The Verification Bottleneck

Toward Adaptive, Lossless Speculative Decoding for Autoregressive Language Models

Binod Kumar

AgentsArchitects.ai

Mentor, E-Cell, Indian Institute of Technology Bombay and Banaras Hindu University, India

June 2026

Abstract

Speculative decoding accelerates autoregressive language-model inference by proposing tokens with a cheap draft model and verifying them in parallel with the target model, preserving the target output distribution exactly. Most of the literature has optimized the draft: better draft architectures, tree-structured proposals, and reduced draft vocabularies. A recent systematic measurement of speculative decoding on a production engine complicates that focus. It reports that verification by the target model, not drafting, dominates end-to-end execution, that acceptance length varies sharply across token positions, requests, and datasets, and that observed speedups fall well short of the theoretical bound. This survey takes that finding as its organizing question and maps the design space it implies. We formalize speculative decoding as a three-term latency model, separate draft-side from verify-side optimization, and place recent methods within it. On the draft side, we examine the vocabulary projection bottleneck and the move from static reduced vocabularies (FR-Spec, VocabTrim, EAGLE-3) to per-step vocabulary speculation (SpecVocab). On the verify side, we examine semi-autoregressive drafting and load-aware, confidence-scheduled verification (DSpark). We argue that the field is converging on a single under-formalized object: an inference procedure that is adaptive to live load and per-token acceptance, while remaining lossless with respect to the target distribution. We state seven properties such a procedure must satisfy, show that no current method satisfies all seven, and close with six open problems, the sharpest of which is reducing the per-token cost of verification itself rather than the cost of the draft.

Keywords: speculative decoding; autoregressive inference; verification bottleneck; draft model; acceptance length; vocabulary speculation; semi-autoregressive generation; lossless acceleration; adaptive scheduling; throughput

1. Introduction

Autoregressive decoding is sequential by construction. Each generated token requires a full forward pass over the target model, and the pass is bound by memory bandwidth rather than arithmetic: for a single sequence at small batch size, the hardware spends most of its time moving parameters from high-bandwidth memory to the compute units, and the matrix multiplications are comparatively cheap. Generating a sequence of length n therefore costs roughly n memory-bound passes, and latency is dominated by the count of those passes rather than by the work inside each one.

Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) breaks the one-pass-per-token coupling. A small draft model proposes a block of candidate tokens, and the target model verifies the entire block in a single forward pass, accepting the longest prefix consistent with its own distribution and correcting the first divergence. Because a single target pass over k positions costs far less than k separate passes, and because the verification can be made distribution-preserving, the method produces output identical in distribution to standard sampling from the target while doing fewer expensive passes. Losslessness is the property that distinguishes

speculative decoding from lossy accelerations such as pruning or quantization, and it is the reason the method has been adopted in production inference engines (Kwon et al., 2023; Zheng et al., 2024) and at scale (Tang et al., 2025).

Most subsequent work has improved the draft. The EAGLE series introduced feature-level autoregression and tree-structured drafting to raise acceptance length (Li et al., 2024a, 2024b, 2025b). Auxiliary-head methods such as Medusa (Cai et al., 2024) and recurrent or retrieval-based drafters (Cheng et al., 2024; Gritta et al., 2025) attack the same target. The implicit assumption across this line is that the binding constraint is the quality and speed of the proposal.

A recent systematic study questions that assumption. Evaluating multiple speculative-decoding variants on a production-grade engine across workloads, model scales, and batch sizes, the authors report that verification by the target model dominates execution time, that acceptance length varies markedly across positions, requests, and datasets, and that the gap between observed and theoretical speedup is substantial (the *Performance or Illusion?* study). If verification is the dominant term, then a research program that optimizes only the draft is optimizing a subdominant cost.

This survey is organized around that observation. We make four contributions. First, we give a compact latency decomposition that separates the draft cost, the verification cost, and the acceptance length, and use it to define draft-side and verify-side optimization precisely. Second, we place recent methods in that decomposition, treating draft-side vocabulary speculation (SpecVocab) and verify-side confidence scheduling (DSpark) as complementary attacks on different terms. Third, we extract from the empirical landscape seven properties that an adaptive, lossless inference procedure must satisfy, and show that no current method satisfies all of them. Fourth, we state six open problems, foregrounding verification-cost reduction and provably bounded adaptive scheduling.

