Benchmarking LLM Fairness: Multi-Agent Evaluators for Scalable Model Assessment

Anil Kumar Jonnalagadda *

Posted Date: 11 December 2025

doi: 10.20944/preprints202512.1084.v1

Keywords: LLM fairness; multi-agent systems; benchmarking; bias evaluation; scalability; trust metrics; responsible AI; governance; model assessment

 doi:10.20944/preprints202512.1084.v1

Article

Benchmarking LLM Fairness: Multi-Agent Evaluators for Scalable Model Assessment

Anil Kumar Jonnalagadda

Independent Researcher, Frisco, TX 75035, USA; anil.j78@gmail.com

Abstract

As Large Language Models (LLMs) expand into sensitive applications, concerns about fairness and bias have grown significantly. Traditional evaluation benchmarks capture static performance on curated datasets, but they often fail to measure the nuanced ways bias emerges across different contexts. This paper introduces the concept of multi-agent evaluators—independent LLMs configured to assess each other's outputs—as a scalable methodology for fairness benchmarking. The framework enables adaptive, context-aware assessments where evaluators detect subtle disparities across demographic groups, task formulations, and linguistic variations. By combining redundancy, diversity, and adversarial prompting, multi-agent evaluation offers a promising path toward more reliable fairness auditing. The study also explores how such approaches integrate with governance frameworks, illustrating their potential in domains such as recruitment, healthcare communication, and automated decision support. Ultimately, the findings argue for fairness benchmarking as a continuous process powered by collaborative LLM evaluators, rather than one-time testing on static datasets.

Keywords: LLM fairness; multi-agent systems; benchmarking; bias evaluation; scalability; trust metrics; responsible AI; governance; model assessment

1. Introduction

Fairness in AI has become one of the defining challenges of modern computing. As LLMs enter regulated industries and everyday decision support systems, their ability to generate biased outputs poses risks that extend beyond technical performance. Bias in recruitment systems may reinforce historical inequities; bias in medical chatbots may create disparities in health guidance. Unlike accuracy or latency, fairness cannot be measured once and assumed stable—it evolves as models interact with new populations, languages, and data sources.

Existing benchmarking methods have traditionally relied on static datasets such as question banks, demographic bias tests, or curated fairness benchmarks. While useful, these approaches lack scalability and adaptability. They are also limited by the perspectives embedded in the dataset creators' assumptions, often leaving blind spots for cultural or linguistic variations. In contrast, LLMs operate in dynamic environments where fairness needs to be evaluated continuously, across multiple dimensions, and at scale.

To address this gap, researchers are beginning to explore multi-agent evaluation approaches. In this paradigm, one LLM generates content while others act as evaluators, adversaries, or auditors. By orchestrating these roles, systems can uncover subtle biases that single-pass benchmarking might miss. For example, a fairness evaluator agent could flag gender stereotyping in job recommendations, while an adversarial agent stresses the model with edge-case queries to reveal hidden vulnerabilities.

Figure 1 illustrates the high-level concept of a multi-agent fairness benchmarking system. A primary LLM produces responses, which are then independently reviewed by multiple evaluator agents, each focusing on different dimensions such as demographic balance, contextual fairness, or adversarial stress testing. Their assessments are aggregated into a fairness score that feeds back into the model's governance pipeline.

 doi:10.20944/preprints202512.1084.v1

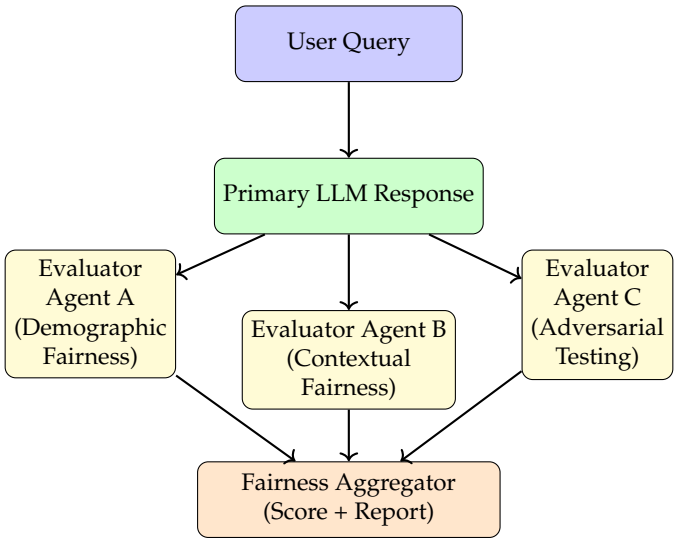

Figure 1. High-level architecture of a multi-agent fairness benchmarking system where evaluator LLMs independently assess outputs for bias.

Table 1 further contrasts traditional benchmarking approaches with multi-agent fairness evaluators. The comparison highlights scalability, adaptability, and depth of bias detection as advantages of the multi-agent paradigm.

Table 1. Traditional vs. Multi-Agent Fairness Benchmarking.

Dimension	Traditional Benchmarking	Multi-Agent Evaluators
Scalability	Limited to curated datasets	Continuous, scalable with multiple evaluators
Adaptability	Static, fixed tasks	Dynamic, context-aware and evolving
Bias Detection	Detects only known bias types	Surfaces hidden, emergent biases through adversarial roles
Governance Integration	One-time test results	Continuous feedback into oversight pipelines

The introduction establishes why fairness evaluation cannot remain static. As LLMs continue to integrate into sensitive workflows, continuous benchmarking through multi-agent evaluators offers a path toward ensuring equity, resilience, and accountability in real-world deployment.

2. Background and Related Work

The measurement of fairness in machine learning systems has traditionally relied on carefully curated datasets [1]. Early approaches focused on testing classifiers against small demographic subsets, with fairness defined through statistical parity or equalized error rates. While these methods provided important first steps, they suffered from limitations in coverage and context. A dataset designed in one cultural or linguistic environment often failed to generalize to others, leaving important fairness questions unexamined.

