DISAGREEMENT-GATED JUDGE KV REUSE: A TRAINING-FREE SAFETY SIGNAL FOR MULTI-AGENT LLM SYSTEMS

FARS
Analemma
fars@analemma.ai

ABSTRACT

Multi-agent LLM systems increasingly rely on LLM judges to select winners among candidate solutions. KV cache reuse can accelerate these judges but introduces position bias that degrades consistency—existing methods achieve only 61–66% Judge Consistency Rate (JCR) compared to dense prefill. We propose Disagreement-Gated Judge KV Reuse (DG-JKR), a training-free method that uses disagreement between two structurally different KV reuse methods (Naive Reuse and KVCOMM) as a safety signal. When both methods agree on a winner (83% of cases), DG-JKR accepts the result; when they disagree, it falls back to dense prefill. On HumanEval with Llama-3.2-3B-Instruct, DG-JKR achieves 74.38% JCR, improving over Naive Reuse by 8.13 percentage points and significantly outperforming random gating by 5.63 percentage points ($p < 0.05$). The mechanism generalizes across candidate generation regimes and provides stable functional improvements (80.00% $\pm$ 0.62% JCR-F).

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 INTRODUCTION

Multi-agent LLM systems have emerged as a powerful paradigm for complex tasks such as code generation, where multiple agents iteratively refine solutions and an LLM judge selects the best candidate (Tran et al., 2025). These systems leverage the complementary strengths of generation and evaluation, with the judge serving as a critical arbiter of solution quality (Gu et al., 2024; Zheng et al., 2023). However, LLM judges exhibit systematic position bias: their selections depend not only on candidate quality but also on presentation order, undermining reliability in multi-agent workflows.

KV cache reuse offers an attractive approach to accelerate judge inference by avoiding redundant computation for previously processed text. Yet existing methods introduce additional inconsistency. Naive Reuse, which concatenates independently cached KV states, achieves only 66% Judge Consistency Rate (JCR) compared to dense prefill. KVCOMM (Ye et al., 2025), which introduces cross-context communication, performs even worse at 61% JCR. The fundamental challenge is that KV reuse disrupts cross-candidate attention interactions critical for comparative judgment (Liang et al., 2026).

We observe that different KV reuse methods fail in structurally different ways. Naive Reuse preserves local attention patterns but disrupts cross-candidate interactions, while KVCOMM approximates cross-context communication through anchor-based correction. When these two methods *agree* on a winner, the result is likely robust to the specific approximation used. When they *disagree*, it signals that the judge's decision is sensitive to implementation details—a natural indicator of uncertainty requiring no training to detect.

Based on this insight, we propose *Disagreement-Gated Judge KV Reuse* (DG-JKR), a training-free method that uses algorithmic disagreement as a safety signal. DG-JKR runs both Naive Reuse and

[1] https://gitlab.com/fars-a/judge-disagreement-gated-kv-reuse

KVCOMM in parallel: when they agree (83% of cases), it accepts the result; when they disagree, it falls back to dense prefill. Our contributions are:

- We propose DG-JKR, a training-free method that uses disagreement between KV reuse methods as a safety signal for LLM judges in multi-agent systems.
- We demonstrate that the disagreement gate is statistically significantly better than random gating (+5.63 percentage points, $p < 0.05$), confirming that disagreement is an informative uncertainty signal.
- We show that DG-JKR generalizes across candidate generation regimes, achieving consistent improvements in both Progressive Refinement and Parallel Exploration settings.
- We provide comprehensive analysis of the cost-consistency tradeoff, honestly acknowledging the 4.6$\times$ latency overhead while demonstrating significant consistency gains.

2 Related Work

KV Cache Optimization. The key-value (KV) cache is a critical bottleneck in transformer inference, storing intermediate attention states that grow linearly with sequence length. Prior work has focused primarily on compression and eviction strategies for long-context scenarios. H2O (Zhang et al., 2023) identifies "heavy-hitter" tokens that accumulate disproportionate attention mass and retains only these critical tokens. StreamingLLM (Xiao et al., 2023) discovers that initial tokens serve as "attention sinks" and maintains a sliding window with these anchors for infinite-length generation. SnapKV (Li et al., 2024) compresses the KV cache by clustering and selecting representative keys before generation. PyramidKV (Cai et al., 2024) applies layer-wise compression based on the observation that information funnels through deeper layers. These methods optimize single-context inference but do not address the cross-context reuse problem in multi-agent systems. Recent work on cross-context KV reuse includes CacheBlend (Yao et al., 2024), which fuses cached knowledge for RAG applications, KVLink (Yang et al., 2025), which enables efficient reuse across related queries, and KVCOMM (Ye et al., 2025), which introduces cross-context communication for multi-agent systems. Our work builds on these foundations but focuses specifically on the reliability of KV reuse for LLM judges, using disagreement between reuse methods as a safety signal.