2. Speculative Decoding, Formalized

Let p denote the target model and q the draft model over a shared vocabulary V. Standard autoregressive sampling draws each token from $p(x_t \mid x_{<t})$ in its own forward pass. Speculative decoding instead proceeds in rounds. In each round, q generates a candidate block of k tokens autoregressively, conditioned on the current context. The target model p then evaluates all k positions in a single parallel forward pass, yielding $p(x \mid x_{<t})$ at each position. A verification rule, typically the rejection-sampling scheme of Leviathan et al. (2023) and Chen et al. (2023), accepts the longest prefix that is consistent with sampling from p and resamples a single corrected token at the first rejection. The accepted tokens plus the correction extend the context, and the round repeats.

Two quantities govern the result. The acceptance length a is the expected number of draft tokens accepted per round, reflecting the alignment between q and p. The per-round cost is the draft cost to produce k tokens, written $c_q(k)$, plus the cost of one target verification pass, written $c_p(k)$. Per accepted token, the latency is approximately

$$L \;=\; \frac{c_q(k) + c_p(k)}{a}.$$

This is the whole optimization surface in one line. Draft-side work lowers c_q. Acceptance-improving work raises a. Verify-side work lowers c_p or makes the choice of how much to verify depend on the live value of a. The classic mistake the latency model exposes is to optimize a in isolation: a method can raise acceptance length and still lose end-to-end if it raises c_q or c_p by more than the acceptance gain returns. Acceptance length and throughput are different metrics, and the measurement literature is explicit that the former does not imply the latter

(the *Performance or Illusion?* study; Zhao et al., 2025).

Three structural facts about the terms matter for what follows. The verification cost c_p scales with target model size and with the vocabulary projection at each verified position, and the measurement study finds it dominant in practice. The acceptance length a is not a constant: it decays along the draft block, because tokens proposed further from the conditioning context are less reliable, and it varies across requests, datasets, and token positions. And the draft cost c_q is itself dominated, for modern large-vocabulary models, by the output projection over V rather than by the draft network proper, a point we take up next.

3. The Draft Side: The Vocabulary Projection Bottleneck

A lightweight draft model is often light only in its network. Its final step computes logits over the full target vocabulary, an operation that scales with $|V|$ and the model dimensionality d. As vocabularies have grown, from tens of thousands of tokens in early transformers to well over a hundred thousand in current models (Tao et al., 2024; Takase et al., 2025; Huang et al., 2025), this projection has become the dominant component of draft time. Zhao et al. (2025) identified the output-distribution computation as the principal drafting cost in the widely used EAGLE framework. The draft is fast in network terms and slow in wall-clock terms, because its last act is an expensive projection.

The first response was to shrink the projection by restricting the draft vocabulary to a fixed frequent subset, exploiting the Zipfian token-frequency distribution (Zipf, 1949). FR-Spec prunes the output embeddings of rare tokens using frequencies from a large pretraining corpus (Zhao et al., 2025); VocabTrim performs the analogous pruning using target-generated synthetic data (Goel et al., 2025); and EAGLE-3 trains a reduced-vocabulary head from scratch rather than pruning after training (Li et al., 2025b). A vocabulary size around 32K is reported as a throughput-optimal static choice (Zhao et al., 2025).

The limitation is structural rather than a matter of tuning. A fixed subset is context-agnostic. When the next target token falls outside the subset, the draft cannot propose it, the round terminates early, and the speculative speedup for that round is lost. The static methods therefore trade acceptance for projection cost along a fixed frontier set before inference.

SpecVocab (Williams et al., 2026) reframes the problem as one of speculation rather than pruning: in addition to speculating on the next tokens, it speculates on which slice of the vocabulary is relevant at each step. A low-dimensional ranking module computes approximate logits over the full vocabulary from a reduced intermediate hidden state, the top-k indices form a per-step candidate set $\mathcal{K}_t$, and exact logits are computed only over $\mathcal{K}_t$ using a fused kernel that reads each selected embedding from memory once. The complexity moves from $O(|V|\,d)$ for the full projection, or $O(|V'|\,d)$ for a reduced static vocabulary V', to $O(|V|\,d' + k\,d)$, with $d' \ll d$. Reported results across two model families place SpecVocab above EAGLE-3, FR-Spec, and VocabTrim in acceptance length, with average throughput gains over EAGLE-3 of roughly 4 to 8 percent depending on model, reaching about 8 percent on the larger open model in the study. The method needs exact logits for only about 1 to 2 percent of the vocabulary per step, over an order of magnitude fewer tokens than the static 32K subset, and its custom kernel achieves a 3x to 5x speedup over a naive indexed projection. The relevant limit is its scope: SpecVocab lowers c_q and preserves a, but it does not touch the dominant verification term c_p.