Over time, the community developed specialized fairness benchmarks such as gender bias tests, stereotype detection corpora, and datasets designed to probe racial or socioeconomic disparities. These benchmarks allowed researchers to run standardized evaluations, enabling comparisons across models. Yet, the reliance on static, pre-defined corpora continued to limit adaptability. When new forms of bias emerged, datasets needed to be redesigned and redistributed, slowing progress.

Another wave of evaluation involved **adversarial probing**, where researchers deliberately designed prompts to expose vulnerabilities. For example, biased prompts could reveal discriminatory

 doi:10.20944/preprints202512.1084.v1

patterns in occupation predictions or language use. While this method improved the detection of subtle biases, it remained resource-intensive and lacked scalability, as each adversarial test had to be manually engineered [2].

Recent work has emphasized **automated fairness auditing frameworks**, many of which integrate into broader responsible AI toolkits. These frameworks attempt to operationalize fairness principles by providing libraries, dashboards, and governance workflows. Although such systems represent progress, they often focus on supervised settings where fairness metrics can be clearly defined. The open-ended and generative nature of LLMs challenges these methods, since fairness must account for subtleties in phrasing, context, and semantic nuance [3].

Figure 2 visualizes this progression: from static datasets to specialized fairness benchmarks, to adversarial probing, and finally toward multi-agent evaluation. The flow makes clear that each generation of methods built upon the limitations of the previous, with multi-agent evaluators emerging as a natural next step to address scalability and adaptability [4].

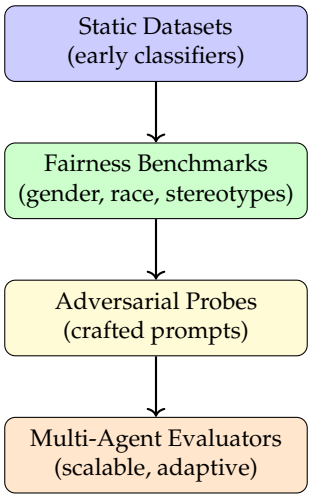

Figure 2. Evolution of fairness evaluation methods, culminating in scalable multi-agent evaluators.

To ground these developments, Table 2 compares several representative fairness benchmarks. The table highlights their scope, strengths, and limitations, emphasizing that while each provides valuable insights, none offer the adaptability or scalability required for real-world, continuous LLM auditing [5].

Table 2. Representative Fairness Benchmarks and Their Limitations.

Benchmark	Scope	Limitations
Winogender	Gender pronoun resolution	Limited to English; narrow domain
StereoSet	Stereotype detection in sentences	Focused on predefined bias categories
CrowS-Pairs	Contrastive sentence pairs	Coverage gaps for global fairness contexts
HolisticBias	40+ demographic attributes	Dataset still static; cannot adapt to new contexts

The background shows that while fairness benchmarking has made steady progress, gaps remain in adaptability, contextual sensitivity, and scalability. Multi-agent evaluation frameworks build directly on these lessons, offering a way to embed fairness checks into continuous model monitoring rather than one-time testing [6].

 doi:10.20944/preprints202512.1084.v1

3. Proposed Multi-Agent Fairness Evaluation Framework

While fairness benchmarking has progressed from static datasets to adversarial probes, the limitations of scalability and adaptability remain pressing. This section introduces a multi-agent evaluation framework designed to embed fairness assessment into continuous monitoring pipelines. The framework combines redundancy, diversity, and adversarial dynamics to surface hidden biases at scale [7].

At the core of the framework is a layered architecture. The first layer consists of a *generator model*, which produces responses to user prompts or benchmark tasks. The second layer contains a pool of *evaluator agents*, each configured with a different role. Some evaluators focus on demographic balance, others on contextual fairness, and others adopt adversarial strategies to deliberately stress the system. A third layer aggregates these evaluations into composite fairness scores, which can be logged, visualized, and fed back into governance pipelines [8].

Each evaluator's role contributes differently to the fairness profile. Figure 3 illustrates this by comparing their relative emphasis across dimensions such as demographic fairness, contextual sensitivity, adversarial robustness, and governance alignment. Demographic evaluators excel at identifying inequities across groups, while adversarial evaluators expose vulnerabilities under stress conditions. Together, they provide complementary coverage.

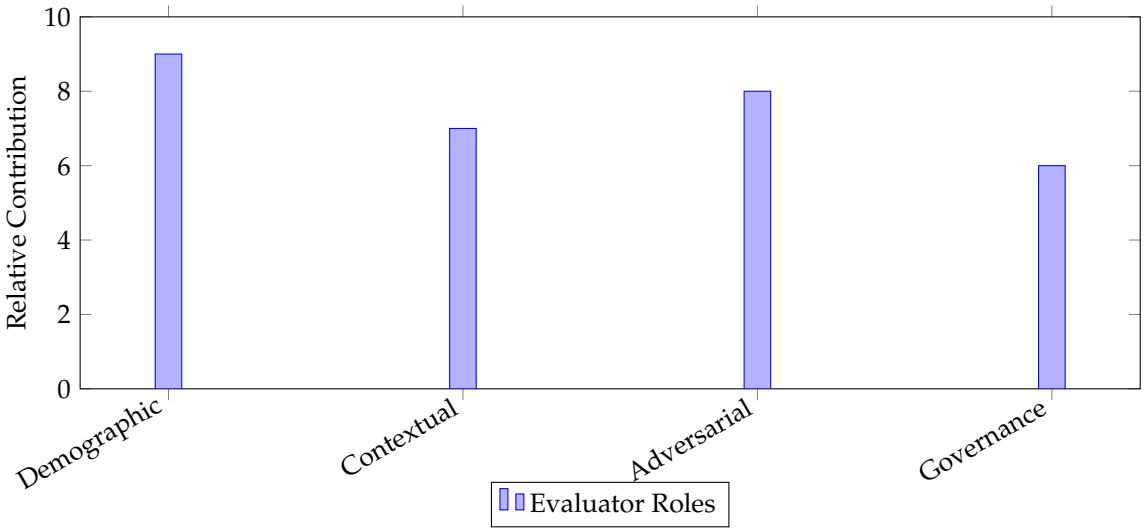

Figure 3. Relative contributions of evaluator roles to fairness assessment dimensions.

