

# Joint Optimization of Retrieval-Augmented KV Caching and Adaptive Speculative Decoding for Autoregressive Generation

Chenyi Huang

chuan128@jh.edu

Johns Hopkins University

Zitan Chen

Carnegie Mellon University

Zhenning Guo

New York University

Wentao Zhang

University of Southern California

---

## Short Report

**Keywords:** KV cache, speculative decoding, autoregressive generation, cost model, LLM serving

**Posted Date:** October 14th, 2026

**DOI:** <https://doi.org/10.21203/rs.3.rs-11354781/v1>

**License:**   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

**Additional Declarations:** The authors declare no competing interests.

---

# Joint Optimization of Retrieval-Augmented KV Caching and Adaptive Speculative Decoding for Autoregressive Generation

Chenyi Huang<sup>1,\*</sup>, Zitan Chen<sup>2</sup>, Zhenning Guo<sup>3</sup>, Wentao Zhang<sup>4</sup>

<sup>1</sup>Johns Hopkins University, Baltimore, 21218, USA

<sup>2</sup>Carnegie Mellon University, Pittsburgh, 15213, USA

<sup>3</sup>New York University, New York, 10012, USA

<sup>4</sup>University of Southern California, Los Angeles, 90089, USA

\*chuan128@jh.edu, zitanc@alumni.cmu.edu, zg986@nyu.edu, wentao@alumni.usc.edu

**Abstract**—Retrieval-assisted key-value (KV) reuse and speculative decoding remove different costs, but their naive composition can leave draft and verification overhead after prefill is shortened. RAKV-ASD combines exact causal prefix validation with a measured controller that selects reuse and  $k \in \{0, 1, 2, 4, 6, 8\}$ . Across 3,600 timed executions on two GPT-2-family pairs, every greedy output matched target-only decoding. Pooled results show 64.0% lower latency and 170.8% higher throughput than reuse plus fixed  $k=4$ , while remaining within 0.45% of the KV-only baseline. The controller selected  $k=0$ , achieved 100%  $k$ -decision accuracy and 0.32% mean regret despite 32.1% point-prediction MAPE because costs rank actions rather than provide calibrated latency forecasts. Sensitivity and a third-pair study place the nonzero- $k$  crossover at predicted acceptance  $\geq 0.894$ . The result is a measured boundary and a mechanism that avoids counterproductive speculation, not a universal speedup.

**Keywords**—KV cache, speculative decoding, autoregressive generation, cost model, LLM serving

## I. INTRODUCTION

Transformer decoding reuses layerwise key-value (KV) states to avoid recomputing causal history [1], [2]. Serving systems add iteration scheduling [3], paged memory [4], and prefix-aware execution [5]; Prompt Cache [6], Hydragen [7], and CacheBlend [8] extend reuse across requests. Speculative decoding drafts with a smaller model and verifies with the target without changing the target distribution [9], [10]. Feature-level [11] and tree-based [12] variants broaden its regime, but wall-clock gains remain hardware- and workload-dependent [13], [14]. Serial drafting, verification, and rejection costs remain after a prefix hit shortens target prefill.

RAKV-ASD asks which reuse and speculation action should run for each request on one 8-GB GPU. It evaluates reuse jointly with  $k \in \{0, 1, 2, 4, 6, 8\}$ , including  $k=0$ . Its contributions are exact model-keyed prefix validation with bounded memory, an implementation-matched measured cost model, and a 3,600-execution evaluation with sensitivity, controller, and larger-pair studies. Fixed speculation is uneconomic in this regime; joint selection avoids its penalty without overstating the small difference from target-only decoding.

## II. BACKGROUND AND MOTIVATION

### A. Safe Cross-Request KV Reuse

A reusable state must match the exact causal prefix. RAKV-ASD stores token IDs, model/tokenizer identifiers, KV tensors, next-token logits for complete prompts, metadata, and byte counts. A 12-byte BLAKE2b digest of the first 16 tokens selects a candidate bucket; exact longest-common-prefix (LCP) validation then establishes safety, never semantic similarity. For a partial hit of  $q$  tokens, only  $q-1$  positions are cloned and the boundary token is recomputed; exact repeats may reuse stored next logits. Each model pool has a fixed dtype/configuration and a 512-MiB budget, ignores prefixes shorter than 16 tokens, refreshes duplicates, and evicts least-recently-used entries. Lookup is expected  $O(1)$ , validation  $O(bL)$ , and the prototype's eviction scan  $O(N)$ .

