1.21**
	DQwen-1.5B	0.999	0.999	12.23	**12.90**	0.41	**0.45**	1.25	**1.32**
	Vicuna-68M	0.055	0.063	8.33	10.89	0.26	0.34	0.85	1.11
	OPT-350M	0.265	0.279	9.99	9.95	0.23	0.33	1.02	1.02
Qwen3-32B	Autoregressive	-	-	15.77		-	-	-	-
	Qwen2.5-0.5B	0.999	0.999	20.97	**24.47**	0.42	0.38	1.33	**1.55**
	Qwen3-0.6B	1.000	1.000	19.16	**24.78**	0.43	0.42	1.21	**1.57**
	DQwen-1.5B	0.999	0.999	18.83	**21.90**	0.35	0.39	1.19	**1.39**
	Vicuna-68M	0.055	0.063	14.67	18.92	0.19	0.33	0.93	1.20
	OPT-350M	0.265	0.279	17.28	**24.01**	0.18	**0.21**	1.10	**1.52**
Phi-4	Autoregressive	-	-	23.14		-	-	-	-
	Qwen2.5-0.5B	0.649	0.654	17.64	**35.58**	0.19	**0.39**	0.76	**1.54**
	Qwen3-0.6B	0.649	0.654	26.80	28.59	0.34	0.27	1.16	1.24
	DQwen-1.5B	0.649	0.654	22.91	24.07	0.15	0.19	0.99	1.04
	Vicuna-68M	0.078	0.095	22.19	28.09	0.13	0.05	0.96	1.21
	OPT-350M	0.400	0.429	31.64	28.33	0.21	0.20	1.37	1.22

Table 1: Performance comparison between TokenTiming and baselines, AR and TLI (Timor et al., 2025a), across a diverse set of target and draft models. The primary metrics are the speedup ratio relative to AR and absolute Tokens Per Second (TPS). The results demonstrate TokenTiming's significant and consistent performance superiority over TLI. Comparatively optimal results for each target model are highlighted in **bold**.

Target Model	Draft Model	Math		Program		Translation		Summarize		QA	
		TLI	Ours	TLI	Ours	TLI	Ours	TLI	Ours	TLI	Ours
DeepSeek-R1 -Distill-Llama-70B	Qwen2.5-0.5B	1.03	**1.48**	1.05	**1.34**	1.62	**2.54**	0.94	**1.07**	0.99	**1.08**
	Qwen3-0.6B	0.95	**1.34**	0.88	**1.35**	1.01	**2.53**	1.02	**1.06**	1.11	1.08
	DQwen-1.5B	1.05	**1.58**	0.99	**1.24**	1.10	**1.65**	1.05	**1.31**	1.03	**1.53**
	Vicuna-68M	0.97	**0.98**	0.81	**1.04**	1.22	**1.44**	1.02	**1.30**	1.05	**1.22**
	OPT-350M	1.02	**1.18**	1.05	**1.11**	0.96	**2.05**	1.01	**1.09**	0.97	**1.08**
Llama-3.1-70B	Qwen2.5-0.5B	0.92	**1.34**	0.88	**1.47**	1.09	**1.10**	0.89	**1.06**	1.51	1.32
	Qwen3-0.6B	1.04	**1.57**	1.30	1.17	0.73	**1.54**	0.86	**0.91**	1.00	**1.11**
	DQwen-1.5B	0.95	**1.43**	1.21	**1.24**	1.15	**1.64**	1.11	**1.34**	1.22	**1.64**
	Vicuna-68M	1.21	**1.25**	1.05	**1.36**	0.84	**1.22**	1.07	**1.22**	1.04	**1.21**
	OPT-350M	0.94	**1.36**	1.14	**1.44**	0.99	**1.33**	1.12	**1.14**	1.26	**1.32**
Qwen3-30B-A3B	Qwen2.5-0.5B	1.15	**1.62**	0.76	**1.23**	1.31	**1.45**	1.21	**2.03**	0.71	**1.04**
	Qwen3-0.6B	1.24	1.05	0.92	**1.25**	1.21	1.06	0.82	**1.11**	0.57	**0.85**
	DQwen-1.5B	1.14	**1.35**	0.92	**1.32**	1.11	**1.16**	0.91	**1.10**	0.88	**0.95**
	Vicuna-68M	1.02	**1.35**	1.13	**1.36**	0.95	**1.52**	1.25	**1.44**	1.03	**1.34**
	OPT-350M	1.06	1.04	0.84	**1.02**	0.75	**0.91**	0.96	**1.18**	0.74	**0.77**
Qwen3-32B	Qwen2.5-0.5B	1.44	**2.53**	1.29	**1.80**	1.91	**2.49**	1.41	1.28	1.01	**1.13**
	Qwen3-0.6B	1.16	**2.05**	1.01	**2.47**	1.50	**1.61**	1.35	**1.80**	1.44	1.41
	DQwen-1.5B	1.13	**1.15**	1.12	**1.15**	1.01	**1.12**	0.97	**1.14**	0.97	**1.05**
	Vicuna-68M	0.83	**1.45**	0.93	**1.39**	0.94	**1.44**	1.25	**1.63**	1.06	**1.44**
	OPT-350M	1.16	**1.92**	1.77	**2.16**	1.66	1.24	0.91	**1.05**	0.91	**1.52**
Phi-4	Qwen2.5-0.5B	0.90	**1.57**	0.94	**1.60**	0.67	**1.31**	0.71	**1.55**	0.76	**1.55**
	Qwen3-0.6B	1.04	**1.51**	1.21	**1.61**	1.05	**1.20**	1.09	1.07	1.48	1.21
	DQwen-1.5B	1.21	**1.45**	1.32	**1.75**	1.06	**1.52**	1.07	**1.74**	1.12	**1.45**
	Vicuna-68M	0.87	**1.52**	0.83	**0.88**	0.74	**1.41**	0.87	**1.64**	1.29	**1.30**
	OPT-350M	1.64	1.34	1.30	**1.41**	1.43	**1.52**	1.59	1.42	1.43	**1.55**