4. The Verify Side: Acceptance Decay and Load-Dependent Cost

Two facts make verification the harder term. First, acceptance decays along the block, so verifying a long draft commits target compute to positions increasingly likely to be rejected. Second, the value of verification depends on the serving regime. Speculative decoding spends

spare target compute to avoid memory traffic, which is favorable at small batch size where the hardware is memory-bound and compute is idle. As batch size grows, both speculative and standard decoding move toward the compute-bound regime, and the extra verification work competes for compute that is already saturated, so the speedup shrinks and can become negative (Nightjar; the *Performance or Illusion?* study). A fixed speculation length, the common default in production engines, is therefore optimal only at a single point on the load curve.

Two ideas address these facts. The first is to make the draft cheaper to condition without lengthening the dependency chain. Semi-autoregressive drafting proposes a block with a parallel backbone and a small sequential head, reducing the suffix-decay penalty that pure autoregressive drafting incurs late in the block. The second is to make the verification budget a runtime decision. A confidence head can estimate, per drafted position, the probability that the token will survive verification given accepted predecessors, and a scheduler can then set the verification length per request against live hardware utilization, verifying more when compute is idle and less when it is busy.

DSpark (DeepSeek, 2026) combines both. It pairs a semi-autoregressive draft with a confidence head whose per-position scores are calibrated by a post-hoc temperature-scaling step, reported to reduce expected calibration error from the single-digit-percent range to about one percent, and a hardware-aware scheduler that sets verification length from a throughput curve profiled once at startup. The system is lossless: outputs match the target distribution. The reported gains are sizeable but warrant care in interpretation. Offline accepted-length improvements over EAGLE-3 and a sibling drafter are reported in the range of roughly 16 to 31 percent, and per-user generation speedups of roughly 60 to 85 percent are reported on the vendor's own models against the vendor's prior multi-token-prediction baseline, measured in the vendor's serving regime and not yet independently reproduced on other stacks. The numbers should be read as a strong existence proof for load-aware, calibrated verification scheduling, not as a settled cross-stack benchmark. The relevant limit is reproducibility and coupling: the headline figures are baseline- and regime-specific, and the deployed configuration is tied to a particular model family.

5. A Design Space, and Where Each Method Sits

The latency model gives a clean way to locate methods. Draft-side methods lower c_q; acceptance methods raise a; verify-side methods lower c_p or make the verification budget responsive to a and to load. The recent literature populates the space as follows.

EAGLE-3 raises a through feature-level autoregression and tree drafting and lowers c_q through a fixed reduced vocabulary, and it is the standard strong baseline. Its reduced vocabulary is the static choice the draft-side successors improve on. FR-Spec and VocabTrim lower c_q by pruning the projection, the former from corpus frequencies and the latter from target-generated frequencies, with the target-frequency variant generally stronger across tasks. SpecVocab lowers c_q further and recovers the acceptance lost to static pruning by making the vocabulary subset per-step and context-aware. None of these four touches c_p.

DSpark is the one entry that attacks the verify side directly, lowering the effective c_p per accepted token by scheduling the verification budget against live load and per-position acceptance estimates, with semi-autoregressive drafting reducing acceptance decay so that the scheduled budget is spent on positions more likely to survive. The inference engines vLLM (Kwon et al., 2023) and SGLang (Zheng et al., 2024) supply the substrate that makes these costs measurable rather than theoretical; the measurement study that motivates this survey was possible precisely because the engine is optimized enough that the vocabulary and verification costs are not masked by launch overhead.

Read together, the draft-side and verify-side lines are complementary rather than competing. SpecVocab and DSpark optimize different terms of the same expression, and there is no reported obstacle to composing per-step vocabulary speculation on the draft with confidence-scheduled verification on the target. The near-term frontier is composition across the terms, not a single method that subsumes them.

6. Properties of an Adaptive, Lossless Inference Procedure

The empirical picture points to a single under-formalized object: an inference procedure that adapts to live conditions while preserving the target distribution exactly. We state seven properties such a procedure should satisfy. They are intended as a specification against which methods can be measured, not as claims that any one method meets them.

1. Exact losslessness. The procedure must preserve the target output distribution, so that acceleration never trades against sample quality. This is the property that separates speculative decoding from lossy compression, and it must hold under any adaptive choice the procedure makes.