### B. Profiling the Interaction

At context 512, GPT-2/DistilGPT-2 target/draft prefill took 25.74/16.72 ms, a target step 9.02 ms, and  $k=4$  verification 8.67 ms. GPT-2 Medium/GPT-2 took 59.98/25.98, 14.80, and 14.91 ms, respectively. Lookup was 0.015 ms and cached state used 108/264 KiB per token. Reuse is cheap; nonzero  $k$  still adds serial draft work and often removes too few target steps. Longer contexts favor reuse, whereas longer outputs repeat the speculative-cycle cost.

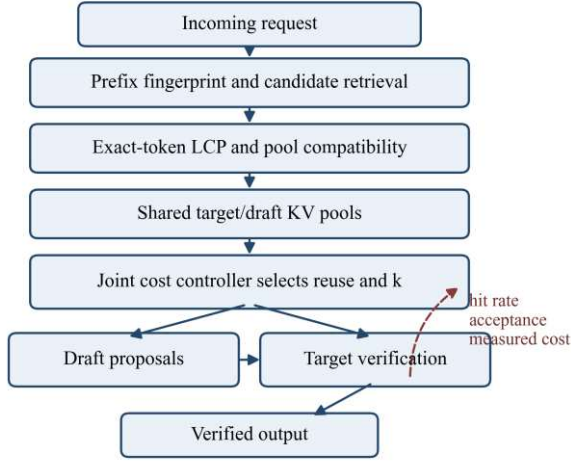


Fig. 1. RAKV-ASD runtime. Retrieval is separated from exact safety validation; observed hit rate, acceptance, and measured cost feed the next decision.

### C. Boundary Intuition

Speculation helps only when avoided target steps exceed serial draft, verification, and correction costs. Reuse subtracts the same target-prefill term from  $k=0$  and  $k>0$ , while a speculative path may also initialize a draft cache; a longer hit can therefore make speculation relatively less attractive. A cache hit and high acceptance are insufficient unless their saving exceeds the current target/draft kernel ratio. For short outputs, one-time prefill ranking dominates; for long outputs, acceptance and repeated cycle costs dominate. Memory may also exclude a faster candidate. The controller compares all feasible reuse/ $k$  actions instead of optimizing the mechanisms sequentially.

## III. RAKV-ASD FRAMEWORK

### A. Verification and Online Statistics

For  $k>0$ , the draft emits  $k$  candidates sequentially; the target verifies them together, accepts the matching prefix, corrects the first mismatch, and crops both caches to the committed context. The accepted/proposed ratio updates an exponentially weighted estimate  $\alpha_t = 0.8\alpha_{t-1} + 0.2(a_t/p_t)$ , initialized at 0.55;  $k=0$  causes no update. Expected accepted progress is

$$E_{k(\alpha)} = \frac{1-\alpha^k}{1-\alpha}. \quad (1)$$

Calibration measures target/draft prefill, one-token steps, and verification for each  $k$ ; runtime observations refine acceptance and timing. This follows distribution-preserving speculative sampling [9], [10] while exposing implementation cost.

### B. Implemented Joint Cost Model

Let  $L$  be prompt length,  $q$  the validated LCP,  $r \in \{0,1\}$  the reuse action, and  $O$  the requested output. The timing symbols in (2)–(3) denote measured target and draft prefill per token, one-token target and draft steps, and  $k$ -token verification. With  $q=0$  when  $r=0$ , the controller estimates

$$C_{0(r)} = t_{lookup} + (L - rq)t_p^T + Ot_T, \quad (2)$$

$$C_{k(r)} = t_{lookup} + (L - rq)(t_p^T + t_p^D) + \left\lfloor \frac{O}{E_{k(\alpha)}} \right\rfloor \{kt_D + t_{V(k)} + (1 - \alpha^k)t_T\} + 0.02Ot_T \left\lceil \frac{M(k,r)}{B} \right\rceil, \quad k > 0. \quad (3)$$

The last term is a measured-scale memory penalty; infeasible  $M(k,r)>B$  actions are excluded. The controller chooses the remaining argmin. These estimates rank actions within a workload; they are not calibrated wall-clock forecasts. Experiments use batch one, so no unsupported queue term is included. Algorithm 1 summarizes the path.