Table 2: Speedup comparison between TokenTiming and baseline across different task categories. The table shows results for Math, Programming, Translation, Summarization, and Question Answering tasks. Results where our method outperforms the baseline are highlighted in **bold**.

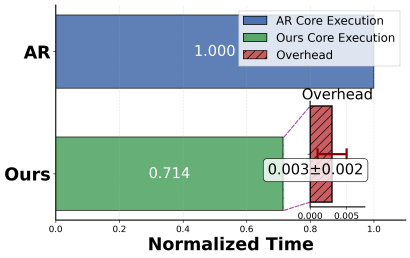

Figure 3: Total Time Cost (with TokenTiming overhead)

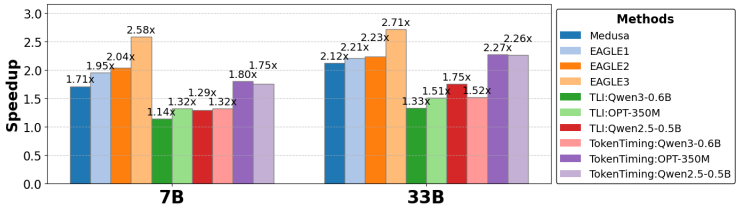

Figure 4: Speed-up vs. homogeneous-vocabulary SOTA on various target model scales and draft models. TokenTiming bridges the performance gap while keeping the universal draft-model advantage.

TLI's peak performance is capped at 1.09×. Furthermore, the analysis highlights the critical role of draft model selection in speculative decoding. Our results indicate that TokenTiming is highly effective at leveraging different draft models to maximize performance. For example, with the Llama-3.1-70B target, TokenTiming paired with OPT-350M yields a 1.32× speedup. For the Phi-4 model, TokenTiming achieves its peak performance of 1.54× speedup with the Qwen2.5-0.5B draft model, again substantially outperforming TLI's best result of 1.37×.

This enhanced speedup is often correlated with a robust token accept rate. In many configurations, TokenTiming not only delivers higher TPS but also maintains a competitive or even higher accept rate than TLI. This suggests that TokenTiming's mechanism is more efficient at generating candidate sequences that are accepted by the target model, leading to more effective acceleration.

5.2.2 Approaching Homogeneous-Vocabulary SD Acceleration

Comparable Performance to Homogeneous-Vocabulary SOTA. As shown in Fig. 4, on 7B-target model, the best homogeneous-vocabulary SD method (EAGLE-3) reaches 2.58× speed-up, while the strongest heterogeneous-vocabulary baseline (TLI) peaks at 1.32×. TokenTiming with OPT-350M closes this gap to 1.80×, only 0.78× away from EAGLE-3. On 33B-target model the trend is identical: TokenTiming-OPT-350M delivers 2.27×, which is within 0.44× of EAGLE-3 (2.71×) and already surpasses Medusa (1.71×) and EAGLE-1 (2.21×).

More Draft Model Choices without Re-training. Medusa/EAGLE requires extra parameters integrated into the target model or costly draft-head re-training when the target is switched. TokenTiming is plug-and-play: any off-the-shelf 68M–350M model can be used while instantaneously yielding 1.05×–2.27× speed-up across four diverse draft models, demonstrating the flexibility that homogeneous-vocabulary SOTA cannot provide.

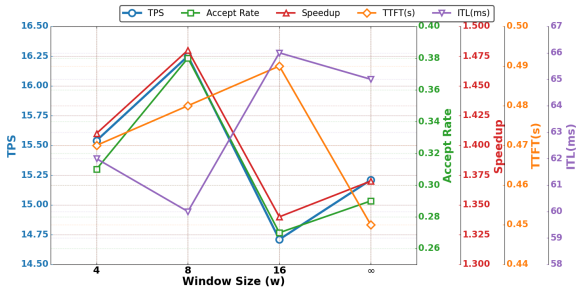

Figure 5: Performance metrics under various settings of w: $w = 4$, $w = 8$, $w = 16$, and $w = \infty$.