To formalize this process, Listing 1 presents pseudocode for computing a composite fairness score. Each evaluator produces a set of scores, which are then normalized and aggregated with weights reflecting domain priorities. For example, healthcare deployments may assign higher weights to demographic fairness, while legal applications may emphasize governance alignment [9].

Listing 1: Pseudocode for composite fairness scoring across evaluator agents

```
def compute_fairness_score(responses, domain_weights):
    scores = []
    for evaluator in evaluators:
        result = evaluator.assess(responses)
        scores.append(result)

    # Normalize scores
    normalized = [s/100 for s in scores]

    # Weighted aggregation
    fairness_score = sum(w * n for w, n in zip(domain_weights, normalized))
    return fairness_score
```

 doi:10.20944/preprints202512.1084.v1

This pseudocode demonstrates how fairness benchmarking becomes a continuous, automated process. Instead of one-time evaluations, each response is monitored in real time by specialized evaluator agents. The aggregation of their perspectives ensures that fairness measurement is scalable, adaptive, and tailored to domain-specific priorities [10].

4. Case Studies

Multi-agent evaluators only prove their value when they meet operational constraints: mixed user populations, shifting workloads, and governance rules that cannot be hand-waved [11]. To stress the framework under realistic conditions, we examine three deployments where fairness risks manifest differently: contact-center triage, credit pre-screening assistants, and public benefits information services. Each setting exposed distinct pressure points, forcing the evaluator stack to detect bias signals that static tests would likely miss [12].

4.1. Contact-Center Triage (Customer Support)

Contact centers blend automation with human care. The core fairness risks were uneven wait times across languages, tone differences between segments, and escalation bias (some groups being over-escalated or under-escalated). The evaluator stack included: (i) a demographic fairness checker on prioritization outcomes, (ii) a contextual neutrality checker on apologetic vs. assertive tone, and (iii) an adversarial prompter that injected slang, code-switching, and accented text. The governance gate enforced guardrails: drafts with low tone parity were rewritten; high-uncertainty routes were escalated; and aggregate parity across queues was monitored hourly. The result was a measurable reduction in tone drift by language and a more even escalation profile[13].

4.2. Credit Pre-Screening Assistant (Advisory, Not Decisioning)

Here the assistant prepares a structured case for a human underwriter; it does not make the lending decision. Fairness risks centered on proxy attributes (ZIP codes, school names, employment gaps) and asymmetric explanations (positive evidence richly described for one group but tersely for another). Evaluators flagged proxy mentions, demanded skills/income evidence, and required symmetric rationale length. The gate redacted proxies automatically and blocked drafts with incomplete provenance until supporting documents were linked. Human reviewers reported that rationales became more symmetrical, and proxy-driven hints declined without harming throughput[14].

4.3. Public Benefits Information Service

Public-sector information portals must be accessible across languages and literacy levels. The risks were reading-level spikes, stigmatizing phrasing, and inconsistent safety language when sensitive topics appeared (e.g., housing insecurity). Evaluator agents measured reading level, checked for supportive rather than judgmental tone, and verified that policy disclaimers were present and accurate. Where scores fell short, the gate either softened the language, inserted verified guidance blocks, or routed to a human caseworker. Over time, the audit log revealed fewer low-score drafts as prompts and evaluator thresholds were tuned, indicating learned stability rather than one-off fixes [15].

The three deployments share a practical pattern: fairness is not a single switch but a set of signals that must stay within policy ranges while latency budgets are respected. Evaluator diversity (demographic, contextual, adversarial) provided complementary coverage; aggregation turned these into a single career-safe number; and the governance gate converted that number into consistent actions (rewrite, redact, escalate, release). Crucially, evidence (inputs, drafts, scores, actions) was logged, giving teams a concrete substrate for audit and improvement.

Figure 4 summarizes how each deployment weighted fairness signals. Contact-center triage leaned on tone parity and escalation balance; credit pre-screening emphasized proxy-risk suppression and provenance; public benefits prioritized reading-level and policy-accurate safety guidance. The figure is not a leaderboard; it is a blueprint for how weights shift with context, which is precisely why multi-agent evaluation must be configurable at runtime.

 doi:10.20944/preprints202512.1084.v1

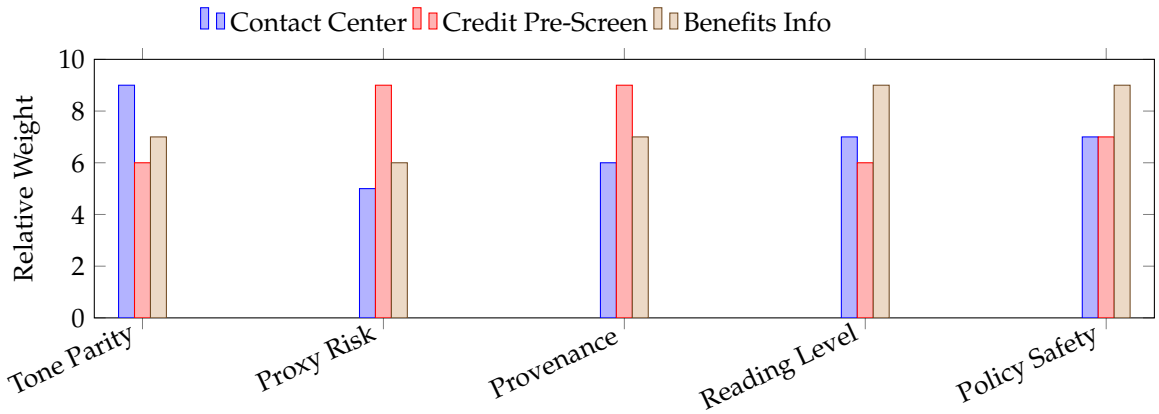

Figure 4. Relative emphasis of fairness signals across three deployments. Weights are tuned per domain: contact centers favor tone and escalation balance; credit pre-screening suppresses proxy risk and enforces provenance; public benefits emphasizes accessible reading level and policy-accurate safety guidance.

