

# Evidence-Based Validation of AI-Generated Enterprise Decisions

Mr. Girish Rajagopalan Sesha

Cognizant Technology Solutions

**Abstract** - Enterprise recommendations require evidence that is sufficient for a particular action, not merely factual sentences. A fully supported answer can omit an approval, rely on a superseded policy, or apply a rule to the wrong population. This article develops a context-indexed evidence-obligation ledger: a domain-approved set of mandatory premises exists independently of generated text, and every premise must carry applicable, current support before evidence acceptance. The ledger exposes omitted obligations, records unresolved contradictions, and invalidates acceptance when dependent sources or context change. The contribution is a decision-completeness representation and its lifecycle semantics; it is not a new entailment algorithm or constraint language. A deterministic feasibility demonstration matched authored routing expectations in 36 cases spanning 12 scenario types. An output-support-only proxy accepted 24 cases that the authored contract required to be reviewed. These are conformance findings with preassigned support labels, not estimates of LLM accuracy or comparative enterprise effectiveness. An experiment diagram, protocol, complexity analysis, and reproducibility package define the remaining empirical validation.

**Keywords** - Evidence validation; mandatory premises; retrieval-augmented generation; decision completeness; enterprise AI; feasibility study.

## I INTRODUCTION AND RESEARCH GAP

Enterprise AI recommendations can trigger access changes, procurement commitments, or service actions. Fact-checking asks whether an assertion is supported. A decision also requires all relevant prerequisites, the correct policy scope, valid authority, and an evidence state that remains current until approval. “The manager approved access” may be true while the requested application is prohibited for contractors or the proposed expiry exceeds the limit. Correct facts are therefore necessary inputs to many enterprise decisions, but are not a sufficient acceptance criterion.

Retrieval-augmented generation (RAG) connects generation to external information [1], but hallucination remains a limitation [2]. ALCE evaluates citations [3]; FActScore decomposes generated text into atomic facts and assesses support [4]. RAGAs and ARES offer response and retrieval evaluation methods [5], [6]. These approaches motivate evidence assessment. The engineering gap addressed here is a missing operational contract between the premises a decision requires and the claims an answer happens to express.

Hagström et al. [7] examine unreliable, insufficient, and difficult retrieved contexts, cautioning against synthetic settings that misrepresent retrieval conditions. Ge et al. [8] study conflicting evidence and source credibility. This article builds on those concerns but focuses on enterprise obligations: which policy governs this action, which prerequisites remain missing, and when a previously accepted record must be reassessed. It extends the Enterprise AI Transformation Framework (EATF) at the individual recommendation boundary.

## A Research question and contribution

The research question is whether decision-specific evidence obligations can identify incomplete or inapplicable recommendations that output-level support checks accept, while preserving valid decisions at manageable cost. The new artifact specified here is a context-indexed obligation ledger with three semantics: requirement instantiation before answer inspection, an explicit unresolved entry for an omitted requirement, and dependency-bound expiry of acceptance. Its observable outputs are an obligation-completion state, a repair set, and an invalidation set. These outputs make decision completeness separately testable from factual precision.

This is a scoped representational contribution, not a claim that missing-field validation, rule engines, provenance, or temporal reasoning are new. A rule system or knowledge graph can implement the same semantics. The study contributes an explicit mapping from narrative recommendations to those obligations, a conditional acceptance invariant, and an executable routing demonstration. Broad originality over all compliance systems and improved LLM performance are not established.

## II METHOD AND FEASIBILITY SCOPE

The study follows design-science methods [9], [10]: define the decision problem, specify the artifact, analyze failure cases, demonstrate executable routing, and formulate comparative evaluation. Completed work consists of the artifact specification, worked analysis, and deterministic routing demonstration; semantic and human evaluations remain planned. The demonstration uses authored synthetic fixtures, a fixed schema, and preassigned support labels. It contains no generative model, semantic entailment classifier, live enterprise data, or actual access changes.

## A Mapping validation and coverage

Each requirement maps to a premise identifier, applicable context, authoritative owner, evidence field, failure route, and

test case. Positive and negative fixtures check both acceptance and review. The mapping is internally exercised, but has not been independently validated by domain experts. Independent reviewers should confirm the prerequisite schemas, adjudicate ambiguous exceptions, and inspect whether narrative-to-premise extraction preserves meaning. Internal conformance cannot establish that every real requirement has been identified.