5.2.3 Optimization on DTW Constraints

We conducted experiments under different settings of w: $w = 4$, $w = 8$, $w = 16$, and $w = \infty$, where $w = \infty$ corresponds to the case without the Sakoe-Chiba Band. The results are shown in the Fig. 5. By selecting an appropriate window size w, the acceleration effect of TokenTiming on SD can be partially improved, resulting in higher TPS, Accept Rate, and Speedup. As shown in Fig. 6, the upper bound of Δpos convergence exceeds the setting of our best-performing $w = 8$ in some model pairs, and the proportion of cases where $\Delta pos > 8$ is non-trivial. However, compared to $w = \infty$, $w = 8$ still achieves performance improvement. This result indicates that appropriate constraints can better preserve the probability distribution of tokens.

5.2.4 Task-Specified Performance

Tab. 2 compares the task-level speedup of TokenTiming against TLI.TokenTiming consistently outperforms TLI on all five evaluated tasks. For mathematics, TokenTiming reaches 2.53× with the

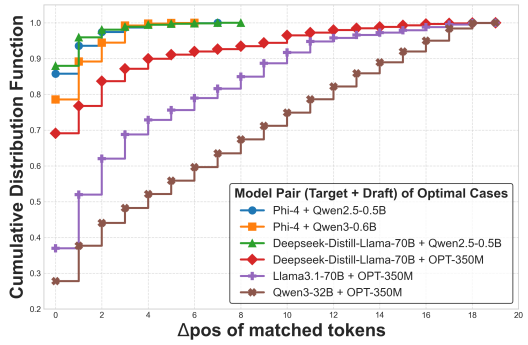

Figure 6: Cumulative Distribution Function of Δpos of DTW matched tokens. $w = \infty$. Δpos of DTW matched tokens converges to an upper bound.

Figure 7: Repetition-N (Shao et al., 2019) in settings with a different number of generated tokens.

pair Qwen3-32B + Qwen2.5-0.5B, while TLI only achieves 1.44×. Similar gaps appear in summarization (2.54× vs. 1.62×) and translation (1.60× vs. 0.94×). Although the absolute speedup on programming and QA is slightly lower, TokenTiming still maintains a noticeable margin.

The gain is tightly correlated with the capability of the draft model. Strong draft models such as Qwen2.5-0.5B and Qwen3-0.6B boost the accept rate and push speedup beyond 2× on reasoning-intensive tasks. Conversely, light-weight drafts (OPT-350M, Vicuna-68M) reduce the advantage, especially on math and code generation where token-level accuracy is crucial. Nevertheless, even with these weaker drafts, TokenTiming remains superior to TLI, confirming the robustness of the DTW alignment when vocabulary overlap is limited.

5.2.5 Tradeoff Analysis

Fig. 3 shows that the integration of Dynamic Token Warping into the decoding pipeline introduces an additional computational step[3], creating a negligible time overhead (0.1% to 0.5% of overall runtime) compared to simpler, direct token-matching methods. This overhead is an inherent trade-off for the enhanced flexibility and accuracy of the alignment process. However, our empirical analysis indicates that this additional cost is effectively amortized by the substantial gains in overall generation throughput (1.4× speedup).

[3]In Appendix F, it is shown that for the CPU/GPU stream timeline, the blocking time introduced by TokenTiming in one decoding cycle is only 663μs, which is trivial compared to other parts.

5.2.6 DTW Bridges the Gap of Mismatched Vocabularies

Fig. 6 shows that the sequence position deviation of matched proxy target token and draft token is not exclusively zero, where position means the positional order in the two token sequence. If the vocabularies of the two models were perfectly aligned, the DTW matching result would converge precisely on the diagonal. The presence of this deviation, therefore, demonstrates that our algorithm is effectively matching these imperfectly aligned vocabularies. We observe that a significant proportion of the deviation is non-zero, which highlights a key advantage of the DTW algorithm over the TLI algorithm. TokenTiming is capable of handling skewed probability distribution mappings, rather than one-to-one mappings, in a sequential manner, thereby increasing the accept rate.

5.2.7 Eliminate Potential Inflated Performance

While our results demonstrate significant speedup, scenarios such as repetitive generation loops can artificially inflate this metric. In these cases, the draft model easily predicts subsequent repetitive tokens, leading to a near-perfect accept rate and a high number of accepted tokens per step. This corresponds to technically fast but non-meaningful generation, as shown in Fig. 7. To avoid distortion, we exclude such pathological repetitions (approximately **15%** of test samples) from our main analysis, following prior work showing they can amplify token probabilities and inflate speed metrics (Xu et al., 2022). The remaining 85% of samples, covering standard tasks, thus provide a representative assessment of typical generation behavior.

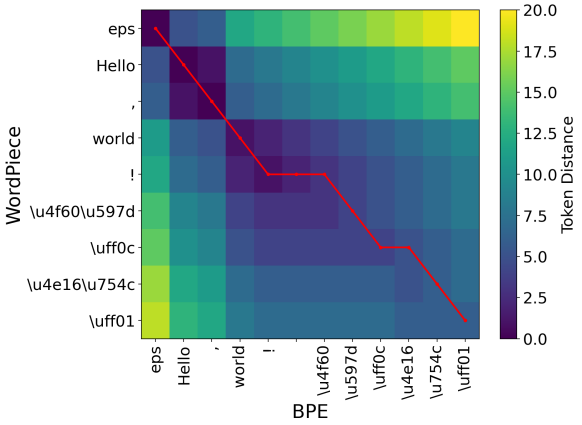

Figure 8: Alignment path for fragmentarily tokenized prompt. The prompt was constructed by inserting special symbols which have different tokenized results between the draft and target tokenizers and which are highly-fragmented at least in one tokenizer.