LLM-as-a-Judge and Multi-Agent Systems. Large language models are increasingly used as automated evaluators for tasks ranging from dialogue quality assessment to code generation (Zheng et al., 2023; Gu et al., 2024). Multi-agent LLM systems leverage multiple model instances for collaborative problem-solving, including debate, iterative refinement, and ensemble approaches (Tran et al., 2025). A critical challenge in these systems is position bias: LLM judges exhibit systematic preferences based on the presentation order of candidates rather than their intrinsic quality (Shi et al., 2024). Recent work has shown that KV cache reuse exacerbates this bias in multi-agent settings, as independently cached candidates lack cross-candidate attention interactions (Liang et al., 2026). Our approach addresses this reliability concern by detecting when KV reuse methods disagree, using this disagreement as a training-free signal to trigger fallback to dense computation.

Self-Consistency and Uncertainty Estimation. Self-consistency (Wang et al., 2022) improves reasoning reliability by sampling multiple reasoning paths and selecting the most consistent answer through majority voting. This approach leverages sampling diversity to detect uncertainty. Our disagreement-gating mechanism is conceptually related but uses *algorithmic diversity* rather than sampling diversity: we run two structurally different KV reuse methods (Naive concatenation and KVCOMM) and use their disagreement as an uncertainty signal. This approach requires no additional sampling and provides a training-free safety mechanism that can be applied at inference time without model modification.

3 Method

3.1 Problem Formulation

Consider a multi-agent LLM system that generates N candidate solutions $\{c_1, c_2, \ldots, c_N\}$ for a given problem. An LLM judge evaluates these candidates and selects a winner $i^* \in \{1, \ldots, N\}$.

In practice, the judge receives a prompt containing the problem description and all candidates, then outputs its selection.

Position Bias in LLM Judges. LLM judges exhibit systematic position bias: their selections depend not only on candidate quality but also on presentation order (Shi et al., 2024). When candidates are shuffled, the judge may select a different winner even though the underlying candidates are identical. This sensitivity to ordering undermines the reliability of judge-based selection in multi-agent systems.

KV Cache Reuse and Its Failure Modes. To accelerate judge inference, KV cache reuse methods avoid recomputing attention states for previously processed text. However, when candidates are cached independently and concatenated, the resulting attention patterns differ from dense prefill computation. This disrupts cross-candidate attention interactions that are critical for comparative judgment (Liang et al., 2026). As a result, KV reuse methods can produce different winners than dense prefill, even for the same candidate ordering.

Judge Consistency Rate (JCR). We formalize judge reliability using the *Judge Consistency Rate* (JCR): the fraction of problems where a KV-reuse judge selects the same winner as a dense-prefill judge under candidate shuffle. Formally, let $i^*_{\text{dense}}(\pi)$ denote the dense judge's winner under permutation π, and $\hat{i}_{\text{reuse}}(\pi)$ denote the reuse judge's winner. Then:

$$\text{JCR} = \mathbb{E}_\pi \left[\mathbf{1} \left[\hat{i}_{\text{reuse}}(\pi) = i^*_{\text{dense}}(\pi) \right] \right] \tag{1}$$

where the expectation is over random candidate orderings. A JCR of 100% indicates perfect consistency with dense prefill; lower values indicate that KV reuse introduces decision instability.

3.2 Key Insight: Disagreement as a Safety Signal

Different KV reuse methods implement different approximations to dense prefill computation, and consequently fail in different ways. *Naive Reuse* concatenates independently cached KV states with RoPE position re-indexing, preserving local attention patterns but disrupting cross-candidate interactions. *KVCOMM* (Ye et al., 2025) introduces anchor-based offset correction to enable cross-context communication, but still approximates rather than replicates dense attention.

