FLASHMTP: FLASH MULTI-TOKEN PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

The autoregressive nature of large language models (LLMs) fundamentally constrains inference speed. Standard autoregressive decoding generates one token per forward pass, and its sequential dependency prevents parallel prediction of future tokens. The key challenge is to exploit information to predict multiple future tokens while effectively allocating candidate tokens per forward pass. To address this problem, we propose FlashMTP (Flash Multi-Token Prediction), a plug-and-play, training-free adaptive multi-token prediction method. FlashMTP expands the effective candidate set using the outputs at all evaluated candidate positions and dynamically allocates the candidate tokens of each pass according to path acceptance probabilities. By improving the coverage of future tokens and the utilization of candidate tokens, FlashMTP accepts more consecutive tokens per pass. FlashMTP is lossless under exact arithmetic in the sense that greedy decoding yields exactly the autoregressive output, and stochastic decoding samples every token from the model's conditional distribution rather than from the candidate set. Empirical evaluations across multiple generation benchmarks demonstrate that FlashMTP improves mean accepted tokens per decoding step (MAT) and generation throughput, with speedups of up to $3.7\times$ over vanilla autoregressive decoding. This work paves the way for efficient LLM inference through adaptive multi-token prediction. Code will be released upon acceptance.

1 INTRODUCTION

Next-Token Prediction and Scaling Laws jointly constitute the dominant paradigm of contemporary artificial intelligence. Since the introduction of the Transformer architecture (Vaswani et al., 2017), artificial intelligence has developed rapidly, giving rise to state-of-the-art large language models (LLMs) represented by ChatGPT (OpenAI, 2026), Kimi (Kimi-Team, 2026) and GLM (GLM-5-Team, 2026). LLMs involve months of pre-training, weeks of post-training, and milliseconds of inference. Unlike the one-time nature of model training, inference is executed countless times in real-world deployment. Consequently, improving the efficiency of LLM inference is of paramount practical importance.

However, the autoregressive nature of LLMs forces sequential token generation, leading to high inference latency and low generation throughput. Multi-Token Prediction (MTP), which equips the target model with additional heads or modules, has been introduced to address this problem by emitting candidate tokens for multiple future positions within a single forward pass and confirming them (Gloeckle et al., 2024; DeepSeek-AI, 2024; Cai et al., 2025). Meanwhile, Speculative Decoding has also been introduced to address the same problem. Draft-model-based speculative decoding introduces an auxiliary draft model to propose candidate tokens (Bachmann et al., 2025), which incurs higher deployment cost because an additional model has to be deployed. Draft-model-free speculative decoding dispenses with the draft model entirely and derives candidates from the model itself, for example by reusing n-grams found along its decoding trajectory or by retrieving matching text from the context guided by the logit of the last token (Fu et al., 2024; Liu et al., 2026).

Although existing methods have achieved remarkable success in accelerating LLM inference, they primarily focus on where candidate tokens come from. However, once the sources are chosen, every remaining setting is determined prior to decoding. The assignment of candidate tokens to each source and the maximum reachable depth for each source never change no matter how often the candidates each source produces are actually accepted. This imposes key limitations on these methods. First, **static sources limit the coverage of future tokens.** Token acceptance within a

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

single pass follows a chain-like dependency. A future token can be accepted only if every preceding token has been accepted, so covering the second and third future tokens is nearly decisive for overall performance, and the absence of the true token at any depth halts acceptance, regardless of how many candidates follow. Covering those tokens requires knowing what the model would predict after a candidate, which static sources can supply only when the text happens to repeat. Second, **static allocation limits the utilization of candidate tokens.** The likelihood that a candidate will be accepted differs enormously across sources, depths and levels of the model's confidence, and shifts with the model, the task and the textual style. A static policy cannot account for this, so methods following such a policy keep allocating candidate tokens to types of candidates that are rarely accepted, while rarely allocating them to those that are usually accepted. This raises a natural question:

How can we improve both the coverage of future tokens and the utilization of candidate tokens per forward pass?

To answer this question, this work focuses on achieving efficient and lossless LLM inference through a novel plug-and-play approach that adapts both factors during decoding. Unlike previous works that treat each forward pass as a one-shot verification of a static prediction, we provide a novel perspective by treating **every forward pass as a prediction of what follows each candidate it evaluates and as a measurement of how often each kind of candidate is accepted**. At every evaluated position, including those of rejected candidates, the model returns its full next-token distribution and its own next-token choice. The distribution predicts what follows the candidate at that position, which is exactly what covering the second and third future tokens requires and what static sources provide only when the text repeats. A candidate is accepted only if it equals the model's own next-token choice at the preceding position, so each pass marks every candidate as accepted or rejected. Counted separately for each source, depth and level of evidence strength, these outcomes give an estimate of how likely each kind of candidate is to be accepted, so the candidate tokens of the next pass can go to the paths on which all candidates are most likely to be accepted. Both the coverage of future tokens and the utilization of candidate tokens, the two factors that determine the accepted length, thus adapt to what the model actually accepts.

Inspired by this, we propose FlashMTP (Flash Multi-Token Prediction), a plug-and-play, training-free adaptive multi-token prediction method. FlashMTP expands the effective candidate set using the outputs at all evaluated candidate positions and dynamically allocates the candidate tokens of each pass according to path acceptance probabilities estimated from the model's own choices. By improving the coverage of future tokens and the utilization of candidate tokens, FlashMTP accepts more consecutive tokens per pass. Every committed token is the model's own choice, or under sampling a draw from the model's own distribution, so decoding is lossless under exact arithmetic.

Our contributions are:

- We propose FlashMTP, a plug-and-play, training-free and lossless multi-token prediction method that expands the effective candidate set from the outputs at all evaluated candidate positions and allocates the candidate tokens of each pass by path acceptance probabilities.
- We derive an upper bound on mean accepted tokens per decoding step (MAT), showing that the second and third future tokens govern the accepted length.
- We evaluate FlashMTP across multiple benchmarks and demonstrate its effectiveness in increasing MAT and generation throughput.

2 Related Work

In this section, we review two lines of related work, namely Multi-Token Prediction and Speculative Decoding.

Multi-Token Prediction. Multi-Token Prediction (MTP) is an emerging paradigm for accelerating LLM inference, which empowers the model to forecast multiple future tokens within a single forward pass rather than predicting only one token at each step of vanilla autoregressive decoding (Gloeckle et al., 2024; Cai et al., 2024; Ankner et al., 2024; DeepSeek-AI, 2024; Liu et al., 2025; Cai et al., 2025; Grivas et al., 2026; Draxler et al., 2026; Cui et al., 2026). These methods incorpo-

rate multi-token prediction into model training or train additional prediction modules, whereas our FlashMTP requires no additional training.

Speculative Decoding. Draft-model-based Speculative Decoding relies on an independent drafter. Such methods use a separate model to propose candidate tokens for parallel evaluation, with suitable acceptance procedures preserving the original output distribution (Leviathan et al., 2023; Chen et al., 2023). Subsequent work investigates draft training and sampling efficiency (Zafrir et al., 2025; Bachmann et al., 2025). PARD enables parallel drafting through low-cost model adaptation (An et al., 2026). SpecInfer organizes proposals into shared-prefix trees for parallel verification (Miao et al., 2024), while SpecTr formulates multi-candidate speculative sampling through optimal transport (Sun et al., 2023). Further work studies canonical decompositions and convex optimization formulations for optimal multi-draft sampling (Khisti et al., 2025; Thomas & Pal, 2026). Draft-model-free methods instead obtain candidates from retrieved text or information computed during inference. REST retrieves candidate tokens from a datastore, and CREST reduces the storage requirements of retrieval-based proposals (He et al., 2024; Ho et al., 2024). Other methods exploit candidate n-grams, top-ranked tokens from earlier verification steps or last-logit rankings (Fu et al., 2024; Luo et al., 2024; Liu et al., 2026). Different from the above work, FlashMTP makes both the candidate set and its allocation adapt during decoding. It expands the candidate set with the outputs at all evaluated positions, estimates from the model's own choices how often each kind of candidate is accepted, and allocates the candidate tokens of each pass by path acceptance probability, without an auxiliary drafter.