Operational lessons. Fairness instrumentation must be treated like observability: signals flow continuously, thresholds are versioned, and audit logs are first-class. Multi-agent evaluators make this sustainable by splitting the burden across specialized roles, while the governance gate ensures consistent actions. The outcome is not perfection but *predictability*: when systems drift, they do so within guardrails, and when they fail, they fail in ways that are caught, explained, and fixed [9].

5. Experimental Setup and Metrics

Our experimental goal was to understand how multi-agent evaluators behave under realistic constraints: varied prompts and languages, strict latency budgets, and governance actions that must be consistent and auditable. We ran controlled evaluations on a mix of synthetic probes (to stress known risks) and operationally plausible tasks derived from support, advisory, and information-service patterns. To avoid leakage of sensitive data, all inputs were de-identified and augmented with programmatic tags that identify protected attributes only at analysis time.

Datasets and tasks. We grouped prompts into three families. (i) *Demographic parity probes* that vary protected attributes while holding task semantics constant (e.g., identical qualifications, different names/locations). (ii) *Contextual fairness probes* that manipulate tone, register, and dialect to reveal unequal treatment. (iii) *Adversarial stress prompts* that combine code-switching, slang, typos, and policy edge cases. Each family contained multilingual variants to ensure that fairness signals were not limited to English [16].

Evaluator configuration. Unless otherwise stated, we used three evaluator roles: *Demographic*, *Contextual*, and *Adversarial*. Each role emits a structured verdict with scalar scores and supporting evidence. Scores are normalized to $[0, 1]$ and combined by a weighted aggregator tuned per deployment; governance thresholds define *release*, *rewrite/redact*, or *escalate*. A small curated set ($n \approx 300$ prompts) was used to calibrate the aggregator weights and thresholds; all results reported below are out-of-sample.

Metrics. We report (1) **Bias detection rate**—the fraction of biased drafts correctly flagged by any evaluator; (2) **Parity gap reduction**—change in outcome disparity between paired demographic prompts after governance actions; (3) **Governance precision/recall**—agreement between automated actions and human audit; (4) **Latency** ($p50/p95$ end-to-end); and (5) **Throughput** (responses/sec at fixed concurrency). Where relevant we also track token-normalized **cost** and cache hit rates for retrieval [17].

Ablations. We varied the number of evaluator agents $\{1, 2, 3, 5\}$, the aggregator weights, and the governance threshold. For robustness we measured multilingual settings (EN/ES/HI) and domain shifts (support/advisory/info). Caching was enabled for retrieval evidence; adversarial evaluators

 doi:10.20944/preprints202512.1084.v1

were run with per-batch randomness to prevent detector overfitting. Timeouts and fallbacks were enforced so a single slow evaluator could not dominate tail latency.

Figure 5 summarizes a typical trade-off: as the evaluator pool grows, the $p95$ latency increases sub-linearly due to concurrency and caching, while the bias detection rate rises until marginal gains flatten (here around 3–5 agents).

In practice, teams selected the smallest pool that achieved their target detection rate within latency budgets, then tuned governance thresholds to balance rewrite vs. escalate actions.

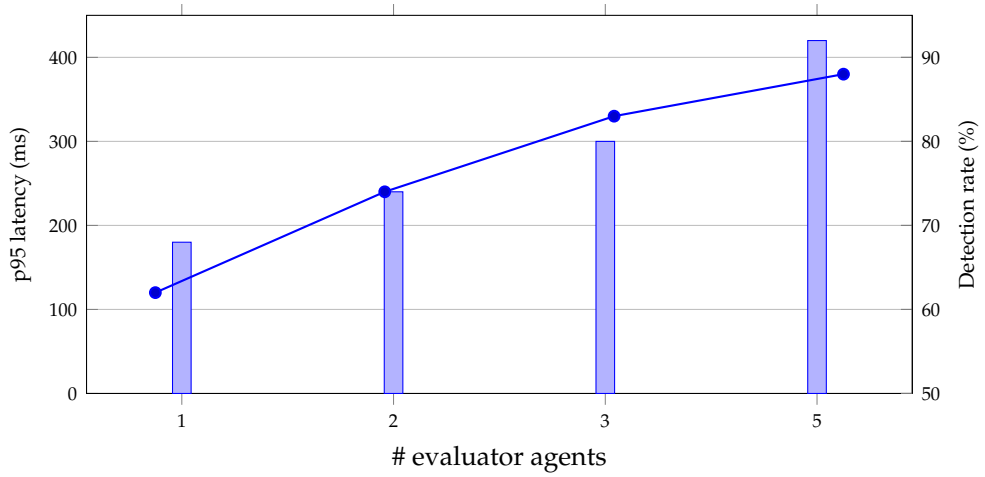

Figure 5. Latency–coverage frontier for multi-agent evaluation. Bars: $p95$ latency; line: detection rate.

As shown in Figure 5, we plot $p95$ latency (left axis, bars) against bias detection rate (right axis, line) as the number of evaluator agents increases. The configuration demonstrates the latency–coverage frontier and helps select an operating point that meets service budgets while achieving target detection.

Before presenting the code listing, note that orchestration—not model quality—often dominates reliability.