2. Instrumented term separation. The procedure must expose the draft cost, the verification cost, and the acceptance length as separately measured quantities, because the dominant term cannot be optimized if it is not isolated.

3. Load-conditioned behavior. Verification depth and draft length must be functions of live batch size and utilization, since the optimal speculation length moves with the position on the memory-bound to compute-bound curve.

4. Acceptance-conditioned depth. The procedure must estimate per-position acceptance probability and condition the verification budget on it, rather than committing target compute to positions likely to be rejected.

5. Calibrated estimates. Any acceptance or confidence estimate that drives scheduling must be calibrated, since miscalibrated estimates mis-budget verification and can erase the gain they were meant to produce.

6. Stated throughput-latency operating point. The procedure must expose its position on the trade between per-request latency and aggregate throughput, and move along that frontier deliberately, since a single configuration is optimal at only one load.

7. Workload-representative evaluation. Acceptance and throughput must be reported under realistic load and task mixes, because measurements at batch size one on research prototypes do not predict behavior on a loaded engine.

Most reported methods satisfy one or two of these. Almost none satisfy all seven. The gap between the properties and the methods is the research surface this survey is pointing at.

7. Open Problems

The following problems are, in our reading, the most consequential and the most clearly posed.

A standing methodological caution follows from the latency model: a reported acceptance-length improvement is not a speedup, and a speedup measured at one batch size on one stack does not transfer to another. Claims should be stated against the term of L they move.

1. **Verification-cost reduction.** Drafting has received intense attention; verification, the dominant term, has received much less. What lowers the per-token cost of the target verification pass itself, losslessly, without simply shrinking the target? Candidate directions include sparsified or hierarchical verification and verification that reuses partial target com-

putation across positions.

2. **Provably bounded adaptive scheduling.** Given live load, per-position acceptance statistics, and a profiled throughput curve, what verification depth and draft length minimize expected per-token latency, and can a scheduler approach that optimum with regret bounds rather than heuristics? The confidence-scheduling results to date are empirical; the control problem is well-defined and largely unanalyzed.

3. **Closing the gap to the theoretical bound.** The measurement study reports a substantial gap between observed speedup and the analytical upper bound. What fraction of that gap is irreducible given memory-bandwidth and scheduling constraints, and what fraction is recoverable engineering? A tight characterization would tell the field where further effort can and cannot pay.

4. **Acceptance prediction under distribution shift.** Confidence-scheduled verification depends on calibrated per-position acceptance estimates. How stable is that calibration as workloads drift across tasks and domains, and what online recalibration keeps it valid without a separate labeled signal?

5. **Retrieval and structure in the draft.** Acceptance decays along the block, and context-aware vocabulary speculation shows that conditioning the draft on the right structure recovers acceptance cheaply. What draft conditioning, whether retrieval-based, structural, or hierarchical, raises acceptance length per unit of draft cost, and how does it interact with the verification budget?

6. **Cross-stack reproducibility and evaluation.** Reported speedups are often baseline- and regime-specific. What is a standard, workload-representative protocol for reporting speculative-decoding gains, including the baseline, batch size, task mix, and engine, so that results from different stacks are comparable at all? Existing acceptance-centric reporting is insufficient, since acceptance length does not determine throughput.

Conclusion

The center of gravity in speculative-decoding research is shifting. The draft side is now well populated, from tree drafting to static vocabulary reduction to per-step vocabulary speculation, and the latency model explains why those methods help: they lower the draft term or raise acceptance. The measurement evidence indicates that the binding constraint lies elsewhere, in the verification term and in the load- and position-dependence of acceptance, and the most recent verify-side work treats the verification budget as something to schedule rather than fix. What the field is converging on, without yet naming it precisely, is a single object: a decoding procedure that is adaptive to live load and per-token acceptance while remaining exactly lossless. The draft finds candidates; verification decides what they cost. The open problems live on the side that decides.

References

Note: identifiers and years below should be verified against the primary record before formal submission.

References

[1] Williams, M., Kwon, Y. D., Li, R., Kouris, A., and Venieris, S. I. *Speculative Decoding with a Speculative Vocabulary.* Per-step context-aware vocabulary speculation; throughput and acceptance gains over EAGLE-3, FR-Spec, and VocabTrim. Preprint, arXiv:2602.13836,

2026.

[2] (Sky Computing Lab et al.) *Speculative Decoding: Performance or Illusion?* Systematic measurement on a production engine (vLLM) across SD variants, scales, and batch sizes; identifies verification as the dominant cost and a gap to the theoretical bound. Preprint, arXiv:2601.11580, 2026.