Completeness is bounded by the declared schema and tool boundary: retrieval access, requirement instantiation, support, authority, validity, applicability, conflict, routing, and invalidation. Unknown schemas produce review. A change log must link a schema revision to affected decisions and regression cases. A model may suggest a new requirement, but cannot silently add, delete, or approve a mandatory requirement in the production registry.

### III ARCHITECTURE AND TECHNICAL ENFORCEMENT

Figure 1 places evidence validation between recommendation generation and a separate authorization workflow. An action envelope identifies the recommendation, request context, schema version, claimed premises, source spans, and dependent object versions. Evidence acceptance means that the configured proof obligations are satisfied; it does not grant permission to execute an action.

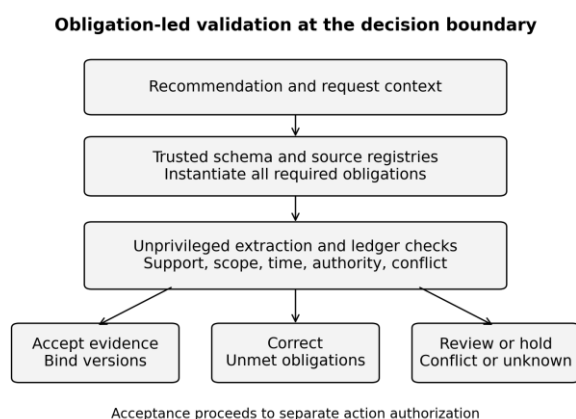


**Fig. 1. Obligation-led evidence validation and the separate authorization boundary.**

#### A Mandatory premises versus atomic facts

An atomic fact is a small assertion extracted from generated text. A mandatory premise is an evidence obligation instantiated from an approved decision schema and context, whether or not the answer mentions it. One premise can require several atomic assertions and documents; one assertion can help satisfy several premises. The difference is the origin and denominator, not the grammatical size of a sentence. The approach therefore complements atomic support scoring rather than replacing it.

For temporary contractor access, the schema requires identity category, application eligibility, authorized approval, and permitted expiry. An answer expressing only the first three can have three supported atomic assertions out of three, yet satisfy only three mandatory obligations out of four. An omitted expiry remains an unresolved ledger entry. If “authorized approval” requires both a signed approval and a reviewer

delegation record, two facts jointly discharge one obligation. Table I contrasts this contract with related validation models.

**Table I. Validation units and capabilities**

| Approach               | Unit and denominator          | Decision relevance                                     |
|------------------------|-------------------------------|--------------------------------------------------------|
| Atomic fact scoring    | Assertions in output          | Omitted prerequisites need a separate reference        |
| Rule compliance [11]   | Configured predicates         | Equivalent checks possible with the same schema        |
| Graph constraints [12] | Nodes, paths and shapes       | Can require missing properties; needs evidence mapping |
| Obligation ledger      | Required premises for context | Tracks missing support, repairs and expiry             |

#### B Ledger and acceptance rule

Let  $M(x,v)$  be the nonempty prerequisite set selected for context  $x$  under approved schema version  $v$ . Each entry contains a predicate, supporting spans, authority class, applicability condition, validity interval, status, and dependency identifiers. Status is supported, contradicted, unresolved, or not applicable. A not-applicable entry is discharged only by an explicit schema-approved exemption; the generator cannot assign this status to bypass a requirement. An unknown or empty schema returns review.

Define  $G$  as every mandatory entry being supported or explicitly exempt, current, applicable, and free of an unresolved decisive conflict. Define  $H$  as the record matching the recommendation and relevant context, source, and schema versions. Acceptance is  $A = G \text{ AND } H$ . The repair set contains entries failing  $G$ ; the invalidation set contains accepted records depending on changed objects. Under correct schema selection and reliable checking, acceptance implies no known unmet obligation in the inspected state. This conditional invariant does not imply source truth or adequate business judgment.

#### C Worked routing example

A recommendation first cites an archived rule and omits expiry. The ledger requests current eligibility evidence and an expiry premise. A corrected answer citing the active rule still cannot pass if contractor eligibility is unresolved. Once all obligations are discharged, acceptance records the source and request versions. A subsequent contractor-status change invalidates that acceptance even if the generated text remains unchanged. Correctable gaps may receive a bounded retry; unresolved authority or exhausted retries go to review.