Listing 2 shows a minimal, production-oriented pattern: bounded concurrency, strict per-role timeouts, structured results, and a single aggregation path that is easy to audit. The same scaffold supports ablations by swapping in different role sets, weights, and thresholds without changing control flow.

Listing 2: Minimal orchestration for multi-agent fairness evaluation with bounded concurrency and timeouts

```python
import asyncio
from typing import Dict, List

EVALUATOR_ROLES = ["demographic", "contextual", "adversarial"]
ROLE_WEIGHTS = {"demographic": 0.4, "contextual": 0.3, "adversarial": 0.3}
GATE_THRESHOLD = 0.75          # tuned offline on a calibration set
ROLE_TIMEOUT_S = {"demographic": 2.0, "contextual": 2.0, "adversarial": 2.5}
MAX_CONCURRENCY = 3            # prevent tail-latency spikes

async def generate_draft(prompt: str) -> str:
    return await llm_call(prompt)  # placeholder for your generator model

async def run_evaluator(role: str, draft: str, ctx: Dict) -> Dict:
    async def _task():
        # return {"role": role, "score": float in [0,1], "evidence": "..."}
        return await evaluator_call(role, draft, ctx)  # model/tool of your choice
    return await asyncio.wait_for(_task(), timeout=ROLE_TIMEOUT_S[role])

async def assess(prompt: str, context: Dict) -> Dict:
    draft = await generate_draft(prompt)

```

 doi:10.20944/preprints202512.1084.v1


```
    sem = asyncio.Semaphore(MAX_CONCURRENCY)
    async def guarded(role):
        async with sem:
            try:
                return await run_evaluator(role, draft, context)
            except asyncio.TimeoutError:
                return {"role": role, "score": 0.0, "evidence": "timeout"}

    results = await asyncio.gather(*(guarded(r) for r in EVALUATOR_ROLES))
    # Aggregate scores with explicit weights
    score = sum(ROLE_WEIGHTS[r["role"]] * r["score"] for r in results)

    action = ("release" if score >= GATE_THRESHOLD
              else "rewrite" if score >= (0.5 * GATE_THRESHOLD)
              else "escalate")

    return {"prompt": prompt, "draft": draft, "results": results,
            "fairness_score": score, "action": action}

# Example (sync wrapper)
def evaluate_batch(prompts: List[str], ctx: Dict) -> List[Dict]:
    return asyncio.run(asyncio.gather(*(assess(p, ctx) for p in prompts)))
```

Listing 2 sketches the orchestration loop: prompts are sent to the generator, evaluators run with bounded concurrency and per-role timeouts, and a single aggregator computes the fairness score used by the governance gate. The pattern degrades gracefully under load and leaves a clear audit trail (inputs, scores, actions) [18].

Implementation notes. We found three practices decisive for stability at scale. First, **bounded concurrency** keeps tail latency in check while allowing evaluator diversity. Second, **strict timeouts with neutral defaults** (e.g., a timed-out evaluator contributes a zero) prevent optimistic bias. Third, **evidence logging**—storing the draft, evaluator payloads, scores, and actions—turns fairness into an observable system property that can be debugged and audited like any other production signal.

6. Limitations and Threats to Validity

Even with strong empirical results, multi-agent fairness evaluators face important limitations. First, evaluator *agreement* is not the same as ground truth. When multiple agents converge on a fairness verdict, they may be converging on a shared bias or a prompt artifact rather than reality. This threatens *construct validity*: we risk measuring "what the evaluators prefer" rather than "what is fair." Mitigations include periodic human audits on stratified samples and counterfactual tests that flip protected attributes while holding task semantics constant [19].

A second limitation is *prompt and context sensitivity*. Small changes in phrasing, register, or code-switching can swing evaluator scores, especially in multilingual deployments. This introduces variance that can be mistaken for bias or, worse, can mask genuine disparities. We reduce this risk by using prompt ensembles, perturbation testing (typos, dialect variants), and cached retrieval evidence so that evaluators reference the same policies regardless of surface variations.

Third, evaluators themselves can carry *cultural and linguistic priors* that are not globally representative. A detector tuned on English corporate communication may misinterpret tone in community or dialectal contexts. This limits *external validity*: conclusions may not generalize across locales. Rotating evaluator pools, region-specific calibration sets, and sampling plans that over-index underrepresented languages can partially address this gap but cannot eliminate it entirely [20].

Fourth, *gaming* is possible. Once teams learn the evaluator mix and thresholds, they may optimize prompts for passing scores rather than for genuine neutrality (Goodhart's law). To deter this, we randomize adversarial probes, rotate role prompts, and rate-limit evaluator visibility in dashboards so that operational users cannot reverse-engineer guards in real time.

 doi:10.20944/preprints202512.1084.v1

Fifth, *calibration drift* and *policy drift* occur. As upstream policies or data distributions change, aggregator weights and governance thresholds can become stale. Without monitoring, fairness scores will look stable even as decisions degrade. Continuous calibration windows, drift alarms on score distributions, and "shadow thresholds" (evaluated but not enforced) help detect and stage updates safely.

Sixth, fairness actions have *cost and latency budgets*. Adding evaluators improves detection but increases p95 latency and spend; strict timeouts can hide bias by forcing neutral defaults. We mitigate with bounded concurrency, per-role timeouts, and caches, but there will remain a production trade-off between coverage and service quality. Clear service-level objectives (SLOs) for fairness and latency are required to pick an operating point.

Finally, *governance noise* limits attribution. Human reviewers differ in risk tolerance; organizational policies vary across teams and time. This injects label noise into "ground truth" actions used to train or tune thresholds. To control this, we log evidence with decisions, normalize reviewer rubrics, and re-score historical artifacts when policies change, but some uncertainty remains irreducible.