5.2.8 Modes on Highly-fragmented Tokenizations

Fig. 8 shows the alignment paths of the tokenized results for the prompt with special symbols by two common tokenization methods, WordPiece and Byte-Pair Encoding(BPE) (Sennrich et al., 2016). We can observe that the special symbols are converted into the UTF8 encoding format when used for calculating Token Distance. The UTF8 format has a minimum resolution of one byte during tokenization (Wang et al., 2019), and it has strong specificity in the alignment path. Even if the granularity of tokenization for special characters by the two tokenizers is different, the token at the end of the two tokenization sequences is still aligned. This is consistent with the probability transfer assignment logic of TokenTiming, that is, the probability of the end token of the draft token sequence block and the target token sequence block is equal. The alignment result demonstrates robustness.

6 Conclusion

This paper introduced TokenTiming, a dynamic alignment algorithm that eliminates the shared-vocabulary constraint of existing speculative decoding methods. By employing a dynamic warping approach to build a lossless probability mapping between heterogeneous vocabularies, TokenTiming enables any off-the-shelf model to serve as a draft model without re-training. Experiments show our method achieves up to a 1.57× speedup over previous state-of-the-art speculative decoding algorithms across various generation tasks. By removing the fundamental bottleneck of heterogeneous vocabularies, TokenTiming makes speculative decoding a more versatile and practical tool for LLM acceleration.

Limitations

The probability form calculated in TokenTiming is one-hot, i.e., it directly transfers the probability of the top-1 token. The calculation of DTW is based on the character granularity rather than the semantic granularity. There are certain differences in the alignment effect between languages with different word tokenization granularities.

Ethics Statement

TokenTiming focuses on expanding the application scope of speculative decoding models. The related research involves time series alignment and language tokenization techniques, aiming to enhance the decoding efficiency of large language models. There are no technical security risks or risks of technical abuse.

Acknowledgements

We sincerely thank the reviewers, the area chairs, and the program chairs for their constructive comments. This work was supported by the Pioneer R&D Program of Zhejiang (No. 2024C01021) and Zhejiang Province "Leading Talent of Technological Innovation Program" (No. 2023R5214).

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, and 8 others. 2024. Phi-4 technical report. *Preprint*, arXiv:2412.08905.

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. 2024. Hydra: Sequentially-Dependent Draft Heads for Medusa Decoding. *Preprint*, arXiv:2402.05109.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads. *Preprint*, arXiv:2401.10774.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John

Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Jian Chen, Vashisth Tiwari, Ranajoy Sadhukhan, Zhuoming Chen, Jinyuan Shi, Ian En-Hsu Yen, and Beidi Chen. 2024. Magicdec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. *arXiv preprint arXiv:2408.11049*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2024. Break the Sequential Dependency of LLM Inference Using Lookahead Decoding. *Preprint*, arXiv:2402.02057.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Siqi Kou, Lanxiang Hu, Zhezhi He, Zhijie Deng, and Hao Zhang. 2024. CLLMs: Consistency Large Language Models. In *Forty-first International Conference on Machine Learning*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024a. EAGLE-2: Faster Inference of Language Models with Dynamic Draft Trees. *Preprint*, arXiv:2406.16858.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024b. EAGLE: Speculative Sampling Requires Rethinking Feature Uncertainty. *Preprint*, arXiv:2401.15077.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2025. EAGLE-3: Scaling Up Inference Acceleration of Large Language Models via Training-Time Test. *Preprint*, arXiv:2503.01840.

Xiaoxuan Liu, Cade Daniel, Langxiang Hu, Woosuk Kwon, Zhuohan Li, Xiangxi Mo, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. 2024. Optimizing Speculative Decoding for Serving Large Language Models using Goodput. *Preprint*, arXiv:2406.14066.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 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*. ACM.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gu̇lçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

OpenAI. 2025. gpt-oss-120b and gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

H. Sakoe and S. Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu, and Xiaoyan Zhu. 2019. Long and diverse text generation with planning-based hierarchical variational model. *Preprint*, arXiv:1908.06605.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama.

Ziteng Sun, Jae Hun Ro, Ahmad Beirami, and Ananda Theertha Suresh. 2024. Optimal block-level draft verification for accelerating speculative decoding. *arXiv preprint arXiv:2403.10444*.

Nadav Timor, Jonathan Mamou, Daniel Korat, Moshe Berchansky, Gaurav Jain, Oren Pereg, Moshe Wasserblat, and David Harel. 2025a. Accelerating llm inference with lossless speculative decoding algorithms for heterogeneous vocabularies. *Preprint*, arXiv:2502.05202.

Nadav Timor, Jonathan Mamou, Daniel Korat, Moshe Berchansky, Oren Pereg, Moshe Wasserblat, Tomer Galanti, Michal Gordon, and David Harel. 2025b. Distributed speculative inference (dsi): Speculation parallelism for provably faster lossless language model inference. In *International Conference on Learning Representations*.