3 FLASHMTP

In this section, we introduce FlashMTP (Flash Multi-Token Prediction), a plug-and-play, training-free adaptive multi-token prediction method; Fig. 1 gives an overview.

3.1 PRELIMINARIES

Multi-token prediction with one forward pass. Let c denote the committed context and g_1 the next token, already known from the previous pass. A pass evaluates candidate tokens for several future positions at once. The candidates are arranged by shared prefixes, so every distinct candidate prefix u below g_1 occupies one position, attends to c, g_1 and its own ancestors only, and receives the conditional distribution $P_\theta(\cdot \mid c, g_1, u)$ over the next token together with its most likely token, which we call the model's own choice at that position. Under greedy decoding a candidate is accepted if it equals the model's own choice at its parent position and all of its ancestors were accepted; the accepted path is committed, and the model's own choice at its last position becomes the root of the next pass. We write $A \geq 1$ for the number of tokens committed by a pass, $\mathrm{MAT} = \mathbb{E}[A]$, and $q_k = \Pr[A \geq k \mid A \geq k-1]$ for $k \geq 2$. A pass evaluates the root and a set number of candidate tokens in one forward pass, so the speedup is governed by how many of these candidate tokens are accepted.

Static candidate sources and static allocation. Training-free methods obtain candidate tokens from text in the context that matches the most recent tokens, from the ranking of the model's last logit, or from a table of successors observed in earlier passes. Candidate sources and allocation policies determine which future tokens can be evaluated within each pass. FlashMTP updates successor information and uses acceptance feedback to adapt allocation across sources and depths. The challenge is therefore twofold. The candidates of a pass must cover the true future tokens, above all the second and third ones, and the candidate tokens must be allocated to candidates that will actually be accepted. FlashMTP addresses the first through the coverage of future tokens and the second through the utilization of candidate tokens, both obtained from the outputs of the forward passes themselves.

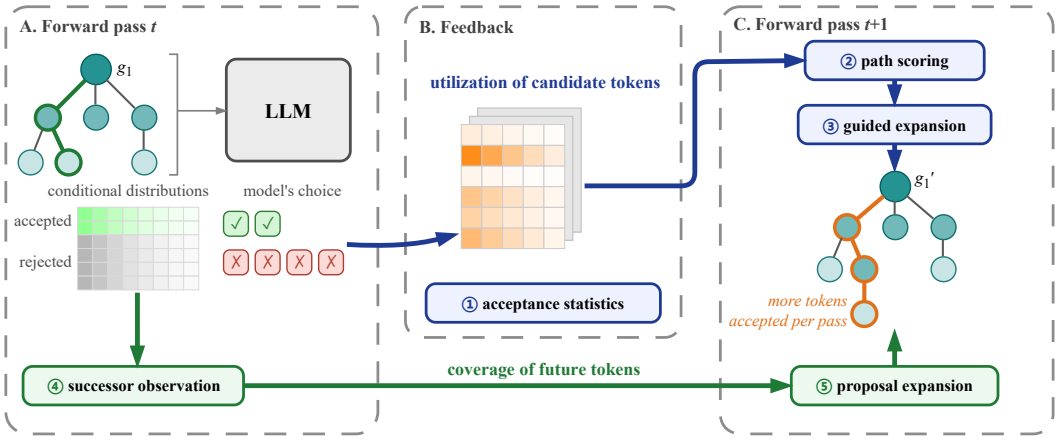


Figure 1: **Overview of FlashMTP.** (A) One forward pass over the known next token g_1 and the candidate tokens returns, at every evaluated position, the model's conditional distribution over the next token and the model's own choice, which marks each candidate as accepted or rejected; the candidates along the highlighted path are accepted, so one pass commits several consecutive tokens. ④ Successor observation stores the top tokens of every evaluated position, rejected ones included. (B) ① Acceptance statistics aggregate the accept-or-reject marks into acceptance probabilities per source, depth and evidence strength. (C) ② Path scoring values a candidate by the product of these probabilities along its path, and ③ guided expansion admits candidates in decreasing order of that value until the candidate tokens of the pass are allocated, which raises their utilization (blue). ⑤ Proposal expansion queries the stored successors and the context below every admitted candidate, which together with ④ raises the coverage of future tokens (green). The next pass, whose root g_1' is the model's own choice at the last accepted position, therefore accepts a longer path (orange).

3.2 An Upper Bound on Accepted Tokens

The expected accepted length can be expressed through the conditional acceptance rates:

$$\mathrm{MAT} = \mathbb{E}[A] = \sum_{k\geq 1} \Pr(A \geq k) = 1 + \sum_{k\geq 2} \prod_{j=2}^{k} q_j. \tag{1}$$

Here, conditional rates beyond depths with zero survival probability are set to zero.

If $q_j \leq \bar{q} < 1$ for all $j \geq 3$, the geometric-series bound gives

$$\mathrm{MAT} \leq 1 + q_2 \sum_{r=0}^{\infty} \bar{q}^r = 1 + \frac{q_2}{1-\bar{q}}. \tag{2}$$

The chain identity highlights the importance of covering the early future tokens, since q_2 appears in every term beyond the root and q_3 in every term beyond the second token. If the true token is missing at one depth, candidates below that depth cannot extend the accepted sequence. Raising MAT therefore requires candidates for the second and third future tokens that are conditioned on the candidate above them, and an allocation that directs the candidate tokens of a pass to the paths most likely to be accepted in full.

3.3 Expanding the Candidate Set

Observation. A pass evaluates every candidate position, not only those that end up accepted, and at a rejected position the model still returns its full conditional distribution, which by the argument above is exactly the prediction of the token that follows that candidate. Conventional decoding keeps only the distributions on the accepted path; FlashMTP reads all of them as candidate supply for later passes, which expands the effective candidate set and raises the coverage of future tokens.

Successor observation. After each pass, the most likely next tokens at every evaluated position, rejected positions included, are stored in an index of successors. FlashMTP stores successor observations under token and token-pair keys and uses them to propose candidates during acceptance-guided expansion. Each entry is stored under two keys, the token at the position and the pair formed with the token before it, one precise and one broad. Each stored token carries the model's confidence in it, quantised into a few bands; this confidence is never used as a score but serves as a conditioning variable for the acceptance statistics below. Newer observations take precedence, the index is bounded in size, and it persists across the requests of a session. In addition, when acceptance stops at a position, the rejected candidates there were the mutually exclusive guesses for the token the model actually chose; combining their distributions, weighted by the model's confidence in each guess, forecasts the successors of the next root even when the index has never seen it.

Proposal expansion. Candidates for the next pass are not proposed in one shot. Whenever a candidate is admitted, it becomes the parent of a new query, and all sources are asked again, with the admitted candidate as the most recent token, for the tokens that may follow it. Successors come from the index under both keys, matching text is retrieved from the context by the most recent tokens ending at the candidate, and at the root the ranking of the last logit is used as well. Deeper candidates are therefore conditioned on the candidate above them, as the chain structure requires, since a candidate for the third future token is a successor of a specific guess for the second, not of the root.

3.4 Allocating Candidate Tokens

Candidate acceptance rates vary across proposal sources, depths and evidence strengths, as well as across models and tasks. FlashMTP initializes conditional acceptance estimates with source-dependent priors and updates them using feedback from candidates whose ancestor prefixes are valid. These estimates are retained across the requests of a session. Candidate positions are then allocated according to estimated path acceptance probabilities (Fig. 2).

Acceptance statistics. The model's own choice at a position tells, for every candidate below it, whether that candidate was accepted, on rejected branches as well as on the accepted path. FlashMTP records this outcome for every candidate token whose ancestors were all accepted; a position conditioned on a rejected prefix emits a choice that autoregressive decoding would never see, so its outcomes are not counted. The outcomes are aggregated into acceptance probabilities per group of candidates, a group being defined by the source that proposed the candidate, the depth at which it stands, and the strength of the evidence behind it, namely the confidence band for tokens from the index and the width and frequency of the match for text retrieved from the context. Each group starts from a weak prior and is renormalised once it has accumulated many outcomes, so the probabilities follow the current text rather than a constant chosen in advance.