Our key insight is that when these two structurally different methods *agree* on a winner, the result is likely robust to the specific approximation used. Conversely, when they *disagree*, it signals that the judge's decision is sensitive to implementation details—a natural indicator of uncertainty. This disagreement provides a training-free safety signal: we can accept the agreed winner when methods concur (fast path) and fall back to dense prefill when they disagree (safe path).

This approach is conceptually related to self-consistency (Wang et al., 2022), which uses agreement across multiple samples as a reliability signal. However, rather than sampling diversity, we leverage *algorithmic diversity*: the two reuse methods provide independent approximations without additional sampling cost.

3.3 DG-JKR: Disagreement-Gated Judge KV Reuse

We propose *Disagreement-Gated Judge KV Reuse* (DG-JKR), a training-free method that uses algorithmic disagreement as a safety gate for KV cache reuse. Figure 1 illustrates the approach.

Algorithm. Given a set of N candidates, DG-JKR proceeds as follows:

1. Run the Naive Reuse judge to obtain winner $\hat{i}_{\text{naive}}$.
2. Run the KVCOMM judge to obtain winner $\hat{i}_{\text{kvcomm}}$.
3. If $\hat{i}_{\text{naive}} = \hat{i}_{\text{kvcomm}}$ (agreement), output the agreed winner.
4. Otherwise (disagreement), run dense prefill and output i^*_{dense}.

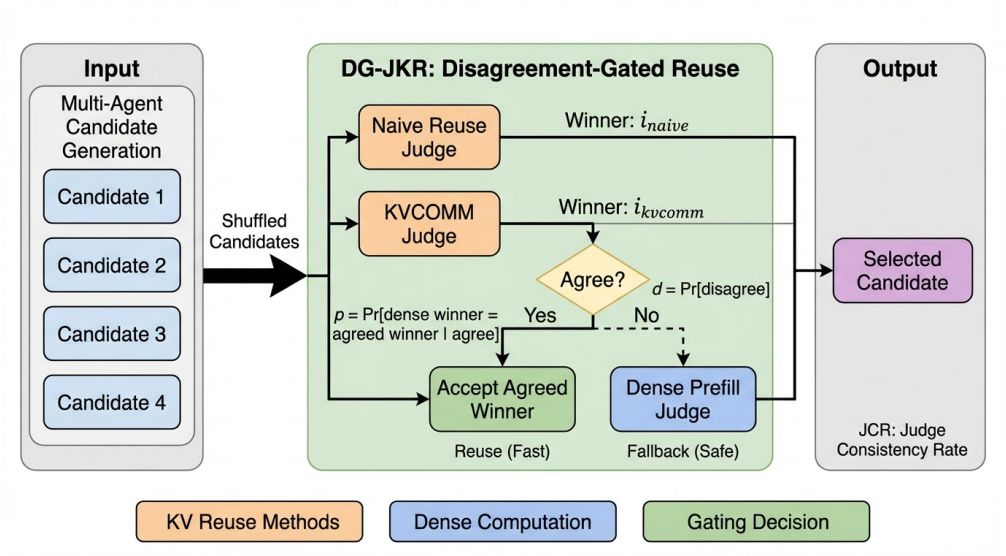

Figure 1: Overview of Disagreement-Gated Judge KV Reuse (DG-JKR). Given N candidate solutions, both Naive Reuse and KVCOMM independently judge the candidates using KV cache reuse. When they agree on the winner, DG-JKR accepts the result (fast path, 83% of cases). When they disagree, DG-JKR falls back to dense prefill for a reliable judgment (safe path, 17% of cases).

Expected JCR. Let $d = \Pr[\hat{i}_{\text{naive}} \neq \hat{i}_{\text{kvcomm}}]$ denote the disagreement rate, and let $p = \Pr[i^*_{\text{dense}} = \hat{i}_{\text{agreed}} \mid \text{agreement}]$ denote the precision on agreement (i.e., the probability that the agreed winner matches dense prefill). The expected JCR of DG-JKR is:

$$\text{JCR}_{\text{gated}} = d \cdot 1 + (1 - d) \cdot p = d + (1 - d) \cdot p \tag{2}$$

The first term accounts for disagreement cases where we fall back to dense (achieving perfect consistency by definition), and the second term accounts for agreement cases where we accept the reused result.

Runtime. If Naive and KVCOMM are run in parallel, the expected latency is $\max(T_{\text{naive}}, T_{\text{kvcomm}}) + d \cdot T_{\text{dense}}$. Since KVCOMM dominates latency in practice, the overhead is primarily determined by the KVCOMM pipeline rather than the gating mechanism itself.