Nadav Timor, Jonathan Mamou, Oren Pereg, Hongyang Zhang, and David Harel. 2025c. Out-of-vocabulary sampling boosts speculative decoding. *Preprint*, arXiv:2506.03206.

Changhan Wang, Kyunghyun Cho, and Jiatao Gu. 2019. Neural machine translation with byte-level subwords. *Preprint*, arXiv:1909.03341.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. 2024. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 7655–7671, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Jin Xu, Xiaojiang Liu, Jianhao Yan, Deng Cai, Huayang Li, and Jian Li. 2022. Learning to break the loop: Analyzing and mitigating repetitions for neural text generation. *Preprint*, arXiv:2206.02369.

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

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *Preprint*, arXiv:2205.01068.

Weilin Zhao, Tengyu Pan, Xu Han, Yudi Zhang, Ao Sun, Yuxiang Huang, Kaihuo Zhang, Weilun Zhao, Yuxuan Li, Jianyong Wang, and 1 others. 2025. Fr-spec: Accelerating large-vocabulary language models via frequency-ranked speculative sampling. *arXiv preprint arXiv:2502.14856*.

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.

A Consistency and Losslessness Proof

Before proceeding to the core proof, we must formalize the process by which the draft probability distribution $p(t)$ is generated and prove its consistency across mismatched vocabularies. Let $\mathcal{V}_d$ and $\mathcal{V}_t$ be the vocabularies for the draft and target models, respectively. The draft model, M_d, generates a sequence of tokens $D = (d_1, \ldots, d_K)$ where $d_i \in V_d$. This output is transformed into a distribution $p(t)$ over $\mathcal{V}_t$ through a two-stage procedure:

A.1 Deterministic Re-tokenization and String Consistency

The first stage involves a mapping $F_{\text{retokenize}} : \mathcal{V}_d^* \to \mathcal{V}_t^*$. Let f^d_{decode} and f^t_{encode} be the decoding function of the draft and the encoding function of the target, respectively.

Lemma 1 (String-level Consistency)**.** *The proxy target token sequence* $T = (t_1, \ldots, t_m) = f^t_{encode}(f^d_{decode}(D))$ *is identical to the draft sequence* D *in the string space.*

Proof. Let $S = f^d_{\text{decode}}(D)$ be the string generated by the draft model. Modern tokenizers (e.g., BPE, SentencePiece) satisfy the **lossless invertibility** property for any valid UTF-8 string: $f^t_{\text{decode}}(f^t_{\text{encode}}(S)) \equiv S$. Substituting the definition of T:

$$f^t_{\text{decode}}(T) = f^t_{\text{decode}}(f^t_{\text{encode}}(S)) = S = f^d_{\text{decode}}(D)$$

Thus, while the token boundaries differ ($K \neq m$), the underlying character streams are identical, ensuring the verification process targets the exact same content. □

A.2 Losslessness of Probability Mapping

The alignment $\pi^* = \text{DTW}(D, T)$ maps indices of D to T. We define the proposal probability $p(t_j)$ based on the following two non-trivial mapping cases:

Case 1: Many-to-One (Draft "a", "b"→ Target "ab"). In this case, multiple draft tokens $\{d_i, \ldots, d_{i+n}\}$ correspond to a single target token t_j. Our implementation assigns the probability of the **terminal** draft token to the target token: $p(t_j) = P_d(d_{i+n} \mid \text{prefix}, d_i, \ldots, d_{i+n-1})$.

Proof. In the target model, $q(t_j)$ represents the probability of completing the string unit "ab" given the prefix. In the draft model, since the prefix and the initial part of the unit "a" are already assumed/-fixed during the sequential alignment, the probability of the final token d_{i+n} "b" represents the draft's confidence in *completing* that specific string block. Under the assumption that M_d approximates M_t, the conditional probability of completing a semantic unit is a consistent estimator of the target's atomic probability for that unit. □

Case 2: One-to-Many (Draft "ab" → Target "a", "b"). Here, one draft token d_i covers multiple target tokens $\{t_j, \ldots, t_{j+k}\}$. Our implementation **copies** the probability: $p(t_j) = \cdots = p(t_{j+k}) = P_d(d_i)$.

Proof. Since d_i is a single token in $\mathcal{V}_d$, $P_d(d_i)$ represents the draft's total confidence in the string "ab". 1) For the first segment t_j ("a"), $P_d(d_i)$ acts as a conservative surrogate. Since "a" is a prefix of "ab", $P(\text{"a"}) \geq P(\text{"ab"})$ holds in a joint distribution. Using $p(t_j) = P_d(d_i)$ ensures p is not overestimated relative to the target's marginal. 2) For subsequent segments like t_{j+1} ("b" given "a"), the draft model has already committed to the full string "ab". Therefore, the probability of the component "b" appearing is implicitly 1.0 within that draft step. By using the original $P_d(d_i)$, we maintain a high proposal probability that matches the target model's conditional confidence, maximizing the accept rate $\min(1, q/p)$. □

B Example of Generated Text

Draft model: DeepSeek-R1-Distill-Qwen-1.5B, Target model: DeepSeek-R1-Distill-Llama-70B, Prefix: The field of Artificial Intelligence has seen tremendous growth in recent years. One of the most exciting areas of research is

---

The field of Artificial Intelligence has seen tremendous growth in recent years. One of the most exciting areas of research is Reinforcement Learning (RL), which involves training agents to make decisions in complex, uncertain environments. In this blog post, we will explore the concept of policy in reinforcement learning, its types, and the importance of policy optimization.