Path scoring. A candidate can be accepted only if all its ancestors are accepted. FlashMTP therefore scores a candidate prefix u by $S(u) = \prod_{e \in u} \hat{p}(e)$, where $\hat{p}(e)$ estimates the acceptance probability of edge e conditional on a valid ancestor prefix. The product is the estimated probability that the whole path is accepted and is the direct reading of the chain identity, so a deep candidate under an unlikely guess has a low value however plausible it is on its own.

Guided expansion. The candidate set of the next pass is built best-first. A priority queue holds proposed candidates ordered by path score. The highest-scoring candidate is admitted, and its proposed successors enter the queue with their own scores. This repeats until all candidate tokens of the pass are allocated or the queue is empty. Branching and depth adapt among the candidate tokens of each pass, and EOS candidates are not extended.

Expanding the candidate set raises the probability that the second and third future tokens are covered; allocating the candidate tokens by path acceptance probability ensures that the covered tokens are the ones the pass actually evaluates. Both adapt to what the model accepts, pass after pass, which is why FlashMTP accepts more consecutive tokens per pass.

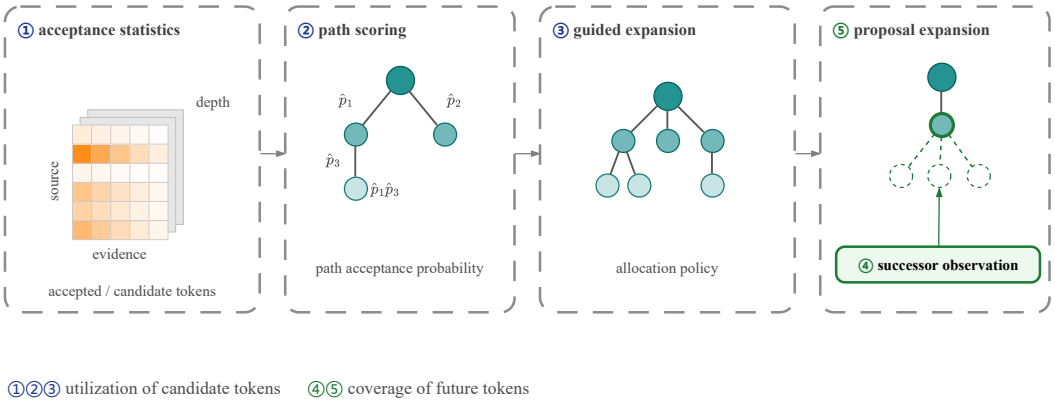


Figure 2: **Allocating the candidate tokens of one pass.** ① Acceptance statistics record, for the candidate tokens of earlier passes, whether each matched the model's own choice at its parent position, and aggregate the outcomes into acceptance probabilities per source, depth band and evidence band. ② Path scoring values a candidate by the product of the probabilities along its path, $\hat{p}_1\hat{p}_3$ for the leaf shown. ③ Guided expansion admits candidates in decreasing order of path acceptance probability until all candidate tokens of the pass are allocated, with no breadth or depth set in advance. ⑤ Proposal expansion queries ④ the successor observations of earlier passes and the matching text in the context below every admitted candidate (green), and the new candidates are scored and admitted in turn until the pass is full. The next forward pass then supplies the outcomes that update ①.

3.5 Lossless Decoding

The distribution at the last accepted position is exactly the model's conditional distribution for the new committed context, so the root of the next pass is the model's own next token and requires no candidate. Under greedy decoding every committed token equals the model's most likely token at its position, so the output is identical to sequential autoregressive decoding under exact arithmetic. For stochastic decoding the pass samples, at each depth, from the exact conditional distribution of the parent position over the full vocabulary. If the sample equals one of the candidates at that depth the pass descends to it, otherwise the sample itself is committed as the root of the next pass. Every committed token is thus a draw from the model's own conditional distribution rather than from the candidate set, the candidate set only decides how many such draws one pass realises, and no residual distribution or rejection step is needed.

4 Experiments

4.1 Experimental Setup

Models. We use Qwen2.5-Instruct (Qwen et al., 2025), DeepSeek-R1 (DeepSeek-AI, 2025) and Qwen3 (Yang et al., 2025).

Baselines. We compare FlashMTP with representative plug-and-play, training-free methods: (i) **LookAhead** (Fu et al., 2024) and (ii) **LogitSpec** (Liu et al., 2026).

Tasks. We evaluate FlashMTP on HumanEval (Chen et al., 2021) and GSM8K (Cobbe et al., 2021), which cover code generation and multi-step arithmetic reasoning, respectively, and on Spec-Bench (Xia et al., 2024), a suite of 480 questions in thirteen categories covering conversation, translation, summarization, question answering, mathematical reasoning and retrieval-augmented generation.

Metrics. We report mean accepted tokens per decoding step (**MAT**) and generation **Speedup**.

Implementation Details. All measurements use a single NVIDIA A800 80GB GPU, float16, the same SDPA attention backend, batch size one, a maximum of 1024 new tokens, and temperature 0 unless stated otherwise; the implementation is built on PyTorch (Paszke et al., 2019) and Transformers (Wolf et al., 2020).

4.2 Main Results

Table 1 reports MAT and speedup for eight model configurations on three benchmarks. We observe the following. (a) FlashMTP attains the highest MAT and the highest speedup in every cell. Its MAT is considerably higher than that of LogitSpec, and its speedup reaches $3.77\times$ on HumanEval with Qwen2.5-7B, against $2.44\times$ for LogitSpec and $1.99\times$ for LookAhead on the same task. (b) The gains carry over from 7B to 14B models and from non-reasoning to reasoning models. For the reasoning models the largest MAT is reached on GSM8K, where the chains of thought repeat phrases and structure that the successor observations cover well. (c) In every cell LookAhead and LogitSpec are much closer to each other than to FlashMTP. All three obtain their candidates without an auxiliary model, so the difference does not lie in having candidates, but in covering the second and third future tokens and in allocating the candidate tokens to paths that are accepted.

Higher MAT does not translate proportionally into speedup because evaluating multiple candidate positions and constructing candidates introduce additional overhead. Moreover, throughput is averaged over questions, whereas MAT is aggregated over decoding steps; their ratio therefore does not directly measure relative step latency.

4.3 Ablation Study

Table 2 removes one component of FlashMTP at a time on Qwen2.5-7B. Without successor observation, only the positions on the accepted path contribute to the index; without proposal expansion, the sources are no longer queried for candidates below the first future token; without acceptance statistics, the acceptance probabilities stay at their priors; without path scoring, a candidate is ranked by the probability of its last edge alone; without guided expansion, candidates are admitted layer by layer in a static order. Every ablation reduces both MAT and speedup on all three benchmarks.

The two coverage components test the first limitation stated in the introduction, that static sources limit the coverage of future tokens. Without successor observation, candidates below rejected positions are no longer available and MAT falls on every benchmark; without proposal expansion, the third future token can no longer be conditioned on the second, and the fall is larger still, in line with the chain structure of acceptance.

The three utilization components test the second limitation, that static allocation limits the utilization of candidate tokens. Freezing the acceptance statistics lowers MAT on every benchmark although the candidate sources are unchanged, so probabilities that do not follow the current text direct candidate tokens to paths that are accepted less often. Replacing path scoring or best-first admission by static rules lowers MAT more than freezing the statistics does, so how the probabilities are used along whole paths matters more than the precision of any single probability. Taken together, the ablations answer the question posed in the introduction. Neither better coverage nor better allocation alone recovers the full result; both are needed, and both are obtained by reading each forward pass twice, as a prediction of what follows each candidate and as a measurement of how often each kind of candidate is accepted.

4.4 Further Analysis

Running time. Every method performs exactly one forward pass per decoding step, so MAT is a property of the algorithm alone, and the speedup additionally reflects the time per step. A FlashMTP step evaluates the root and all candidate tokens of the pass at once and therefore takes longer than an autoregressive step over a single position, and building the candidate set of the next pass on the host adds a small fraction of the step on top of the forward pass. The speedup is approximately the MAT divided by this step-time ratio, which is why it stays below MAT; the larger models, whose forward pass over the full set of positions is relatively more expensive, generally keep a smaller share of their MAT as speedup.

Table 1: Experimental results of FlashMTP on HumanEval, GSM8K and Spec-Bench, in terms of the mean accepted tokens per decoding step (MAT) and the overall speedup ratio.