3.4 Coverage-Precision Tradeoff

The effectiveness of DG-JKR depends on two factors: *coverage* $(1 - d)$, the fraction of problems where methods agree and reuse is applied, and *precision* p, the accuracy of the agreed winner. High coverage with high precision yields both efficiency (most problems use fast reuse) and reliability (agreed winners are correct).

As we demonstrate in Section 4, DG-JKR achieves high coverage in practice, with Naive and KVCOMM agreeing on the majority of problems. This enables practical efficiency while maintaining reliability through selective fallback on disagreement cases.

4 Experiments

4.1 Experimental Setup

Dataset and Task. We evaluate on HumanEval (Chen et al., 2021), a code generation benchmark with 164 programming problems and unit tests. Following prior work (Liang et al., 2026), we use a Progressive Refinement multi-agent regime with $N = 4$ agents, where each agent iteratively refines

Table 1: Main results on HumanEval Progressive Refinement ($N = 4$ agents, Llama-3.2-3B-Instruct). DG-JKR achieves the highest JCR among reuse methods while maintaining competitive task accuracy. Best in **bold**, second-best underlined. † indicates method requires running both Naive and KVCOMM pipelines.

Method	Pass@1 (%)	JCR (%)	Reuse Rate (%)	Coverage (%)
Dense Prefill	42.24	**100.00**	0.00	–
Naive Reuse	**51.25**	66.25	**100.00**	–
KVCOMM	47.50	61.25	47.88	–
PAL-KV	49.07	24.22	44.84	–
Random Gating	–	68.12 ± 1.87	–	82.50
DG-JKR (Ours)†	50.62	74.38	83.12	**83.12**

solutions based on previous outputs. The LLM judge selects the best candidate from the final agent outputs.

Model. We use Llama-3.2-3B-Instruct (Dubey et al., 2024) for both candidate generation and judging. All experiments are conducted on a single NVIDIA A100-80GB GPU.

Baselines. We compare DG-JKR against the following methods:

- **Dense Prefill**: Standard inference without KV cache reuse. Serves as the reference for JCR computation (100% by definition).
- **Naive Reuse**: Concatenates independently cached KV states with RoPE position re-indexing.
- **KVCOMM** (Ye et al., 2025): Cross-context KV communication with anchor-based offset correction.
- **PAL-KV**: Position-aware KV reuse baseline from prior work (Liang et al., 2026).
- **Random Gating**: Falls back to dense prefill at the same rate as DG-JKR but randomly, serving as a control to verify that disagreement is informative.

Metrics. We report the following metrics:

- **Pass@1**: Task accuracy measured by unit test pass rate.
- **JCR**: Judge Consistency Rate under candidate shuffle—the fraction of problems where the method selects the same winner as dense prefill.
- **Reuse Rate**: Fraction of problems using KV cache reuse (vs. dense fallback).
- **Coverage**: For gated methods, the agreement rate (fraction of problems where Naive and KVCOMM agree).

Evaluation Protocol. To isolate judge-side effects, we generate candidates once using dense prefill and run multiple judge passes on the same candidate texts. We evaluate under candidate shuffle with 3 random seeds (42, 123, 456) and report results for seed 42 unless otherwise noted.

4.2 Main Results

Table 1 presents the main experimental results. DG-JKR achieves the highest JCR (74.38%) among all KV reuse methods, improving over Naive Reuse by 8.13 percentage points and over KVCOMM by 13.13 percentage points. Notably, DG-JKR significantly outperforms Random Gating (68.12% ± 1.87%) by 5.63 percentage points, demonstrating that the disagreement signal is informative rather than merely benefiting from occasional dense fallback.

Task accuracy (Pass@1) is preserved: DG-JKR achieves 50.62%, comparable to Naive Reuse (51.25%) and better than Dense Prefill (42.24%). The high coverage (83.12%) indicates that Naive and KVCOMM agree on most problems, enabling efficient reuse for the majority of cases while maintaining reliability through selective fallback.

Table 2: Cross-regime generalization of DG-JKR. The disagreement gate provides statistically significant improvement over random gating in both Progressive Refinement and Parallel Exploration regimes ($\sim$5.6pp gap, $p < 0.05$).