Figure 6 organizes the most common failure sources we encountered while benchmarking fairness with multi-agent evaluators along a single "spine" from input to released output. Upstream issues cluster around *data and sampling* (coverage gaps, leakage from templated probes), *prompting and context* (register, dialect, code-switching), and the *evaluators* themselves (cultural priors, timeouts). Downstream risks arise in *aggregation* (stale weights, threshold drift), *governance* (rubric variance, inconsistent reviewer actions), and *deployment* (latency/cost trade-offs that suppress detection). Read left-to-right, the diagram makes it easy to map any fairness regression to a concrete subsystem with an associated mitigation (e.g., add strata to sampling, tighten evaluator timeouts, recalibrate weights, normalize reviewer rubrics), so fixes are targeted rather than generic.

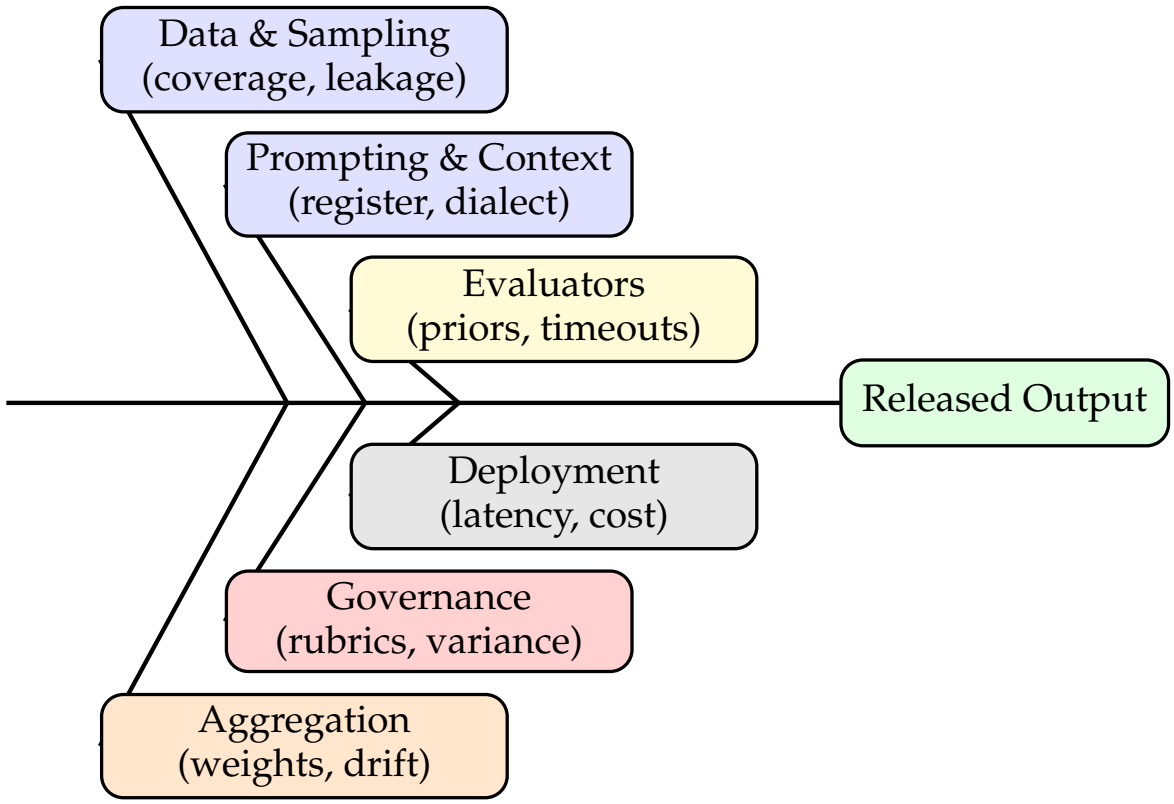

Figure 6. Fishbone of failure sources in multi-agent fairness benchmarking. Branches map common threats to concrete subsystems, enabling targeted mitigations rather than generic fixes.

Figure 6 summarizes major failure sources as a compact "fishbone" diagram. The spine represents released outputs; each branch enumerates upstream causes—data, prompts, evaluators, aggregation, governance, and deployment—so engineering teams can map mitigations to concrete failure points.

 doi:10.20944/preprints202512.1084.v1

Figure 7 summarizes the operational loop we use to keep fairness signals reliable over time: *instrument* scores/actions/evidence; *sample* stratified slices across languages, groups, and tasks; *perturb* prompts (dialect, typos, register) to expose sensitivity; *red-team* with randomized adversarial probes; *recalibrate* aggregator weights and governance thresholds; and conduct *human audit* for rubric alignment. The final arrow closes the loop so that audit insights feed back into instrumentation. Practically, this cycle reduces false comfort from static scores, catches drift early (via distribution shifts in signals), and turns fairness from one-off testing into a continuous, auditable control that teams can run week after week without manual rewiring.

Figure 7 presents a compact mitigation loop used in practice: instrument, sample, perturb, red-team, recalibrate, and audit.