Model	Method	HumanEval		GSM8K		Spec-Bench	
		MAT	Speedup	MAT	Speedup	MAT	Speedup
Qwen2.5-7B	AutoRegressive	1.00	1.00×	1.00	1.00×	1.00	1.00×
	LookAhead	2.22	1.99×	2.02	1.67×	1.78	1.46×
	LogitSpec	2.77	2.44×	2.17	1.70×	1.93	1.54×
	FlashMTP	**4.71**	**3.77×**	**3.55**	**2.40×**	**3.15**	**2.14×**
Qwen2.5-14B	AutoRegressive	1.00	1.00×	1.00	1.00×	1.00	1.00×
	LookAhead	1.82	1.55×	2.02	1.57×	1.71	1.39×
	LogitSpec	2.01	1.67×	2.17	1.60×	1.86	1.49×
	FlashMTP	**3.32**	**2.53×**	**3.35**	**2.24×**	**2.98**	**2.13×**
DeepSeek-7B	AutoRegressive	1.00	1.00×	1.00	1.00×	1.00	1.00×
	LookAhead	1.84	1.61×	2.08	1.72×	1.78	1.65×
	LogitSpec	1.92	1.63×	2.26	1.78×	1.81	1.68×
	FlashMTP	**3.19**	**2.41×**	**3.85**	**2.73×**	**3.03**	**2.49×**
DeepSeek-14B	AutoRegressive	1.00	1.00×	1.00	1.00×	1.00	1.00×
	LookAhead	1.81	1.49×	1.98	1.60×	1.74	1.50×
	LogitSpec	1.94	1.53×	2.09	1.61×	1.79	1.49×
	FlashMTP	**3.16**	**2.32×**	**3.58**	**2.56×**	**3.00**	**2.33×**
Qwen3-8B (non-reason)	AutoRegressive	1.00	1.00×	1.00	1.00×	1.00	1.00×
	LookAhead	1.85	1.59×	1.96	1.79×	1.74	1.62×
	LogitSpec	2.15	1.98×	2.15	1.91×	1.88	1.74×
	FlashMTP	**3.75**	**3.10×**	**3.48**	**2.79×**	**3.08**	**2.57×**
Qwen3-14B (non-reason)	AutoRegressive	1.00	1.00×	1.00	1.00×	1.00	1.00×
	LookAhead	2.12	1.84×	2.03	1.58×	1.70	1.39×
	LogitSpec	2.53	2.27×	2.15	1.61×	1.82	1.45×
	FlashMTP	**4.51**	**3.71×**	**3.51**	**2.38×**	**2.99**	**2.18×**
Qwen3-8B (reason)	AutoRegressive	1.00	1.00×	1.00	1.00×	1.00	1.00×
	LookAhead	1.88	1.73×	1.99	1.81×	1.84	1.70×
	LogitSpec	1.96	1.70×	2.11	1.74×	1.91	1.75×
	FlashMTP	**3.22**	**2.51×**	**3.48**	**2.67×**	**3.24**	**2.70×**
Qwen3-14B (reason)	AutoRegressive	1.00	1.00×	1.00	1.00×	1.00	1.00×
	LookAhead	1.84	1.48×	1.95	1.52×	1.78	1.49×
	LogitSpec	1.94	1.51×	2.03	1.52×	1.84	1.48×
	FlashMTP	**3.17**	**2.31×**	**3.43**	**2.40×**	**3.14**	**2.38×**

Table 2: Ablation study of FlashMTP on HumanEval, GSM8K and Spec-Bench with Qwen2.5-7B, in terms of MAT and speedup ratio.

Method	HumanEval		GSM8K		Spec-Bench	
	MAT	Speedup	MAT	Speedup	MAT	Speedup
w/o acceptance statistics	3.78	2.97×	2.96	2.03×	2.73	1.89×
w/o path scoring	3.53	2.89×	2.36	1.65×	2.23	1.57×
w/o guided expansion	2.50	1.94×	2.43	1.68×	2.28	1.63×
w/o successor observation	3.95	3.28×	2.77	1.95×	2.38	1.70×
w/o proposal expansion	3.24	2.86×	2.21	1.70×	2.17	1.67×
FlashMTP	**4.71**	**3.77×**	**3.55**	**2.40×**	**3.15**	**2.14×**

Sampling temperature. The left panel of Fig. 3 shows MAT on Spec-Bench at temperatures 0, 0.6 and 1.0. MAT varies moderately across the tested temperatures, and FlashMTP at temperature 1.0 remains above both baselines at temperature 0. Under sampling, almost every pass ends when the sampled token falls outside the candidate set and becomes the next root. Under exact arithmetic, the sampling procedure preserves the autoregressive output distribution. This does not require individual sampled texts or their lengths to match, and finite-precision differences remain possible.

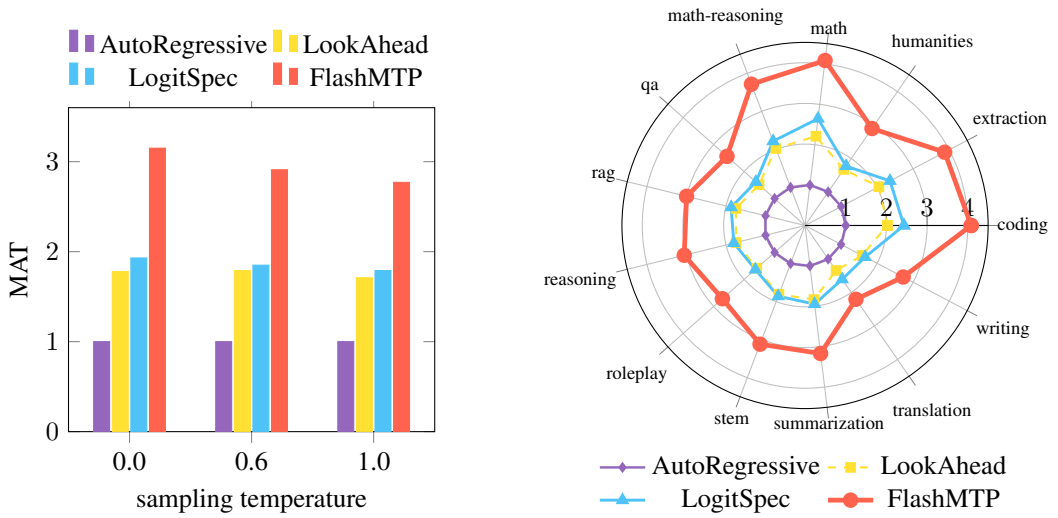


Figure 3: Analysis on Qwen2.5-7B. Left: MAT on Spec-Bench at sampling temperatures 0, 0.6 and 1.0. Right: MAT per Spec-Bench category at temperature 0, computed as the total number of accepted tokens divided by the total number of decoding steps within each category.

Per-category behaviour. The right panel shows MAT per Spec-Bench category. FlashMTP leads in all thirteen categories. Its MAT is highest on coding, math and extraction, where the generated text repeats its own structure, and lowest on translation and question answering.

Precision. In the reported precision experiment, float32 outputs match the autoregressive outputs exactly, whereas bfloat16 and float16 outputs are not fully identical despite aggregate length ratios close to one. These observations concern numerical agreement in finite precision and are distinct from algorithmic losslessness under exact arithmetic. Detailed results are reported in Table 4.

5 Conclusion

We propose FlashMTP, a plug-and-play, training-free adaptive multi-token prediction method that treats every forward pass as a prediction of what follows each evaluated candidate and as a measurement of how often each kind of candidate is accepted. By improving the coverage of future tokens and the utilization of candidate tokens, FlashMTP enables more consecutive tokens to be accepted per forward pass. Experiments demonstrate its effectiveness in accelerating LLM generation, and ablation studies support the contributions of its key components. FlashMTP thus offers a promising route toward efficient LLM inference.