### ALGORITHM 1 RAKV-ASD ONLINE DECISION

**Input:** prompt tokens  $x$ ; target  $T$ ; draft  $D$ ; memory budget  $B$

- 1: Fingerprint  $x[:16]$ ; retrieve model-specific candidates.
- 2: Check tokenizer/pool compatibility and compute exact token LCP  $q$ .
- 3: For  $r \in \{0,1\}$ ,  $k \in \{0,1,2,4,6,8\}$ , predict  $C(k,r)$ ; reject  $M(k,r)>B$ .
- 4: Choose  $(k^*, r^*) = \arg \min C$ ; reuse  $q-1$  tokens for a partial hit.
- 5: If  $k^*=0$ , run  $T$ ; else draft, target-verify, correct, and crop caches.
- 6: Update acceptance, measured costs, cache bytes, reuse count, and LRU time.

## IV. EXPERIMENTAL EVALUATION

### A. Setup

Experiments use WikiText-2 [15] and official HumanEval prompts [16], cropped rather than repeated. Each four-request group starts cold; later requests preserve the configured exact prefix and change the suffix. Baselines are AR, KV-only, Spec-k4, Joint-k4, and RAKV-ASD. Metrics include full lookup/admission/prefill/generation latency, committed-token throughput, TTFT, CUDA  $\Delta$ Peak, observed exact-LCP hit rate, acceptance, and token equality. Table I gives the grid.

TABLE I. EXPERIMENTAL SETUP

| Item                | Setting                                                                    |
|---------------------|----------------------------------------------------------------------------|
| Main target / draft | GPT-2 / DistilGPT-2; GPT-2 Medium / GPT-2                                  |
| Feasibility pair    | GPT-2 Large / GPT-2 Medium (FP32, separate 30-run slice)                   |
| Data and decoding   | WikiText-2 test; HumanEval prompts; greedy; batch 1                        |
| Main grid           | contexts 128/256; output 24; shared prefix 0/50/75%                        |
| Sensitivity grid    | contexts 128/256/512/768; outputs 24/64/128; $k=0/1/2/4/6/8$               |
| Profiling           | prefill through 1,024 tokens; target/draft step; verify; lookup; KV bytes  |
| Hardware            | RTX 5060 Laptop 8 GB; i7-14650HX; 31.7 GiB RAM                             |
| Software            | Windows 11; PyTorch 2.8.0+cu128; Transformers 4.55.4; FP32 eager; TF32 off |

Methods receive three warm-ups and five timed repetitions. Table II first averages the balanced workload cells within each repetition, then reports the mean and sample SD across the five repeat-level means. The original main matrix contributes 2,400 runs and the extended context/output/ $k$  matrix 1,200. Their common five-method comparison remains balanced at 600 runs per method (3,000 total). A separate 30-run GPT-2 Large/GPT-2 Medium FP32 slice is excluded because it does not repeat every cache baseline. All reported data use FP32 eager attention with TF32 off.

### B. Overall Performance

Table II reports both main pairs. Pooled RAKV-ASD latency/throughput were 362.05 ms/84.20 token/s, essentially AR parity, and 64.0%/170.8% better than Joint-k4 because  $k=0$

was selected. KV-only remained strongest at 360.43 ms; RAKV-ASD was 0.45% slower and 0.81% lower-throughput, so it is not a general speedup over reuse alone.

Both pairs show the same ordering. Relative to Joint-k4, RAKV-ASD reduced latency and raised throughput by 78.5%/358.4% for GPT-2/DistilGPT-2 and 39.1%/62.2% for GPT-2 Medium/GPT-2. Identical hit rates isolate speculation; 1.4–10.2-ms SDs are far below the 287–996-ms Joint-k4 gaps but comparable to sub-3-ms differences among AR, KV-only, and RAKV-ASD. Fixed-k paths also used 301–308 MiB versus about 159 MiB and achieved 27.3% acceptance.