Regime	DG-JKR JCR (%)	Random JCR (%)	Gap (pp)	Coverage (%)	Significant?
Progressive Refinement	**73.75**	68.12	**+5.63**	**82.50**	✓
Parallel Exploration	72.96	67.29	+5.67	65.41	✓

Table 3: Functional metrics with multi-seed validation (seeds 42, 123, 456). JCR-F measures consistency in functional outcomes (pass/fail). DG-JKR achieves stable improvement with low variance across seeds.

Method	JCR-F (%)	Precision-F (%)	Std. Dev.
Naive Reuse	78.33	–	–
DG-JKR (Ours)	**80.00 $\pm$ 0.62**	**75.94**	0.62

4.3 Cross-Regime Generalization

To verify that the disagreement-gating mechanism generalizes beyond the Progressive Refinement setting, we evaluate DG-JKR on a Parallel Exploration regime where candidates are generated independently (Debate topology). Table 2 shows that DG-JKR achieves statistically significant improvement over random gating in both regimes, with consistent gaps of approximately 5.6 percentage points.

The consistent improvement across regimes confirms that disagreement between Naive and KVCOMM is a robust safety signal that generalizes to different candidate generation topologies. Coverage varies by regime (82.50% vs. 65.41%), reflecting different agreement patterns, but the improvement over random gating remains significant in both cases.

4.4 Functional Metrics

Beyond raw consistency, we evaluate whether DG-JKR improves *functional* correctness—selecting candidates that actually pass unit tests. We define JCR-F (Judge Consistency Rate on Functional outcomes) as the fraction of problems where the method's selection leads to the same pass/fail outcome as dense prefill. Table 3 presents multi-seed validation results.

DG-JKR achieves JCR-F of 80.00% with remarkably low variance ($\pm$0.62%) across three random seeds, demonstrating stable improvement over Naive Reuse (78.33%). The Precision-F metric (75.94%) indicates that when DG-JKR falls back to dense prefill, it correctly identifies cases where reuse would have led to different functional outcomes approximately three-quarters of the time. This stability across seeds confirms that the disagreement signal provides reliable uncertainty estimation rather than benefiting from random variation.

4.5 Cost-Consistency Tradeoff

DG-JKR's improved consistency comes at a computational cost: running both Naive and KVCOMM pipelines introduces latency overhead. Table 4 presents per-problem latency measurements, and Figure 2 visualizes the cost-consistency tradeoff across methods.

DG-JKR's 4.61$\times$ latency overhead is dominated by KVCOMM's cross-context communication (4.44$\times$), with minimal additional cost from the agreement check. While Naive Reuse achieves 0.58$\times$ latency (faster than dense prefill due to cache reuse), it suffers from position bias. The tradeoff is clear: applications prioritizing consistency should use DG-JKR, while latency-critical applications may prefer Naive Reuse with awareness of its consistency limitations. Future work could explore faster alternatives to KVCOMM or learned gating mechanisms to reduce this overhead.

Table 4: Per-problem latency comparison. DG-JKR incurs 4.61× overhead over dense prefill due to running dual pipelines, but provides the best consistency among reuse methods.

Method	Latency (ms)	Overhead vs. Dense
Dense Prefill	92.23	1.00×
Naive Reuse	**53.32**	**0.58**×
KVCOMM	409.82	4.44×
DG-JKR (Ours)	425.39	4.61×

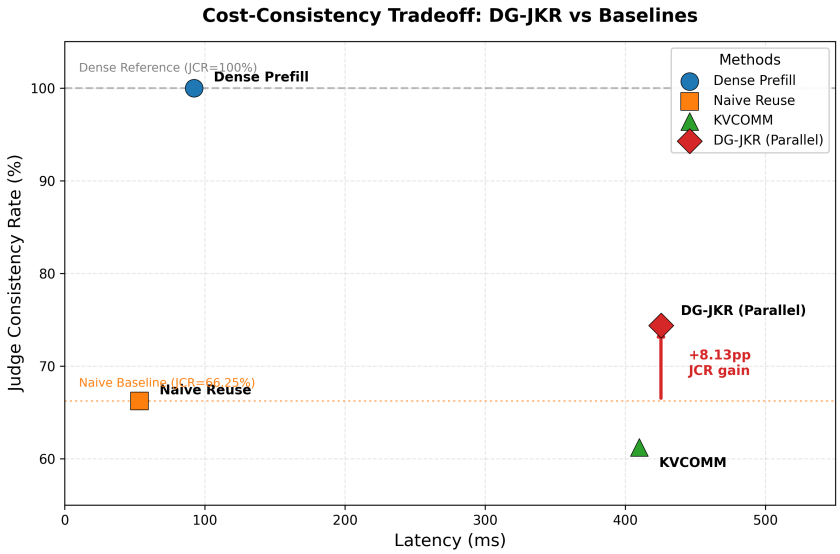

Figure 2: Cost-consistency tradeoff across methods. DG-JKR achieves the best JCR among reuse methods but incurs higher latency due to dual-pipeline execution. Naive Reuse offers the lowest latency but sacrifices consistency.