[3] DeepSeek-AI. *DSpark: Confidence-Scheduled Speculative Decoding with Semi-Autoregressive Generation.* Semi-autoregressive drafting with a calibrated confidence head and hardware-aware verification scheduling; lossless. Technical report, 2026. [identifier: verify]

[4] Leviathan, Y., Kalman, M., and Matias, Y. *Fast Inference from Transformers via Speculative Decoding.* Foundational draft-then-verify formulation with distribution-preserving acceptance. ICML, 2023.

[5] Chen, C., Borgeaud, S., Irving, G., Lespiau, J.-B., Sifre, L., and Jumper, J. *Accelerating Large Language Model Decoding with Speculative Sampling.* Concurrent foundational speculative-sampling work establishing losslessness. Preprint, arXiv:2302.01318, 2023.

[6] Li, Y., Wei, F., Zhang, C., and Zhang, H. *EAGLE-3: Scaling up Inference Acceleration of Large Language Models via Training-Time Test.* Feature-level autoregression, tree drafting, reduced-vocabulary head; the standard strong baseline. Preprint, arXiv:2503.01840, 2025.

[7] Li, Y., Wei, F., Zhang, C., and Zhang, H. *EAGLE-2: Faster Inference of Language Models with Dynamic Draft Trees.* Dynamic draft-tree drafting underpinning later EAGLE work. EMNLP, 2024.

[8] Li, Y., Wei, F., Zhang, C., and Zhang, H. *EAGLE: Speculative Sampling Requires Rethinking Feature Uncertainty.* Feature-level draft autoregression. ICML, 2024.

[9] Zhao, W., Pan, T., Han, X., Zhang, Y., Sun, A., Huang, Y., et al. *FR-Spec: Accelerating Large-Vocabulary Language Models via Frequency-Ranked Speculative Sampling.* Identifies the output-distribution bottleneck in drafting; static frequency-ranked vocabulary pruning. ACL, 2025.

[10] Goel, R., Agrawal, S., Gagrani, M., Park, J., et al. *VocabTrim: Vocabulary Pruning for Efficient Speculative Decoding in LLMs.* Post-training vocabulary pruning from target-generated frequencies. Preprint, arXiv:2506.22694, 2025.

[11] Cai, T., Li, Y., Geng, Z., Peng, H., Lee, J. D., Chen, D., and Dao, T. *Medusa: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads.* Auxiliary decoding heads for parallel drafting. ICML, 2024.

[12] Cheng, Y., Zhang, A., Zhang, X., Wang, C., and Wang, Y. *Recurrent Drafter for Fast Speculative Decoding in Large Language Models.* Recurrent draft head. Preprint, arXiv:2403.09919, 2024.

[13] Gritta, M., Xue, H., and Lampouras, G. *DReSD: Dense Retrieval for Speculative Decoding.* Retrieval-based draft proposal. ACL Findings, 2025.

[14] Stern, M., Shazeer, N., and Uszkoreit, J. *Blockwise Parallel Decoding for Deep Autoregressive Models.* Early propose-and-verify parallel decoding. NeurIPS, 2018.

[15] Xia, H., Yang, Z., Dong, Q., Wang, P., Li, Y., Ge, T., et al. *Unlocking Efficiency in LLM Inference: A Comprehensive Survey of Speculative Decoding.* Survey; source of the

Spec-Bench evaluation and the acceptance-length and throughput metric conventions. ACL Findings, 2024.

[16] Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., et al. *Efficient Memory Management for Large Language Model Serving with PagedAttention (vLLM).* Production inference engine used for the measurement study. SOSP, 2023.

[17] Zheng, L., Yin, L., Xie, Z., Sun, C., Huang, J., Yu, C. H., et al. *SGLang: Efficient Execution of Structured Language Model Programs.* Optimized serving framework with CUDA-graph execution. NeurIPS, 2024.

[18] Tang, B., Fu, C. C., Kou, F., Sizov, G., Zhang, H., et al. *Efficient Speculative Decoding for Llama at Scale: Challenges and Solutions.* Speculative decoding deployed at production scale. Preprint, arXiv:2508.08192, 2025.

[19] Tao, C., Liu, Q., Dou, L., Muennighoff, N., Wan, Z., Luo, P., Lin, M., and Wong, N. *Scaling Laws with Vocabulary: Larger Models Deserve Larger Vocabularies.* Motivates the growth of target vocabularies. NeurIPS, 2024.