TABLE II. PAIR-SPECIFIC MAIN-EVALUATION PERFORMANCE WITH FIVE-RUN VARIATION

| Target / Draft       | Method          | Latency (ms)       | Tok/s              | Hit (%)             |
|----------------------|-----------------|--------------------|--------------------|---------------------|
| GPT-2 / DistilGPT-2  | AR              | 272.6 ± 2.4        | 104.8 ± 0.7        | —                   |
|                      | KV-only         | 270.8 ± 3.4        | 105.7 ± 1.4        | 32.23 ± 0.00        |
|                      | Spec-k4         | 1264.0 ± 5.6       | 23.0 ± 0.1         | —                   |
|                      | Joint-k4        | 1269.9 ± 10.2      | 22.8 ± 0.3         | 32.23 ± 0.00        |
|                      | <b>RAKV-ASD</b> | <b>273.6 ± 1.8</b> | <b>104.5 ± 0.6</b> | <b>32.23 ± 0.00</b> |
| GPT-2 Medium / GPT-2 | AR              | 452.5 ± 1.4        | 63.5 ± 0.2         | —                   |
|                      | KV-only         | 450.1 ± 4.0        | 64.1 ± 0.4         | 32.23 ± 0.00        |
|                      | Spec-k4         | 745.8 ± 6.9        | 38.9 ± 0.4         | —                   |
|                      | Joint-k4        | 739.2 ± 4.1        | 39.4 ± 0.2         | 32.23 ± 0.00        |
|                      | <b>RAKV-ASD</b> | <b>450.5 ± 2.4</b> | <b>63.9 ± 0.3</b>  | <b>32.23 ± 0.00</b> |

### C. Context and Output Sensitivity

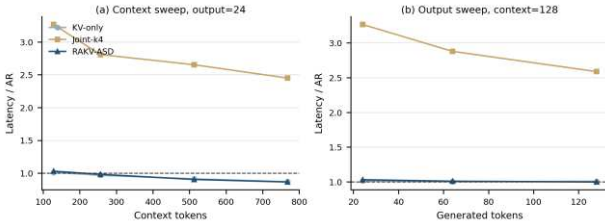


Fig. 2. Latency normalized to AR. Context sweep fixes output at 24; output sweep fixes context at 128. Curves pool both main pairs and reuse actions; lower is better.

With output 24, normalized RAKV-ASD latency falls from about 1.03 to 0.87 as context grows 128→768 because more target prefill can be avoided; Joint-k4 remains 2.45–3.28× AR. With context 128 and output 24→128, k=0 stays near AR while Joint-k4 narrows only from 3.27× to 2.59×. The fixed cost is amortized, but serial drafting still prevents a crossover. A 1,024-token profile measured prefill only because GPT-2 cannot decode beyond its learned positional limit without changing the model; results at 2,048 or 4,096 tokens are therefore not fabricated.

### D. Adaptive Controller Behavior

TABLE III. CONTROLLER VALIDATION OVER 24 WORKLOAD GROUPS

| Metric                          | Measured value       |
|---------------------------------|----------------------|
| Action points / workload groups | 216 / 24             |
| Mean rank correlation           | 0.780                |
| Action / k decision accuracy    | 83.3% / 100.0%       |
| Mean latency regret             | 2.28 ms (0.32%)      |
| Maximum latency regret          | 35.44 ms             |
| Point-prediction MAE / MAPE     | 630.9 ms / 32.1%     |
| Selected-k distribution         | k=0: 100.0%          |
| Predicted acceptance crossover  | $\alpha=0.894-0.973$ |

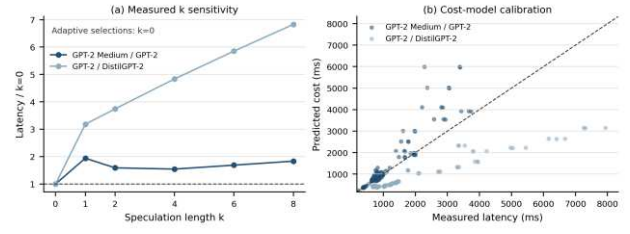


Fig. 3. (a) Measured k sensitivity relative to k=0. (b) Predicted versus measured action latency; the dashed line is equality.

Latency MAPE evaluates point-prediction calibration, while k accuracy evaluates the within-workload ordering used by the argmin. The model ranks feasible (r,k) actions rather than forecasting calibrated wall-clock time, so large point errors need not change the selected k. Across 24 groups, the latency separation between k=0 and nonzero-k actions kept k=0 correctly ranked despite 32.1% MAPE: k accuracy was 100%, Spearman correlation 0.780, full-action accuracy 83.3%, and mean regret 2.28 ms (0.32%). All four action errors were shallow reuse/no-reuse swaps. Ranking and realized regret therefore measure controller utility directly; MAPE still rules out claiming high-precision point prediction.