Future Work. We consider four directions for future work. (i) Enhanced support for reasoning models. As model development shifts from pretraining on next-token prediction toward month-scale, long-horizon agentic reinforcement learning post-training, long reasoning traces greatly increase the computational load and latency of deployment. Future work will evaluate FlashMTP on longer reasoning traces and analyze how their repeated structure affects the coverage of future tokens. (ii) Mixture-of-Experts (MoE) support. Future work will extend FlashMTP to large MoE models and examine how candidate allocation interacts with expert routing and load balancing. For distributed deployments, communication overhead will also be considered when choosing the number of candidate tokens. (iii) Multimodal support. Future work will explore FlashMTP for autoregressive multimodal generation. This requires adapting candidate construction and decoding-state management to different modalities, and evaluating how their token patterns affect the coverage and acceptance of future tokens. (iv) Improving the coverage of future tokens. Future work will investigate low-overhead proposals for future tokens that existing sources cover poorly. The objective is to improve the coverage of both early and deeper future tokens while keeping the additional computation small enough to yield end-to-end acceleration.

AI USE STATEMENT

In this work, we used generative AI tools to assist with and polish writing, and to support literature retrieval and discovery of related work. We have reviewed all AI-assisted work, including checking the accuracy of revised text and verifying the existence and relevance of suggested references against the original sources. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This paper adheres to the Code of Ethics. All datasets and model weights are publicly available. This work proposes a plug-and-play, training-free, lossless acceleration method for efficient LLM inference, and does not involve any sensitive or personally identifiable information.

REPRODUCIBILITY STATEMENT

We provide comprehensive implementation details to ensure, to the best of our ability, the reproducibility of our experimental results. The datasets, models, software stack (PyTorch, Transformers), and hardware (NVIDIA A800 80GB and NVIDIA RTX 3090 24GB) are all listed in the paper. The configuration of FlashMTP and the evaluation protocol are given in the Implementation Details Appendix A, the five ablated components are specified as pseudocode in the Algorithms Appendix B, and the losslessness argument is given in the Losslessness Appendix C. Code will be released upon acceptance.

REFERENCES

Zihao An, Huajun Bai, Ziqiong Liu, Dong Li, and Emad Barsoum. PARD: Accelerating LLM inference with low-cost PARallel draft model adaptation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=XbOyv7iVGL.

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for medusa decoding, 2024. URL https://arxiv.org/abs/2402.05109.

Gregor Bachmann, Sotiris Anagnostidis, Albert Pumarola, Markos Georgopoulos, Artsiom Sanakoyeu, Yuming Du, Edgar Schönfeld, Ali Thabet, and Jonas K Kohler. Judge decoding: Faster speculative sampling requires going beyond model alignment. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mtSSFiqW6y.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=PEpbUobfJv.

Yuxuan Cai, Xiaozhuan Liang, Xinghua Wang, Jin Ma, Haijin Liang, Jinwen Luo, Xinyu Zuo, Lisheng Duan, Yuyang Yin, and Xi Chen. Fastmtp: Accelerating llm inference with enhanced multi-token prediction, 2025. URL https://arxiv.org/abs/2509.18362.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling, 2023. URL https://arxiv.org/abs/2302.01318.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex

Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Ziqiang Cui, Han Shi, Bowei He, Yu Pan, Peiyang Liu, Shengyin Sun, Yankai Chen, Haoli Bai, Yichun Yin, Xue Liu, and Chen Ma. Adamtp: An adaptive training paradigm for multi-token prediction, 2026. URL https://arxiv.org/abs/2608.00434.

DeepSeek-AI. Deepseek-v3 technical report, 2024. URL https://arxiv.org/abs/2412.19437.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Felix Draxler, Justus Will, Farrin Marouf Sofian, Theofanis Karaletsos, Sameer Singh, and Stephan Mandt. Parallel token prediction for language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=AGJomYSrUG.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of LLM inference using lookahead decoding. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 14060–14079. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/fu24a.html.

GLM-5-Team. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Roziere, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 15706–15734. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/gloeckle24a.html.

Andreas Grivas, Lorenzo Loconte, Emile van Krieken, Piotr Nawrot, Yu Zhao, Euan Wielewski, Pasquale Minervini, Edoardo Ponti, and Antonio Vergari. Fast and expressive multi-byte prediction with probabilistic circuits. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=6kCEyw9god.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason D. Lee, and Di He. Rest: Retrieval-based speculative decoding, 2024. URL https://arxiv.org/abs/2311.08252.

Sophia Ho, Jinsol Park, and Patrick Wang. Crest: Effectively compacting a datastore for retrieval-based speculative decoding, 2024. URL https://arxiv.org/abs/2408.04678.

Ashish J Khisti, MohammadReza Ebrahimi, Hassan Dbouk, Arash Behboodi, Roland Memisevic, and Christos Louizos. Multi-draft speculative sampling: Canonical decomposition and theoretical limits. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=N1L5TgtkAw.

Kimi-Team. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Tianyu Liu, Qitan Lv, Hao Li, Xing Gao, Xiao Sun, and Xiaoyan Sun. Logitspec: Accelerating retrieval-based speculative decoding via next next token speculation, 2026. URL https://arxiv.org/abs/2507.01449.

Xiaohao Liu, Xiaobo Xia, Weixiang Zhao, Manyi Zhang, Xianzhi Yu, Xiu Su, Shuo Yang, See-Kiong Ng, and Tat-Seng Chua. L-MTP: Leap multi-token prediction beyond adjacent context for large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=0VDmWjW456.

Xianzhen Luo, Yixuan Wang, Qingfu Zhu, Zhiming Zhang, Xuanyu Zhang, Qing Yang, and Dongliang Xu. Turning trash into treasure: Accelerating inference of large language models with token recycling. *arXiv preprint arXiv:2408.08696*, 2024.

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. 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*, ASPLOS '24, pp. 932–949. ACM, April 2024. doi: 10.1145/3620666.3651335. URL http://dx.doi.org/10.1145/3620666.3651335.

OpenAI. Openai gpt-5 system card, 2026. URL https://arxiv.org/abs/2601.03267.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. Spectr: Fast speculative decoding via optimal transport. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=SdYHLTCC5J.

Rahul Krishna Thomas and Arka Pal. Global resolution: Optimal multi-draft speculative sampling via convex optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=gpsczXOsHn.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020. URL https://arxiv.org/abs/1910.03771.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics ACL 2024*, pp. 7655–7671, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.456. URL https://aclanthology.org/2024.findings-acl.456.

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, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Ofir Zafrir, Igor Margulis, Dorin Shteyman, Shira Guskin, and Guy Boudoukh. Fastdraft: How to train your draft, 2025. URL https://arxiv.org/abs/2411.11055.

APPENDIX OVERVIEW

This appendix is organized as follows. Appendix A lists the configuration of FlashMTP used in every experiment and the evaluation protocol, including how MAT, throughput and speedup are computed. Appendix B gives pseudocode for the five components ablated in the main text, namely acceptance statistics, path scoring, guided expansion, successor observation and proposal expansion. Appendix C states the losslessness argument for greedy and stochastic decoding and reports the precision experiment. Appendix D compares all methods on an NVIDIA RTX 3090 and an NVIDIA A800 with Qwen2.5-7B. Appendix E lists every mechanism of FlashMTP, divided into the allocation group and the supply group. Appendix F provides additional results, namely per-category MAT on Spec-Bench for all model configurations, the accepted-length distribution with the conditional acceptance rates behind the chain identity, results at different sampling temperatures on all three benchmarks, and source-level acceptance probabilities.

A IMPLEMENTATION DETAILS

A.1 FLASHMTP CONFIGURATION

Table 3 lists the settings used in every experiment of this paper; they are identical across models, benchmarks and hardware, and none of them was tuned per benchmark.

Table 3: FlashMTP settings used in experiments.

Setting	**Value**
Successors stored per evaluated position	top-8 tokens with confidence bands
Successor index	8,192 keys
Context match	most recent 3 tokens
Last-logit candidates	top-32 tokens, root only
Acceptance statistics	sources $\times$ depth bands $\times$ evidence bands
Joint sibling estimates	root and up to 4 depth-one nodes
Forecast from rejected siblings	up to 8 tokens, root only
Decoding	at most 1,024 new tokens
Precision	float16

A.2 EVALUATION PROTOCOL

Spec-Bench has 480 questions in thirteen categories, several with two conversation turns; HumanEval has 164 and GSM8K has 300 questions; every method generates at most 1,024 new tokens per turn with the chat template of the model. MAT is the total number of committed tokens divided by the total number of decoding steps over all turns of a benchmark, where one decoding step is one forward pass of the model; throughput is the number of generated tokens divided by the wall-clock generation time of a question, averaged over questions. For the main comparisons, speedup is the ratio of a method's mean throughput to the corresponding autoregressive baseline.