#### D Enforcing the instruction and data boundary

The production design separates a read-only source resolver, an unprivileged extractor, a deterministic contract service, and an authorized action adapter. Retrieved text enters a quoted data field; no document is evaluated as code or loaded as policy. The extractor returns schema-validated candidate claims and evidence links only. Unknown fields are rejected. The contract service obtains approved schemas, precedence, and source metadata from authenticated registries, not from model-supplied document instructions.

The extractor has no policy-write API, approval credential, or unrestricted tool token. The adapter accepts only

authenticated, version-bound validation records and independently checks action authorization. Network permissions and service identities restrict access to registry update endpoints. Document hashes bind inspected versions but do not prove content safety. AgentDojo motivates adversarial tests [13]; OWASP guidance supports keeping privilege controls outside the model [14]. Delimiters and injection filters alone cannot enforce this boundary. The pilot tests only that arbitrary text is ignored by its deterministic checker; it does not test the security of a deployed extractor, registry, or adapter.

#### IV CONCEPTUAL DIFFERENTIATION AND CONFLICT HANDLING

##### A Beyond a collection of checks

The ledger changes the object being validated from an answer to a decision-specific evidence obligation set. Removing an inconvenient sentence cannot improve completeness because M is fixed independently. A repair request names an unmet obligation rather than issuing an unstructured “try again.” A source or context change invalidates only dependent acceptance records. The contribution is this obligation, repair, and invalidation semantics at the narrative-to-decision interface. Its implementation can reuse existing rule and graph engines; no new expressive power over those engines is claimed.

##### B Comparison with compliance rules and graphs

A rule engine can require expiry and approval fields and evaluate their values [11]. SHACL can validate graph shapes, including minimum property counts, and emit validation results [12]. Thus omission detection is not inherently unique to mandatory premises. The additional engineering artifact links each requirement to narrative support, temporal and contextual qualifiers, explicit unresolved evidence, and a downstream record lifetime. With identical mappings and semantics, an equivalent rule or graph implementation should return identical routes. A fair experiment should test mapping effort, traceability, and invalidation cost rather than assume better logical correctness.

For example, a graph containing an approval node can satisfy a structural existence constraint while the cited document does not establish the reviewer’s authority. A suitably enriched graph can also represent that authority and its validity. The ledger makes those obligations explicit before graph construction; it does not resolve the semantic extraction problem by naming it. Missing evidence remains unresolved, whereas a contradicted premise has explicit opposing evidence. Both can prevent acceptance for different recorded reasons.

##### C Conflicting evidence and governed precedence

Hagström et al. [7] motivate realistic context-quality testing; Ge et al. [8] investigate credibility-aware conflict resolution. Here, disagreement is resolved at the affected obligation under a domain-approved precedence policy. First remove inaccessible, expired, or out-of-scope candidates. For normative policy sources, an explicitly superseding authorized rule may dominate an older rule; the rejected source and reason remain in the audit record. Equally authoritative

applicable sources with incompatible values cause review. Source counts and model confidence cannot outvote an unresolved tie.

For instance, an archived policy allowing 30-day contractor access does not override an active, applicable 7-day policy. Two current policies with equal authority and incompatible expiry limits produce a conflict record for the expiry obligation. Conflicting observations about a requester’s identity are not resolved merely by policy precedence; they require the designated identity system or human review. This separates policy authority from factual source credibility. The specified contribution is the obligation-level resolution and acceptance consequence, not a demonstrated improvement over [7] or [8].

#### V FEASIBILITY DEMONSTRATION AND SCALABILITY

##### A Implementation and authored fixtures

A Python reference checker instantiates four fixed obligations, indexes source records, filters evidence by support, validity and scope, and reviews missing obligations, tied conflicting values, unknown schemas, or changed context versions. Higher-authority normative evidence dominates lower-authority evidence under the supplied fixture policy. Only accept and review routes are implemented; correction generation, learned extraction, production authentication, and persistent dependency invalidation are outside this prototype.