Measured sweeps confirm the choice: the best nonzero action was still 209–235% slower than k=0 for GPT-2/DistilGPT-2 and 14–113% slower for GPT-2 Medium/GPT-2, where the best nonzero k shifted between 2 and 4 with context, overlap, and output. The cost model predicts crossover only at  $\alpha=0.894-0.973$ . This is a model-derived boundary, not a measured speedup. Agreement approaching one, cheaper draft steps, fused verification, or a larger target/draft cost ratio is required before this stack should speculate.

TABLE IV. 512-TOKEN MICROPROFILE

| Target / draft     | T prefill (ms) | D prefill (ms) | T step (ms) | D step (ms) | Verify k=4 (ms) | KV KiB/tok |
|--------------------|----------------|----------------|-------------|-------------|-----------------|------------|
| GPT-2 / Distil     | 25.74          | 16.72          | 9.02        | 4.79        | 8.67            | 108        |
| GPT-2 Med. / GPT-2 | 59.98          | 25.98          | 14.80       | 7.76        | 14.91           | 264        |

Table IV explains the pair difference. For GPT-2/DistilGPT-2, one draft step exceeds half a target step, so four serial drafts already exceed target verification. GPT-2 Medium/GPT-2 has a better ratio and lower  $\alpha$  threshold. Verification is nearly flat across several k values, but sequential proposals dominate; fused or tree-based drafting could change this hardware/software-specific boundary. The reuse surface is much shallower: in a representative 128-token, 75%-overlap medium-pair group, no reuse was only 1.57 ms faster than the selected reuse/k=0 action (0.43% regret), while reuse/k=0 was correctly selected at 256 tokens. This local ambiguity does not affect the k=0 boundary.

### E. Ablation, Correctness, and Larger-Pair Feasibility

AR, KV-only, Spec-k4, Joint-k4, and RAKV-ASD isolate target decoding, reuse, speculation, forced composition, and joint selection. All 3,600 main executions matched AR token for token; lookup, insertion, and eviction are included, and hits follow exact LCP validation.

On the larger FP32 pair, GPT-2 Large/GPT-2 Medium used 4,438.9 MiB. At context 128/output 64,  $k=0$  averaged 1,394.24 ms; best nonzero  $k=2$  averaged 2,286.73 ms (64.0% slower) at 75.7% acceptance, with all 30 outputs exact. The gap is much smaller than for GPT-2/DistilGPT-2, consistent with movement toward the predicted boundary, but it still does not cross. An FP16 preflight had 10/30 mismatches under near-tied logits and was excluded rather than weakening the exact-output criterion.

Correctness compares complete generated token lists against AR after generation, not only after each verification batch. Partial hits change the first token after the shared prefix to exercise  $q-1$  boundary recomputation; cold requests cover no-hit, and exact repeats cover stored next logits. The reference uses the same target weights and every timed row records token equality. These cases exercise the cache transition most likely to produce a plausible but incorrect continuation.

#### F. Limitations and Validity

Scope is limited to three GPT-2-family pairs on one laptop GPU, greedy batch-one eager inference, two corpora, controlled prefixes, and contexts no longer than the architecture's 1,024-position limit. The results do not establish behavior for sampling, multi-billion-parameter assistants, tensor parallelism, continuous batching, paged kernels, concurrent queues, or data-center interference. The controller has no queue-pressure term, and the prototype's linear LRU scan should become an ordered structure at scale. A deployable cache ABI must also version layer count, tensor shape and dtype, attention implementation, and positional metadata; RoPE models require verified position and scaling compatibility.

External validity is bounded by the public corpora and controlled prefix construction. Warm-ups reduce cold-start variance, but five repetitions do not represent production interference, and the measured cost model must be recalibrated after any hardware, kernel, driver, model, or batching change. Internal validity rests on balanced method counts, identical prompts, synchronized GPU timing, explicit warm-ups, and target-token references. Few-millisecond host noise makes the 0.14% RAKV-ASD-AR difference near parity, whereas the 64.0% Joint- $k4$  gap is decisive. Construct validity is limited to latency, throughput, TTFT, CUDA allocation, hit rate, and acceptance; energy, loaded tail latency, and service-level violations remain unmeasured.