B ALGORITHMS

This appendix gives algorithms for the five components ablated in the main text, in the order of the rows of the ablation table. Acceptance statistics (Algorithm 1), path scoring (Algorithm 2) and guided expansion (Algorithm 3) allocate the candidate tokens, and successor observation (Algorithm 4) and proposal expansion (Algorithm 5) expand the candidate set. Notation: c is the committed context and g_1 the known next token; a candidate prefix u occupies one evaluated position (row) r with parent row $\pi(r)$; a_r is the model's own choice at row r and $y_{r,1..k}$ its top-k next tokens with confidence bands $\beta_{r,1..k}$; a bucket $b = (\sigma, \min(d, 4), \epsilon)$ combines source, depth band and evidence band; $H[b]$ and $T[b]$ count accepted and evaluated candidate tokens of a bucket, with acceptance

probability $\hat{p}_b = H[b]/T[b]$; $B = 63$ is the number of candidate tokens per pass and $\mathcal{E}$ the set of end-of-sequence tokens. One decoding step first runs guided expansion, which calls proposal expansion and path scoring, to build the candidate set of the pass; it then performs one forward pass over the resulting positions and runs successor observation and acceptance statistics on the outputs of that pass.

Algorithm 1 Acceptance statistics (after a forward pass)

Require: rows $r = 1, \ldots, N-1$ with last token x_r, parent $\pi(r)$ and bucket $b(r)$ recorded when the row was admitted; the model's own choice a_r at every row; tables H, T

$\text{valid}_0 \leftarrow \text{true}$
for $r = 1$ **to** $N-1$ **do** $\text{valid}_r \leftarrow \text{valid}_{\pi(r)} \wedge [\, x_r = a_{\pi(r)} \,]$
end for ▷ a row is valid if every ancestor was accepted
for $r = 1$ **to** $N-1$ **do** ▷ one outcome per candidate token, rejected branches included
  if $\text{valid}_{\pi(r)}$ **then**
    $H[b(r)] \mathrel{+}= [\, x_r = a_{\pi(r)} \,]; \quad T[b(r)] \mathrel{+}= 1$
    if $T[b(r)] > T_{\max}$ **then** $H[b(r)] \leftarrow H[b(r)]/2$; $T[b(r)] \leftarrow T[b(r)]/2$
    end if
  end if
end for

Ensure: $\hat{p}_b = H[b]/T[b]$ for every bucket b, measured on the current generation

Algorithm 2 Path scoring

Require: admitted prefix u with path value $S(u)$ ($S(()) = 1$); candidate x with bucket b_x; tables H, T

$S(u + (x)) \leftarrow S(u) \cdot \hat{p}(x \mid u)$

Ensure: $S(u + (x)) = \prod_{e \in u + (x)} \hat{p}_{b(e)}$, the estimated probability that every candidate on the path is accepted

Algorithm 3 Guided expansion (building the candidate set of the next pass)

Require: root g_1; index $\mathcal{I}$, context c, $\mathcal{L}$, $\mathcal{M}$; tables H, T; limit B; end-of-sequence set $\mathcal{E}$

$\mathcal{A} \leftarrow \{()\}$; priority queue $\mathcal{Q} \leftarrow \emptyset$; EXPAND(), 1)
while $\mathcal{Q} \neq \emptyset$ **and** $|\mathcal{A}| < B + 1$ **do**
  pop the prefix u of maximal $S(u)$ from $\mathcal{Q}$; **if** $u \in \mathcal{A}$ **continue**
  $\mathcal{A} \leftarrow \mathcal{A} \cup \{u\}$; record $b(u)$ for acceptance statistics; EXPAND(u, $S(u)$)
end while
return the prefixes in $\mathcal{A}$, each realised once as an evaluated position attending to c, g_1 and its own ancestors

procedure EXPAND(u, S)
  if $u \neq ()$ **and** the last token of u is in $\mathcal{E}$ **then return**
  end if ▷ a finished sequence is not extended
  for each $(x, \sigma_x, \epsilon_x)$ returned by proposal expansion for u **do**
    $S' \leftarrow S \cdot \hat{p}(x \mid u)$ ▷ path scoring
    if $S' > 0$ **and** $u + (x) \notin \mathcal{A}$ **then** push $u + (x)$ with key S' onto $\mathcal{Q}$
    end if
  end for
end procedure

Ensure: At most B candidate tokens selected by best-first expansion

Algorithm 4 Successor observation (after a forward pass)

Require: rows $r = 0, \dots, N-1$ with prefixes u_r ($u_0 = ()$), top-k tokens $y_{r,\cdot}$ and bands $\beta_{r,\cdot}$ of every row, accepted or not; index $\mathcal{I}$ with capacity M

for $r = 0$ **to** $N-1$ **do**
  $(g, p) \leftarrow$ the last two tokens of (c, g_1, u_r) ▷ grandparent and parent of the position that follows u_r
  INSERT($\mathcal{I}, (p),\ y_{r,\cdot}, \beta_{r,\cdot}$); INSERT($\mathcal{I}, (g, p),\ y_{r,\cdot}, \beta_{r,\cdot}$)
end for

procedure INSERT($\mathcal{I}, \kappa, y, \beta$)
  if $\kappa \in \mathcal{I}$ **then** $(y, \beta) \leftarrow$ the new tokens followed by the previously stored tokens not among them, truncated to k
  end if
  $\mathcal{I}[\kappa] \leftarrow (y, \beta)$ and mark κ as most recently used
  if $|\mathcal{I}| > M$ **then** evict the least recently used key
  end if
end procedure

Ensure: for every prefix evaluated so far, $\mathcal{I}$ holds the model's top-k successors under a one-token and a two-token key

Algorithm 5 Proposal expansion (candidates below an admitted prefix)

Require: prefix u with parent p and grandparent g; index $\mathcal{I}$; context c; last-logit tokens $\mathcal{L}$ and marginal forecast $\mathcal{M}$ (both used only when $u = ()$); propagated run τ_u of the matched text that led to u

$\mathcal{C} \leftarrow ()$ ▷ ordered list; a token already in $\mathcal{C}$ is not added again
for $(i, y) \in \mathcal{I}[(g, p)]$ **do** append $(y,\ \text{two-key index},\ \beta_y)$ to $\mathcal{C}$
end for
$(z, \omega, \nu) \leftarrow$ the most frequent match of the last n tokens of (c, g_1, u) in the context, with match width ω and count ν
if $z \neq ()$ **then** append $(z_1,\ \text{context match},\ \mathrm{band}(\omega, \nu))$ to $\mathcal{C}$ and attach $\tau_{u+(z_1)} \leftarrow z_{2..}$
end if
for $(i, y) \in \mathcal{I}[(p)]$ **do** append $(y,\ \text{one-key index},\ \beta_y)$ to $\mathcal{C}$
end for
if $u = ()$ **then**
  for $(i, y) \in \mathcal{L}$ **do** append $(y,\ \text{last logit},\ \mathrm{rankband}(i))$ to $\mathcal{C}$
  end for
  for $(y, \beta) \in \mathcal{M}$ **do** append $(y,\ \text{marginal},\ \beta)$ to $\mathcal{C}$
  end for
end if
if $\tau_u \neq ()$ **then** append $(\tau_{u,1},\ \text{propagated run},\ \mathrm{runband}(|\tau_u|))$ to $\mathcal{C}$
end if

Ensure: candidates $(x, \sigma_x, \epsilon_x) \in \mathcal{C}$ for the children of u, each weighted by the bucket $b_x = (\sigma_x, \min(|u| + 1, 4), \epsilon_x)$; because the query is issued with u itself as parent, every child is conditioned on the candidate it hangs from

C LOSSLESSNESS

Greedy decoding. Every evaluated position attends to the committed context, the known next token and its own ancestors only, so its output distribution equals the distribution that sequential autoregressive decoding computes at the same prefix. A candidate is committed only if it equals the model's own choice at its parent position and all of its ancestors were committed, and the root of the next pass is the model's own choice at the last accepted position. By induction over the committed tokens, the output of FlashMTP under greedy decoding equals the autoregressive output token for token, under exact arithmetic.