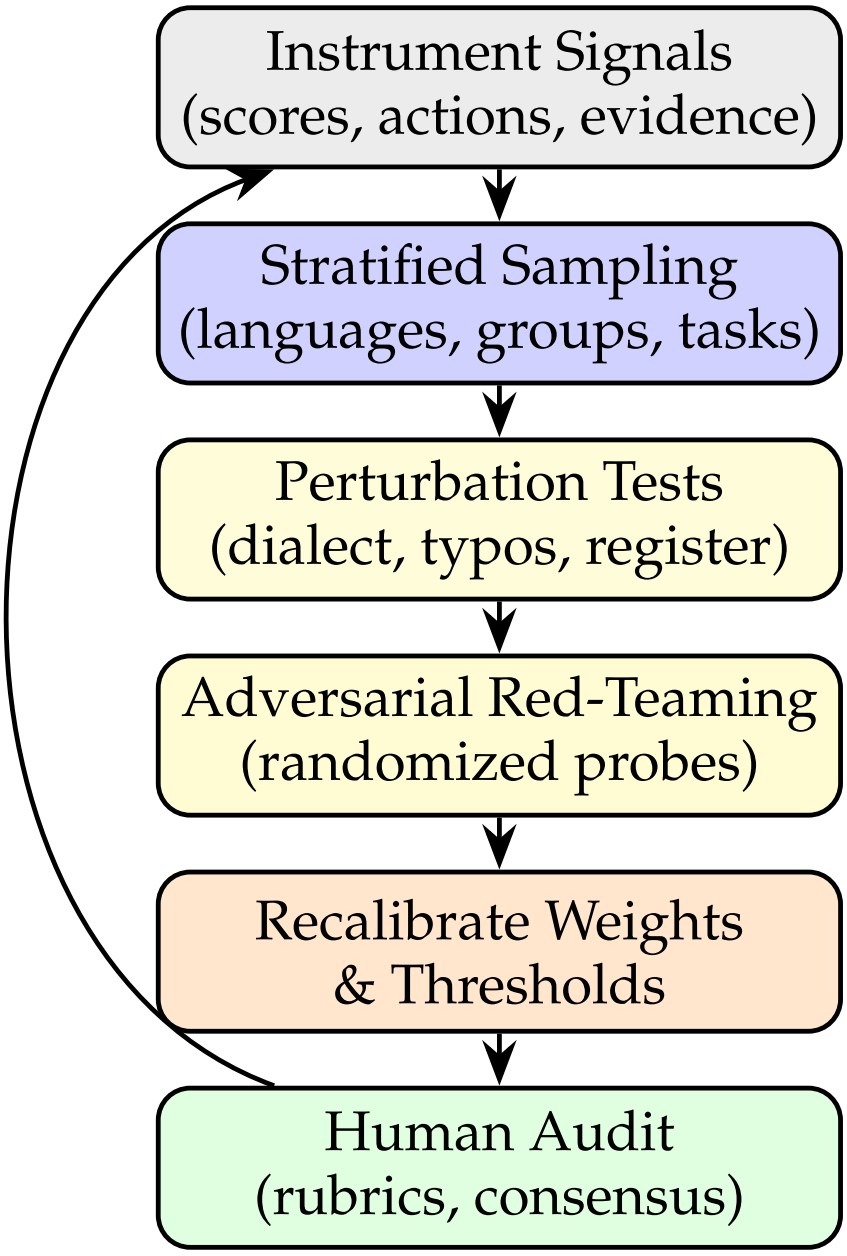

Figure 7. Mitigation loop for threats to validity. Evidence-driven instrumentation enables stratified sampling, perturbation and red-teaming; recalibration and human audit close the loop.

7. Conclusions

This paper argued that fairness for large language models cannot be treated as a one-time certification on static datasets. Instead, it must be an operational property that is continuously

 doi:10.20944/preprints202512.1084.v1

measured, acted upon, and audited. We proposed a multi-agent evaluation paradigm in which specialized evaluators—demographic, contextual, and adversarial—assess drafts from a generator and yield structured signals that can be aggregated into a single, actionable fairness score. The result is not a new leaderboard, but a control surface: a way to keep systems within defined policy ranges while they operate in the wild.

Across case studies in contact-center triage, credit pre-screening, and public benefits information services, the evaluator stack exposed risks that static tests often miss: tone drift by language, proxy attributes creeping into rationales, and missing safety language in sensitive guidance. Because signals were logged alongside inputs, drafts, and actions, teams could trace causes, rehearse mitigations, and tune thresholds with confidence. Fairness became observable and debuggable, much like latency or error rates.

The experimental setup highlighted the practical trade-offs that come with scale. Adding evaluator roles improves detection, but also taxes latency and cost. Our results suggested a sub-linear increase in tail latency due to concurrency and caching, with diminishing returns in detection beyond three to five evaluators. Rather than seeking a universal optimum, we framed the choice of pool size and thresholds as an operating point to be selected against service-level objectives and risk tolerance.

We also surfaced limits and threats to validity. Evaluator agreement is not ground truth; prompts and linguistic context can sway scores; cultural priors may bleed into detectors; and governance adds human variance to the loop. The mitigation cycle we outlined—instrument, sample, perturb, red-team, recalibrate, audit—does not eliminate these issues, but it makes them tractable and transparent. It also creates the organizational muscle memory needed to sustain fairness over time, not just during pilot phases.

For engineering leaders, the important lesson is architectural: fairness needs first-class interfaces. Evaluator outputs should be typed, versioned, and stored; aggregators should be configurable and testable; governance gates should make consistent, auditable decisions; and dashboards should present distributions, not just single scores. When these interfaces exist, teams can evolve roles, weights, and thresholds without rewriting the system, and audits become a matter of replaying evidence rather than recreating it.

Finally, while multi-agent evaluators help contain present-day risks, they also prepare systems for change. New policies, new languages, and new usage patterns can be accommodated by adding or rotating roles, re-calibrating weights, and rehearsing the mitigation loop. In this sense, the approach is less a destination than a practice: a way of working that treats fairness as a living requirement. By embedding that practice into everyday operations, organizations can deploy LLMs with greater confidence that improvements in capability will be matched by improvements in equity and accountability.