What is a Policy in Reinforcement Learning?

In reinforcement learning, a **policy** is a strategy that an agent uses to decide actions based on the current state of the environment. It can be thought of as a function, often denoted as $\pi(a|s)$, which takes the current state (s) and returns an action (a) or a probability distribution over possible actions. The goal of the agent is to learn an optimal policy that maximizes the cumulative reward over time.

Types of Policies

Policies in RL can be categorized in different ways. Here are some common types:

1. **Deterministic Policy**: - A deterministic policy always selects the same action for a given state. It is a direct mapping from states to actions. For example,π(s) = a.
2. **Stochastic Policy**: - A stochastic policy, on the other hand, selects actions probabilistically. It outputs a probability distribution over possible actions given a state. This is often useful in exploration-exploitation trade-offs, where the agent might sometimes choose a suboptimal action to discover better ones.
3. **Parametric Policy**: - Parametric policies are defined by a set of parameters. These parameters can be adjusted during training to improve the policy. Examples include neural networks, where the weights and biases are the parameters.
4. **Non-Parametric Policy**: - Non-parametric policies do not rely on a fixed set of parameters. Instead, they might be represented by lookup tables or other structures that can grow with the data. These are less common in deep RL settings.

C Hardware Configuration Specification

The hardware configuration of our experiment environments can be seen in Tab. 3.

D Related Algorithms

D.1 Token-level Intersection

Token-Level Intersection, Alg. 3, for heterogeneous draft models has been recently introduced by Timor et al. (2025a), which normalizes the intersected

Component	Specification
CPU	2 x Intel Xeon Platinum 8558 (Total 96 Cores / 192 Threads)
RAM	2.0 TiB
GPU	2 x NVIDIA H100 80GB HBM3 - Architecture: Hopper - VRAM per GPU: 80 GB - Total VRAM: 160 GB - Interconnect: PCIe Gen 5 x16
Software	NVIDIA Driver: 570.133.20 CUDA Version: 12.8

Table 3: Hardware Environment Configuration

Algorithm 3 (Token-Level Intersection, TLI (Timor et al., 2025a)), an iteration of speculative decoding for heterogeneous vocabularies with token-level rejection sampling verification

Input: Probability distributions p and q over vocabularies T and D, respectively. Drafting lookahead $i \in \mathbb{N}$.
Output: A sequence of tokens from T, containing between 1 and $i+1$ tokens.
Procedure:
Define a probability distribution q' over the vocabulary $T \cap D$ such that $q'(x) = \frac{q(x)}{\sum_{t \in T} q(t)}$ if $x \in T$ and $q'(x) = 0$ otherwise.
Run Standard Speculative Decoding with p, q', i, c.

distribution by zeroing out all out-of-vocabulary mass in the probability conversion.

D.2 Standard Speculative Decoding

Standard Speculative Decoding (Leviathan et al., 2023; Chen et al., 2023), Alg. 4, accelerates autoregressive generation by pre-generating candidate tokens with a draft model for batch verification, converting sequential decoding into parallel computation. The algorithm ensures distribution consistency via probability ratio acceptance but is limited to homogeneous vocabularies, which TokenTiming overcomes through dynamic alignment.

E LLM Backbones

The experiment selects LLMs with diverse architectures and parameter scales as target and draft models, enabling universal validation of TokenTiming through diversified model configurations. The detailed model configurations are listed in Tab. 4.

Algorithm 4 Standard Speculative Decoding (Leviathan et al., 2023; Chen et al., 2023)

Input: Probability distributions p and q over a vocabulary T. Drafting lookahead $i \in \mathbb{N}$. An input prompt c.
Output: A sequence of tokens from T, containing between 1 and $i+1$ tokens.
Procedure:
for $j \leftarrow 1, \ldots, i$ **do**
  Sample a draft token from the drafter, $d_j \sim q_{c \oplus d_1 \oplus \ldots \oplus d_{j-1}}$.
end for
In parallel, compute the $i+1$ logits of the target model: $p_c,\ p_{c \oplus d_1},\ \cdots,\ p_{c \oplus d_1 \oplus \cdots \oplus d_i}$.
for $j \leftarrow 1, \ldots, i$ **do**
  Let $x \leftarrow c \oplus d_1 \oplus \cdots \oplus d_{j-1}$.
  if $p_x(d_j) \leq q_x(d_j)$ **then**
    With probability $1 - \frac{p_x(d_j)}{q_x(d_j)}$, **reject** d_j and **break**.
  end if
  Accept the draft token d_j.
end for
Let $j \in \{0, 1, \ldots, i\}$ be the number of accepted drafts.
Set $x \leftarrow c \oplus d_1 \oplus \ldots \oplus d_j$.
Sample $t \sim r_x$ where $r_x(\cdot)$ is the modified distribution.
Return $d_1, \ldots, d_j, t$.