Stochastic decoding. With temperature $T > 0$, FlashMTP samples the root of a pass from the softmax of the last accepted position's logits at temperature T over the full vocabulary, and within a pass it samples at each depth from the full-vocabulary distribution of the parent position. If the sample equals one of the candidates at that depth, the pass descends to that candidate; otherwise the sample itself is committed as the root of the next pass. Every committed token is therefore a draw from the model's own conditional distribution at its own prefix. In the language of speculative sampling, the proposal distribution equals the target distribution, the acceptance ratio is identically one, and no residual distribution is needed; the candidate set only determines how many such draws one pass realises. The sequence of committed tokens has the same distribution as autoregressive sampling; individual texts differ, as two runs of autoregressive sampling with different random draws differ.

Precision. The losslessness arguments above assume exact arithmetic. In finite precision, evaluating multiple positions together can change numerical results relative to single-token decoding because operation shapes and reduction orders differ. Such differences can alter token selection and the tokens that follow. Table 4 reports greedy decoding on HumanEval with Qwen2.5-7B on the A800. Float32 outputs match the autoregressive outputs exactly in this experiment, whereas bfloat16 and float16 outputs are not fully identical. Their aggregate length ratios are close to one, but this does not establish token-level agreement or task-quality equivalence. The measured speedups vary across precisions.

Table 4: FlashMTP performance across different precisions. Algorithmic losslessness refers to the decoding procedure under exact arithmetic. Exact match indicates whether all evaluated outputs match autoregressive decoding.

prec	algorithmic losslessness	text_exactmatch_autoregressive	length_ratio	speedup
f32	✓	✓	1.0000	1.36×
bf16	✓	almost	0.9950	3.91×
f16	✓	almost	0.9992	3.77×

D Cross-Hardware Evaluation

Table 5 shows end-to-end performance across hardware platforms on Qwen2.5-7B-Instruct. Meanwhile, the absolute speed of each method on the NVIDIA A800 is higher than that of the corresponding method on the NVIDIA RTX 3090.

Table 5: Performance across hardware platforms

Hardware	Method	HumanEval		GSM8K		Spec-Bench	
		MAT	Speedup	MAT	Speedup	MAT	Speedup
RTX3090	AutoRegressive	1.00	1.00×	1.00	1.00×	1.00	1.00×
	LookAhead	2.22	1.79×	2.02	1.49×	1.78	1.32×
	LogitSpec	2.77	2.17×	2.18	1.51×	1.93	1.38×
	FlashMTP	**4.71**	**3.52×**	**3.56**	**2.26×**	**3.14**	**2.05×**
A800	AutoRegressive	1.00	1.00×	1.00	1.00×	1.00	1.00×
	LookAhead	2.22	1.99×	2.02	1.67×	1.78	1.46×
	LogitSpec	2.77	2.44×	2.17	1.70×	1.93	1.54×
	FlashMTP	**4.71**	**3.77×**	**3.55**	**2.40×**	**3.15**	**2.14×**

E Mechanism Catalogue

The mechanisms of FlashMTP fall into two groups that correspond to the two parts of the method. The allocation group decides which candidates receive the candidate tokens of a pass: the model's

own choice at every evaluated position becomes acceptance probabilities per source, depth and evidence strength, candidates are valued by the product of these probabilities along their path, and candidate tokens are admitted best-first by that value. The supply group decides which candidates exist at all: the distributions returned at all evaluated positions are stored with the model's confidence in an index of successors and queried again below every admitted candidate, together with matching text from the context, the ranking of the last logit, and a forecast of the next root from the rejected siblings at the last accepted position. Table 6 lists the allocation group and Table 7 the supply group.

Table 6: Mechanisms of the allocation group.

Mechanism	**Principle**
Conditional acceptance statistics	acceptance probability per source, depth band and evidence band, from the model's choices
Path acceptance scoring	a candidate is valued by the product of the acceptance probabilities along its path
Acceptance-guided expansion	best-first admission by path value until the candidate tokens are allocated
Evidence-strength conditioning	the third axis of the statistics is a source-specific evidence band, not a rank
Regularized acceptance estimation	weak priors per group, halved once a group has accumulated many outcomes
Prefix-validity conditioning	an outcome is recorded only when all ancestors of the candidate were accepted
Sibling acceptance normalization	mutually exclusive siblings are normalised to their estimated coverage
Candidate-set coverage estimation	how often the true token is among the proposed siblings, per source combination
Root sibling acceptance estimation	the joint sibling estimate applied to the candidates below the root
Depth-one sibling acceptance estimation	the joint sibling estimate applied below the best depth-one candidates
Depth-specific acceptance estimation	separate sibling statistics for the root and for depth one
Unselected-candidate supervision	siblings proposed but not admitted are still marked accepted or not
Segment-length conditioning	the position inside a matched segment is the evidence band of that source
Shared-prefix realisation	each distinct prefix occupies one evaluated position attending to its ancestors

Table 7: Mechanisms of the supply group.

Mechanism	Principle
Verifier successor observation	the top next tokens of every evaluated position, rejected ones included, enter the index
Conditional proposal expansion	all sources are queried again below every admitted candidate
Contextual successor index	a bounded table from a context key to the successors observed after it
Second-order successor conditioning	successors are stored under a one-token key and a two-token key
Recency-based successor retention	newer successors are merged first; least recently used keys are evicted
Persistent prediction state	index and statistics persist across the requests of a session
Successor mass observation	each stored successor carries the model's confidence in it, in five bands
Sibling marginal forecast	rejected siblings at the last accepted position forecast the next root's successors
Context match retrieval	the text that followed the longest match of the recent tokens in the context
Match frequency scoring	among repeated matches, the successor observed most often is preferred
Matched-segment propagation	the remainder of a matched segment is carried below the admitted candidate
Logit-rank proposal support	the highest-ranked tokens of the last logit are offered at the root
Multi-source evidence aggregation	a token from several sources is kept once, with all its sources and the highest probability
Segment-preserving aggregation	a matched segment survives merging even if another source gives the higher probability

F Additional Results

F.1 Per-category results on Spec-Bench

Tables 8 and 9 break the Spec-Bench MAT of every model configuration down into the thirteen categories, computed from the per-question records of the A800 runs. FlashMTP has the highest MAT in every category of every configuration. The ordering of the categories is similar for the three methods. Math, math reasoning and extraction, where the generated text repeats its own structure, are among the easiest and roleplay among the hardest; translation is the hardest category for the non-reasoning models but not for the DeepSeek-R1-Distill and reasoning-mode Qwen3 models, whose chain-of-thought output makes it as easy as the other categories.

F.2 Accepted-length distribution and the chain identity

For Qwen2.5-7B on Spec-Bench, LookAhead never commits more than four tokens per step by construction of its n-gram size and commits a single token in about two thirds of its steps, LogitSpec commits one or two tokens in about 85 percent of its steps, whereas FlashMTP commits a single token in fewer than a fifth of its steps and four or more tokens in nearly a third. Table 10 reports the conditional acceptance rates $q_k = \Pr[A \geq k \mid A \geq k-1]$ estimated from the same records, together with the partial sum $1 + q_2 + q_2 q_3 + \cdots + q_2 \cdots q_8$, which by the chain identity equals $\mathbb{E}[\min(A, 8)]$. Its difference from MAT measures the contribution beyond eight accepted tokens. This contribution is largest on HumanEval but is also nonzero on GSM8K and Spec-Bench for LogitSpec and FlashMTP. FlashMTP has higher q_2 than both baselines on all three benchmarks and higher q_3 than LogitSpec. Conditional rates at deeper positions vary across methods; LookAhead does not extend beyond four accepted tokens in this configuration.

F.3 Sampling temperature

Table 11 extends the temperature analysis of the main text to all three benchmarks.

Table 8: Per-category MAT on Spec-Bench for the Qwen2.5-Instruct and DeepSeek-R1 models (LA: LookAhead, LS: LogitSpec, FMTP: FlashMTP).