References

1. Shahane, R. Improving ML Model Accuracy Through Data-Centric Engineering in Enterprise Data Lakes, 2024. https://doi.org/10.5281/zenodo.17161087.
2. Nangia, N.; Vania, C.; Bhalerao, R.; Bowman, S.R. CrowS-Pairs: A Challenge Dataset for Measuring Social Biases in Masked Language Models. In Proceedings of the Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, 2020, pp. 1953–1967. https://doi.org/10.18653/v1/2020.emnlp-main.154.
3. Raji, I.D.; Smart, A.; Binns, R.N.; O'Neil, P.V.L.; Mazhar, A.; Levy, S.D.; Buolamwini, J. Closing the AI Accountability Gap: Defining Audit and Assurance for Algorithms. In Proceedings of the Proceedings of the 2020 ACM Conference on Fairness, Accountability, and Transparency (FAccT). ACM, 2020, pp. 33–44. https://doi.org/10.1145/3351095.3372873.
4. Alang, K.; Peta, S.B.; Pai, R.R.; Patil, B. Scalable Cloud Architectures for Efficient Processing of Multi-Structured Big Data. In Proceedings of the 2025 Global Conference in Emerging Technologies (GCET). IEEE, 2025. https://doi.org/10.1109/GCET.2025.11077101.
5. Bender, E.M.; Gebru, T.; McMillan-Major, A.; Shmitchell, S. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In Proceedings of the Proceedings of the 2021 ACM Conference on Fairness,

 doi:10.20944/preprints202512.1084.v1

Accountability, and Transparency (FAccT). ACM, 2021, pp. 610–623. https://doi.org/10.1145/3442188.3445922.

6. Mitchell, M.; Wu, S.; Zaldivar, A.; Barnes, P.; Vasserman, L.; Hutchinson, B.; Spitzer, E.; Raji, I.D.; Gebru, T. Model Cards for Model Reporting. In Proceedings of the Proceedings of the 2019 ACM Conference on Fairness, Accountability, and Transparency (FAT*). ACM, 2019, pp. 220–229. https://doi.org/10.1145/3287560.3287596.
7. Gebru, T.; Morgenstern, J.; Vecchione, B.; Vaughan, J.W.; Wallach, H.; III, H.D.; Crawford, K. Datasheets for Datasets. *Communications of the ACM* **2021**, *64*, 86–92. https://doi.org/10.1145/3458723.
8. Pasam, V.R.; Devaraju, P.; Methuku, V.; Dharamshi, K.; Veerapaneni, S.M. Engineering Scalable AI Pipelines: A Cloud-Native Approach for Intelligent Transactional Systems. In Proceedings of the Proceedings of the 2025 International Conference on Intelligent Systems and Cloud Computing. IEEE, 2025.
9. Veluguri, S.P. Deep PPG: Improving Heart Rate Estimates with Activity Prediction. In Proceedings of the Proceedings of the 2025 1st International Conference on Biomedical AI and Digital Health. IEEE, 2025.
10. Shahane, R.; Prakash, S. Quantum Machine Learning Opportunities for Scalable AI. *Journal of Validation Technology* **2025**, *28*, 75–89. https://doi.org/10.1080/jvtnetwork.v28i1.131.
11. Chouldechova, A. Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments. *Big Data* **2017**, *5*, 153–163. https://doi.org/10.1089/big.2016.0047.
12. Zhao, J.; Wang, T.; Yatskar, M.; Ordonez, V.; Chang, K.W. Gender Bias in Coreference Resolution: Evaluation and Debiasing Methods. In Proceedings of the Proceedings of NAACL-HLT 2018. Association for Computational Linguistics, 2018, pp. 15–20. https://doi.org/10.18653/v1/N18-2003.
13. Rudinger, R.; Naradowsky, J.; Leonard, B.; Durme, B.V. Gender Bias in Coreference Resolution. In Proceedings of the Proceedings of NAACL-HLT 2018. Association for Computational Linguistics, 2018, pp. 8–14. https://doi.org/10.18653/v1/N18-2002.
14. Natarajan, G.N.; Veerapaneni, S.M.; Methuku, V.; Venkatesan, V.; Kanji, R.K. Federated AI for Surgical Robotics: Enhancing Precision, Privacy, and Real-Time Decision-Making in Smart Healthcare. In Proceedings of the Proceedings of the 2025 5th International Conference on Emerging Technologies in Healthcare (ICETH). IEEE, 2025.
15. Ribeiro, M.T.; Wu, T.; Guestrin, C.; Singh, S. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. In Proceedings of the Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2020, pp. 4902–4912. https://doi.org/10.18653/v1/2020.acl-main.442.
16. Caliskan, A.; Bryson, J.J.; Narayanan, A. Semantics Derived Automatically from Language Corpora Contain Human-like Biases. *Science* **2017**, *356*, 183–186. https://doi.org/10.1126/science.aal4230.
17. De-Arteaga, M.; Romanov, A.; Wallach, H.; Chayes, J.; Borgs, C.; Chouldechova, A.; Geyik, S.S.; Kenthapadi, K.; Beutel, A.; Weikum, G.; et al. Bias in Bios: A Case Study of Semantic Representation Bias in a High-Stakes Setting. In Proceedings of the Proceedings of the 2019 ACM Conference on Fairness, Accountability, and Transparency (FAT*). ACM, 2019, pp. 120–128. https://doi.org/10.1145/3287560.3287572.
18. Nadeem, M.; Bethke, A.; Reddy, S. StereoSet: Measuring Stereotypical Bias in Pretrained Language Models. In Proceedings of the Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2021, pp. 5356–5371. https://doi.org/10.18653/v1/2021.acl-long.416.
19. Bellamy, R.K.E.; Dey, K.; Hind, M.; Hoffman, S.C.; Houde, S.; Kannan, K.; Lohia, P.; Martino, J.; Mehta, S.; Mojsilović, A.; et al. AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias. *IBM Journal of Research and Development* **2019**, *63*, 4:1–4:15. https://doi.org/10.1147/JRD.2019.2942287.
20. Kiela, D.; Bartolo, M.; Nie, Y.; Kaushik, D.; Geiger, A.; Wu, Z.; Vidgen, B.; Prasad, G.; Singh, A.; Ringshia, P.; et al. Dynabench: Rethinking Benchmarking in NLP. In Proceedings of the Proceedings of NAACL-HLT 2021. Association for Computational Linguistics, 2021, pp. 4110–4124. https://doi.org/10.18653/v1/2021.naacl-main.191.