Model	Type	Params
Target Models		
Llama-3.1-70B	Dense	70B
DeepSeek-R1-Distill-LLama-70B	Distill, Reasoning	70B
Phi-4	Dense	14B
Qwen3-32B	Dense, Reasoning	32B
Qwen3-30B-A3B	MoE, Reasoning	30B
Draft Models		
Vicuna-68M	Fine-Tuned	68M
Qwen3-0.6B	Dense, Reasoning	0.6B
Qwen2.5-0.5B(-Instruct)	Instruction-Tuned	0.5B
OPT-350M	Dense	350M

Table 4: Overview of LLM backbones used in our experiments. The selection covers a wide range of model types and sizes. Roles are indicated by subheadings.

F Generation Hyperparameter

The hyperparameter configurations for text generation of both target models and draft models in our experiment can be seen in Tab. 5.

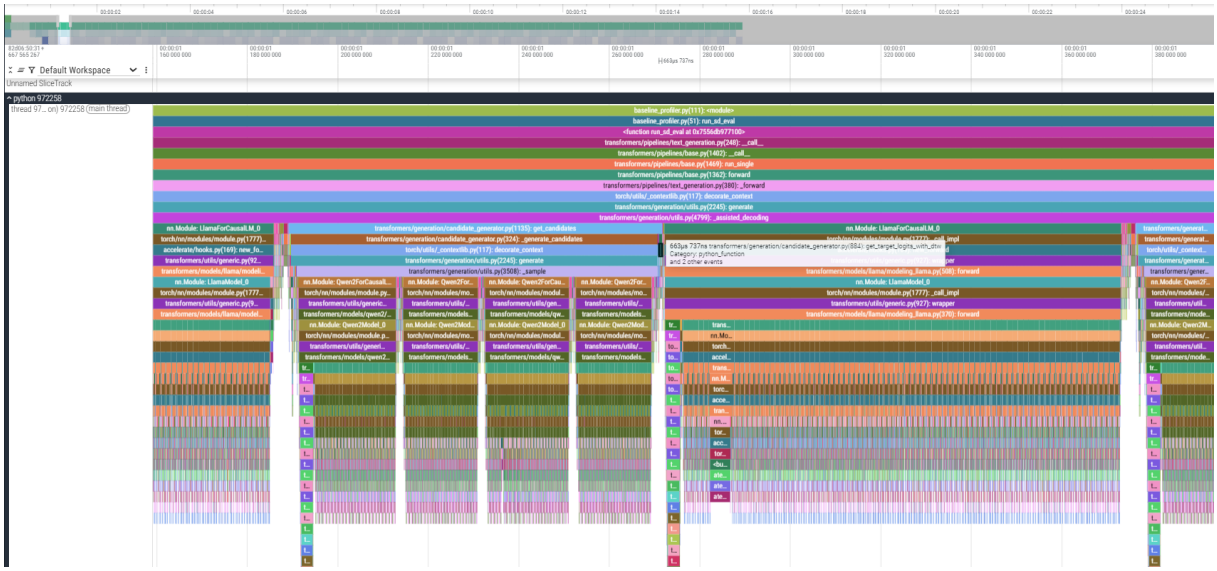

Figure 9: Trace visualization. The process segment for target logits calculation with DTW-related operations takes only $663\mu s$, highlighting its extremely short duration.

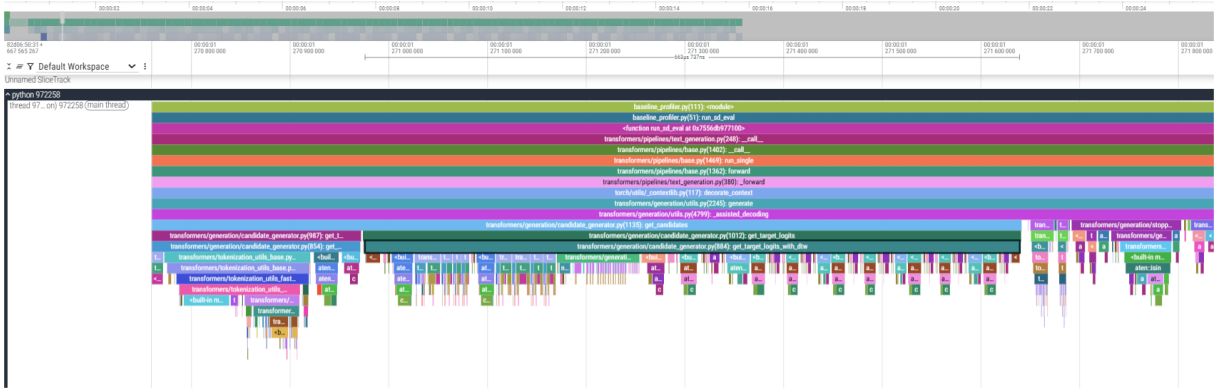

Figure 10: Enlarged view of the process segment for target logits calculation with DTW-related operations in the previous trace.