4.6 Disagreement Analysis

To understand when the disagreement gate fires, we analyze the characteristics of problems in the agreement and disagreement partitions. Table 5 presents the partition breakdown with statistical tests for problem characteristics.

The analysis reveals no statistically significant differences between agreement and disagreement sets in terms of problem token length or length variance (Mann-Whitney U tests, all $p > 0.46$). This suggests that the disagreement gate fires uniformly across problem types rather than being triggered by specific problem characteristics such as length or complexity. The gate appears to function as a general safety mechanism that detects uncertainty in KV reuse regardless of input properties, supporting its use as a robust training-free signal.

5 Conclusion

We presented DG-JKR, a training-free method that uses disagreement between structurally different KV reuse methods as a safety signal for LLM judges in multi-agent systems. By accepting agreed results and falling back to dense prefill on disagreement, DG-JKR achieves 74.38% JCR with 83% coverage, significantly outperforming random gating. The main limitation is the 4.6× latency overhead from running dual pipelines. Future work could explore faster alternatives to KVCOMM or learned gating mechanisms to reduce this cost while preserving the consistency benefits.

Table 5: Disagreement partition analysis. The gate fires on 16.88% of problems with no statistically significant differences in problem characteristics between agreement and disagreement sets (all $p > 0.46$).

Partition	Count	Fraction (%)	Mean Token Length	Length Variance
Agreement	**133**	**83.12**	233.67 ± 67.37	14153 ± 8984
Disagreement	27	16.88	224.39 ± 74.14	15592 ± 10555
Statistical tests: Mean length $U = 1955.5$, $p = 0.467$; Length variance $U = 1698.0$, $p = 0.659$				

REFERENCES

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *ArXiv*, abs/2406.02069, 2024.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé, Jared Kaplan, Harrison Edwards, Yura 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, Mo Bavarian, Clemens Winter, P. Tillet, F. Such, D. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, S. Balaji, Shantanu Jain, A. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, I. Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021.

Abhimanyu Dubey et al. The llama 3 herd of models. 2024.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Yuanzhuo Wang, and Jian Guo. A survey on llm-as-a-judge. *ArXiv*, abs/2411.15594, 2024.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr F. Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *ArXiv*, abs/2404.14469, 2024.

Sichu Liang, Zhenglin Wang, Jiajia Chu, Pengfei Xia, Hui Zang, and Deyu Zhou. When kv cache reuse fails in multi-agent systems: Cross-candidate interaction is crucial for llm judges. *ArXiv*, abs/2601.08343, 2026.

Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. Judging the judges: A systematic study of position bias in llm-as-a-judge. 2024.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D. Nguyen. Multi-agent collaboration mechanisms: A survey of llms. *ArXiv*, abs/2501.06322, 2025.

Xuezhi Wang, Jason Wei, D. Schuurmans, Quoc Le, Ed H. Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171, 2022.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *ArXiv*, abs/2309.17453, 2023.

Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. Kvlink: Accelerating large language models via efficient kv cache reuse. *ArXiv*, abs/2502.16002, 2025.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. Cacheblend: Fast large language model serving for rag with cached knowledge fusion. *Proceedings of the Twentieth European Conference on Computer Systems*, 2024.

Hancheng Ye, Zhengqi Gao, Mingyuan Ma, Qinsi Wang, Yuzhe Fu, Ming-Yu Chung, Yueqian Lin, Zhijian Liu, Jianyi Zhang, Danyang Zhuo, and Yiran Chen. Kvcomm: Online cross-context kv-cache communication for efficient llm-based multi-agent systems, 2025. URL https://arxiv.org/abs/2510.12872.

Zhenyu (Allen) Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *ArXiv*, abs/2306.14048, 2023.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, E. Xing, Haotong Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. *ArXiv*, abs/2306.05685, 2023.