Each timed row records the model pair, corpus, sweep, context/output lengths, configured overlap, exact LCP hit rate, reuse action,  $k$ , acceptance, end-to-end and component timings, memory, committed token IDs, and exactness. Aggregates are recomputed from these rows. Controller validation replays all nine feasible reuse/ $k$  actions in each workload group and computes measured regret for the selected action. The 1,024-token profile and 30-run larger-pair slice remain separate from the balanced aggregate, preventing unequal grids from changing the headline mean.

A conservative deployment should calibrate after model loading, maintain timing estimates per model pair and kernel configuration, start at  $k=0$ , and admit nonzero  $k$  only when

predicted improvement exceeds a noise margin. It should revert when acceptance or measured cycle time violates the estimate. Cache entries must carry a compatibility version and be rejected on any mismatch. Future evaluation should test a larger target with a cheaper draft, fused or multi-token proposal kernels, long-context models with a validated cache ABI, and continuous batching with measured queue state. A positive result should report gains over both AR and KV-only, selected- $k$  distributions, action regret, memory pressure, and decoding equivalence so that a favorable prefill slice cannot hide speculation overhead.

## V. CONCLUSION

RAKV-ASD joins exact prefix reuse with a measured controller that can choose  $k=0$ . Across 3,600 runs it preserved every target output, avoided forced- $k4$  overhead, and remained near AR and KV-only. The tested stack requires predicted acceptance  $\geq 0.894$  for nonzero speculation; even the larger pair approached but did not cross that boundary. Joint selection should therefore treat no speculation as a first-class action, recalibrate after system changes, and enable drafting only when measured costs support it.

## REFERENCES

- [1] A. Vaswani et al., "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, vol. 30, pp. 5998-6008, 2017.
- [2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Technical Report*, 2019.
- [3] G.-I. Yu, J. S. Jeong, G.-W. Kim, S. Kim, and B.-G. Chun, "Orca: A distributed serving system for Transformer-based generative models," in *Proc. 16th USENIX OSDI*, pp. 521-538, 2022.
- [4] W. Kwon et al., "Efficient memory management for large language model serving with PagedAttention," in *Proc. 29th ACM SOSP*, pp. 611-626, 2023, doi:10.1145/3600006.3613165.
- [5] L. Zheng et al., "SGLang: Efficient execution of structured language model programs," *Adv. Neural Inf. Process. Syst.*, vol. 37, pp. 62557-62583, 2024, doi:10.52202/079017-2000.
- [6] I. Gim, G. Chen, S.-S. Lee, N. Sarda, A. Khandelwal, and L. Zhong, "Prompt Cache: Modular attention reuse for low-latency inference," *Proc. Mach. Learn. Syst.*, vol. 6, 2024.
- [7] J. Juravsky, B. Brown, R. Ehrlich, D. Y. Fu, C. Re, and A. Mirhoseini, "Hydragen: High-throughput LLM inference with shared prefixes," *arXiv preprint*, 2024, arXiv:2402.05099.
- [8] J. Yao et al., "CacheBlend: Fast large language model serving for RAG with cached knowledge fusion," in *Proc. 20th ACM EuroSys*, pp. 94-109, 2025, doi:10.1145/3689031.3696098.
- [9] Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from Transformers via speculative decoding," in *Proc. 40th ICML, PMLR 202*, pp. 19274-19286, 2023.
- [10] C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, and J. Jumper, "Accelerating large language model decoding with speculative sampling," *arXiv preprint*, 2023, arXiv:2302.01318.
- [11] Y. Li, F. Wei, C. Zhang, and H. Zhang, "EAGLE: Speculative sampling requires rethinking feature uncertainty," in *Proc. 41st ICML, PMLR 235*, pp. 28935-28948, 2024.
- [12] X. Miao et al., "SpecInfer: Accelerating generative large language model serving with tree-based speculative inference and verification," in *Proc. 29th ACM ASPLOS*, vol. 3, pp. 932-949, 2024, doi:10.1145/3620666.3651335.
- [13] Y. Zhao et al., "Accelerating large-scale reasoning model inference with sparse self-speculative decoding," *Proc. Mach. Learn. Syst.*, vol. 8, 2026.
- [14] X. Liu, J. Yu, J. Park, I. Stoica, and A. Cheung, "Speculative decoding: Performance or illusion?," *Proc. Mach. Learn. Syst.*, vol. 8, 2026.
- [15] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," in *Proc. 5th Int. Conf. Learn. Representations (ICLR)*, 2017.
- [16] M. Chen et al., "Evaluating large language models trained on code," *arXiv preprint*, 2021, arXiv:2107.03374.