Model	Temperature	Top P	Top K	Repetition Penalty
Llama-3.1-70B	0.6	0.9	-	-
Deepseek-R1-Distill-Llama-70B	0.6	0.95	-	-
Qwen3-32B	0.6	0.95	20	-
Qwen3-30B-A3B	0.7	0.8	20	-
Qwen3-0.6B	0.6	0.95	20	-
Qwen2.5-0.5B(-Instruct)	0.7	0.8	20	1.1

Table 5: The text generation hyperparameter configurations for target models and draft models in the experiment.

G Repetition-N

The repetition degree indicator is used to indirectly evaluate the diversity of the generated text. It calculates the ratio of the number of n-grams with a frequency higher than 1 in the generated text to the total number of n-grams according to Alg. 5.

H Token Distance

The Edit Distance, often referred to as Levenshtein Distance, is a measure of the similarity between two strings (tokens). It is defined as the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one token into the other.

H.1 1. Symbol Definition

Let the source token be denoted by $s = s_1 s_2 ... s_m$, and the target token be denoted by $t = t_1 t_2 ... t_n$.

The edit distance between the first i characters of s and the first j characters of t is denoted by

Algorithm 5 N-gram Repetition Calculation

Require: Text sequence S, N-gram order $N = 3$
Ensure: Repetition rates $R = [r_1, r_2, \ldots, r_N]$
Initialize frequency counters $F_1, F_2, \ldots, F_N$
Initialize repetition rates $R = [0, 0, \ldots, 0]$
for each text $t \in S$ **do**
  Tokenize t into word sequence W
  for $n = 1$ **to** N **do**
    Extract all n-grams from W of length n
    Update frequency counts in F_n
  end for
end for
for $n = 1$ **to** N **do**
  Count repeated n-grams: $C_{rep} = |\{f \in F_n : f > 1\}|$
  Count total unique n-grams: $C_{total} = |F_n|$
  $r_n = C_{rep}/C_{total}$
end for

return $R \times 100$ #Convert to percentages

$D(i, j)$. The goal is to compute $D(m, n)$.

The cost of a substitution operation is defined as: $\text{cost}_{sub}(s_i, t_j) = \begin{cases} 0 & \text{if } s_i = t_j \\ 1 & \text{if } s_i \neq t_j \end{cases}$ The cost for a deletion or an insertion is always 1.

H.2 2. Calculation Process

We use a dynamic programming approach to compute the distance. A matrix D of size $(m + 1) \times (n + 1)$ is constructed.

Step 1: Initialization The first row and the first column of the matrix are initialized to represent the edits needed to transform a prefix into an empty string, or an empty string into a prefix.

$$D(i, 0) = i \quad \text{for } i = 0, \ldots, m$$
$$D(0, j) = j \quad \text{for } j = 0, \ldots, n$$

Step 2: Recurrence Relation For every i from 1 to m and every j from 1 to n, the value of $D(i, j)$ is calculated as the minimum of three possible operations:

$$D(i, j) = \min \begin{cases} D(i-1, j) + 1 & \text{(deletion)} \\ D(i, j-1) + 1 & \text{(insertion)} \\ D(i-1, j-1) + \text{cost}_{sub}(s_i, t_j) & \text{(substitution)} \end{cases}$$

Step 3: Final Result The edit distance between the entire token s and token t is the value in the last cell of the matrix: $\text{distance}(s, t) = D(m, n)$

Target	BioMistral-7B	Qwen2.5-Coder-14B
Draft	Qwen2.5-0.5B	Qwen2.5-0.5B
TLI	1.15×	1.09×
TokenTiming	**1.31**×	**1.25**×

Table 6: Speedup on Domain-specific Models

I Trace Profiling

Tokenization, logic operation, etc., these operations performed on the CPU may cause GPU synchronization, which will cause a significant performance degradation. In Fig. 9 and 10, we analyzed the CPU and GPU trace and confirmed that the introduction of DTW did not cause any performance bottlenecks. The blocking time introduced by TokenTiming in one decoding cycle is only 663 μs, which is insignificant compared to other parts.

J Domain-specific Models

We conducted an additional experiment explicitly targeting specialized vocabulary, with results shown in Tab. 6. We fixed Qwen2.5-0.5B as the draft model and selected two domain-specific target models—-BioMistral-7B for the medical domain and Qwen2.5-Coder-14B for the code domain—-both of which exhibit systematically lower vocabulary overlap with general-purpose drafts. This setup directly evaluates whether TokenTiming remains effective under realistic domain-specific conditions.

K Settings of w

Since speculative decoding generates only a bounded number of draft tokens per decoding step, the window size w naturally resides within a finite search space. Our procedure for determining w is as follows. We first perform DTW alignment without any window constraint and collect the offset distribution. As shown in Fig. 6, we then identify, for each model pair, the offset at which the CDF reaches 0.9. These values define a narrowed and more representative search space for w. Within this reduced space, we evaluate candidate window sizes and select the setting that yields the highest mean speedup across model pairs. This procedure leads to the global configuration $w = 8$ used in our experiments. In practice, w can also be tuned adaptively within the same search space—larger w for model pairs with a later CDF plateau and smaller w for those with an earlier plateau.