Category	**Qwen2.5-7B**			**Qwen2.5-14B**			**DeepSeek-7B**			**DeepSeek-14B**		
	LA	LS	FMTP	LA	LS	FMTP	LA	LS	FMTP	LA	LS	FMTP
writing	1.58	1.67	**2.72**	1.53	1.63	**2.66**	1.66	1.66	**2.71**	1.65	1.69	**2.77**
roleplay	1.58	1.63	**2.71**	1.51	1.55	**2.51**	1.59	1.56	**2.58**	1.57	1.56	**2.60**
reasoning	1.74	1.80	**3.06**	1.73	1.86	**3.01**	1.94	1.94	**3.26**	1.84	1.91	**3.21**
math	2.21	2.64	**4.08**	2.13	2.49	**3.78**	1.98	2.14	**3.54**	1.99	2.18	**3.51**
coding	2.02	2.43	**4.08**	1.90	2.10	**3.49**	1.80	1.92	**3.23**	1.81	1.88	**3.13**
extraction	2.05	2.36	**3.88**	2.12	2.62	**3.93**	1.87	2.09	**3.33**	1.77	2.00	**3.23**
stem	1.81	1.86	**3.12**	1.67	1.73	**2.87**	1.65	1.63	**2.72**	1.66	1.65	**2.77**
humanities	1.67	1.77	**2.90**	1.61	1.68	**2.77**	1.60	1.53	**2.53**	1.56	1.58	**2.60**
translation	1.34	1.61	**2.20**	1.27	1.60	**2.10**	1.86	2.05	**3.50**	1.86	2.04	**3.52**
summarization	1.83	1.95	**3.17**	1.74	1.86	**3.02**	1.73	1.74	**2.93**	1.65	1.66	**2.78**
qa	1.51	1.61	**2.56**	1.45	1.59	**2.55**	1.69	1.67	**2.81**	1.65	1.65	**2.79**
math-reasoning	2.02	2.22	**3.71**	1.98	2.22	**3.53**	1.99	2.09	**3.52**	1.98	2.14	**3.59**
rag	1.75	1.87	**3.00**	1.67	1.78	**2.82**	1.82	1.85	**3.07**	1.76	1.81	**3.03**
all	1.78	1.93	**3.15**	1.71	1.86	**2.98**	1.78	1.81	**3.03**	1.74	1.79	**3.00**

Table 9: Per-category MAT on Spec-Bench for the Qwen3 models in non-reasoning and reasoning mode (LA: LookAhead, LS: LogitSpec, FMTP: FlashMTP).

Category	**Qwen3-8B (non-reason)**			**Qwen3-14B (non-reason)**			**Qwen3-8B (reason)**			**Qwen3-14B (reason)**		
	LA	LS	FMTP	LA	LS	FMTP	LA	LS	FMTP	LA	LS	FMTP
writing	1.65	1.61	**2.79**	1.53	1.65	**2.72**	1.68	1.75	**3.03**	1.64	1.68	**2.87**
roleplay	1.57	1.59	**2.64**	1.55	1.56	**2.60**	1.64	1.61	**2.77**	1.60	1.59	**2.64**
reasoning	1.85	2.21	**3.56**	1.78	1.89	**3.08**	2.01	2.12	**3.48**	1.85	1.90	**3.15**
math	2.14	2.44	**3.82**	2.16	2.45	**3.98**	1.95	2.14	**3.43**	1.91	2.07	**3.37**
coding	1.85	2.13	**3.40**	1.82	2.02	**3.34**	1.87	1.98	**3.17**	1.87	1.89	**3.18**
extraction	2.07	2.32	**3.77**	2.03	2.40	**3.92**	1.99	2.21	**3.59**	1.89	2.03	**3.36**
stem	1.68	1.76	**2.87**	1.70	1.79	**2.92**	1.76	1.79	**3.00**	1.70	1.74	**2.90**
humanities	1.63	1.69	**2.78**	1.59	1.65	**2.72**	1.63	1.58	**2.72**	1.57	1.56	**2.65**
translation	1.35	1.62	**2.39**	1.35	1.63	**2.34**	1.94	2.06	**3.59**	1.89	2.03	**3.51**
summarization	1.74	1.83	**3.08**	1.69	1.79	**2.96**	1.69	1.70	**2.99**	1.66	1.67	**2.96**
qa	1.64	1.79	**2.93**	1.57	1.71	**2.83**	1.80	1.83	**3.16**	1.73	1.74	**3.03**
math-reasoning	1.93	2.12	**3.52**	1.95	2.10	**3.51**	1.97	2.12	**3.52**	1.91	2.03	**3.43**
rag	1.77	1.91	**3.08**	1.67	1.74	**2.87**	1.89	1.92	**3.24**	1.79	1.83	**3.13**
all	1.74	1.88	**3.08**	1.70	1.82	**2.99**	1.84	1.91	**3.24**	1.78	1.84	**3.14**

Table 10: Conditional acceptance rates q_k, the partial sum of the chain identity up to $k = 8$, and MAT, Qwen2.5-7B. LookAhead has no steps beyond $A = 4$.

Benchmark	**Method**	q_2	q_3	q_4	q_5	q_6	q_7	q_8	partial sum	MAT
HumanEval	LookAhead	0.50	0.78	0.87	0.00	–	–	–	2.22	2.22
HumanEval	LogitSpec	0.48	0.41	0.73	0.79	0.73	0.86	0.91	2.15	2.77
HumanEval	FlashMTP	0.85	0.70	0.64	0.63	0.69	0.72	0.84	3.45	4.71
GSM8K	LookAhead	0.48	0.67	0.70	0.00	–	–	–	2.02	2.02
GSM8K	LogitSpec	0.64	0.37	0.53	0.52	0.51	0.66	0.67	2.14	2.17
GSM8K	FlashMTP	0.89	0.76	0.66	0.58	0.56	0.50	0.48	3.52	3.55
Spec-Bench	LookAhead	0.37	0.65	0.74	0.00	–	–	–	1.78	1.78
Spec-Bench	LogitSpec	0.48	0.30	0.58	0.59	0.60	0.77	0.79	1.84	1.93
Spec-Bench	FlashMTP	0.81	0.64	0.58	0.58	0.59	0.61	0.62	3.01	3.15

Table 11: Performance across different temperatures

Temp	Method	HumanEval		GSM8K		Spec-Bench	
		MAT	Speedup	MAT	Speedup	MAT	Speedup
0.0	AutoRegressive	1.00	1.00×	1.00	1.00×	1.00	1.00×
	LookAhead	2.22	1.99×	2.02	1.67×	1.78	1.46×
	LogitSpec	2.77	2.44×	2.17	1.70×	1.93	1.54×
	FlashMTP	**4.71**	**3.77×**	**3.55**	**2.40×**	**3.15**	**2.14×**
0.6	AutoRegressive	1.00	1.00×	1.00	1.00×	1.00	1.00×
	LookAhead	2.34	2.21×	2.10	1.75×	1.79	1.47×
	LogitSpec	2.71	2.54×	2.15	1.73×	1.85	1.47×
	FlashMTP	**4.47**	**3.48×**	**3.48**	**2.27×**	**2.91**	**1.94×**
1.0	AutoRegressive	1.00	1.00×	1.00	1.00×	1.00	1.00×
	LookAhead	2.33	2.11×	2.06	1.83×	1.71	1.41×
	LogitSpec	2.68	2.44×	2.11	1.78×	1.79	1.44×
	FlashMTP	**4.37**	**3.33×**	**3.38**	**2.33×**	**2.77**	**1.83×**

F.4 Source-level acceptance probabilities

The acceptance statistics are kept per source, depth band and evidence band. For the Qwen2.5-7B Spec-Bench run, the depth-one acceptance probabilities averaged over the snapshots taken at the end of every conversation turn are 0.16 for both keys of the successor index, 0.13 for the context match, 0.05 for the forecast from rejected siblings and 0.008 for tokens of the last logit below the first rank. The values include the weak priors and are running estimates rather than exact counts, but their spread is the point: the successor index is accepted twenty times more often than the ranking of the last logit. A static allocation that treated these sources alike would give most of its candidate tokens to candidates that are rarely accepted, which is what the acceptance statistics prevent.