The fixture set contains 12 types, each with three request identifiers: valid evidence, omitted premise, stale source, wrong scope, equal-authority conflict, ranked conflict, unknown schema, changed context, unsupported claim, missing source, instruction text with valid evidence, and instruction text with an omitted premise. Nine fixtures are expected to be accepted and 27 reviewed. Repeated identifiers exercise record isolation; they are not 36 independent or representative samples. Expected routes were authored alongside the cases, not labeled by independent experts.

##### B Executed conformance findings

The contract matched all 36 authored expectations, accepting nine and reviewing 27. A deliberately limited support-only proxy checks whether every supplied assertion is marked supported; it does not check omitted requirements or metadata. It accepted 33 cases, including 24 for which the authored contract required review, and reviewed three. Table II reports counts. This demonstrates executable routing and exposes the proxy’s specified blind spots; it is not an end-to-end comparison against RAG, FActScore, a complete citation checker, SHACL, or a commercial compliance system.

Table II. Deterministic fixture outcomes

| Measure                      | Support-only proxy | Contract checker |
|------------------------------|--------------------|------------------|
| Authored cases               | 36                 | 36               |
| Accepted                     | 33                 | 9                |
| Review required but accepted | 24                 | 0                |
| Valid cases reviewed         | 0                  | 0                |
| Expected routes matched      | 12                 | 36               |

The support labels and trusted metadata are preassigned. The checker can therefore match the contract while a real extractor would fail to find an exception or misread evidence. Perfect fixture conformance is not verifier accuracy, a safety guarantee, or a publishable estimate of production benefit. The two instruction-text cases confirm only the absence of a text-to-policy path in this small program. No effect size, human workload saving, or attack-resistance rate is inferred.

### C Complexity bounds and measured scaling

Let  $m$  be the instantiated obligations,  $s$  the source records, and  $e$  the supplied premise-evidence links. With hashed source lookup and links already mapped to obligations, indexing and deterministic routing take expected  $O(s + m + e)$  time and  $O(s + m + e)$  memory, excluding text storage and model inference. If each obligation has at most  $k$  links,  $e$  is at most  $mk$ . The prototype scans links, then takes a maximum and compares top-authority values within each slot; it does not perform all-pairs conflict checks. A richer pairwise semantic conflict detector could require  $O(mk^2)$  comparisons and must be budgeted separately.

Semantic support assessment requires at most  $e$  premise-span evaluations under the chosen cap, plus retrieval and extraction. For  $q$  concurrent requests and  $b$  parallel assessment workers, a useful planning bound is approximately  $\text{ceiling}(qe/b)$  times the per-assessment latency, plus retrieval and orchestration; this is a scheduling estimate under uniform work, not a measured SLA. Updating one dependency requires visiting its  $d$  linked acceptance records, expected  $O(d)$ , if a reverse index is maintained. Without that index, scanning all  $D$  records costs  $O(D)$ . Retrieval complexity depends on the actual index and is not claimed to be linear or logarithmic here.

The scan microbenchmark ran on Python 3.12.14 on Linux x86\_64. Each size used seven batches of ten calls; Table III reports the median batch-average milliseconds per call, excluding construction, retrieval, network, and inference. Instrumented input-scan visits equaled input size; subsequent per-slot passes are additional linear work. The near-linear trend supports the stated scan bound in this environment; shared-host timing is not a production forecast. No warm-up exclusion or confidence intervals were used.

Table III. Local deterministic scan measurements

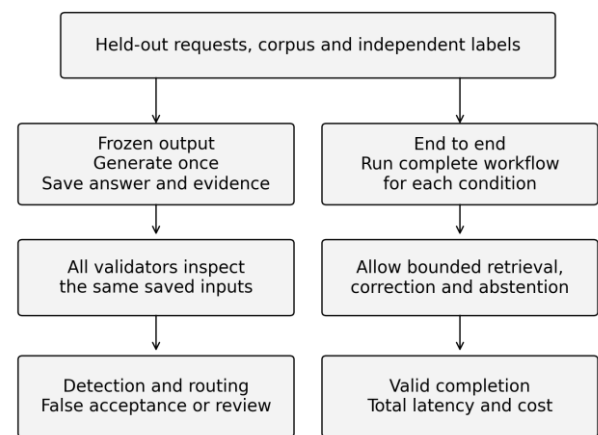
| Evidence links | Input-scan visits | Median ms per call |
|----------------|-------------------|--------------------|
| 40             | 40                | 0.009              |
| 400            | 400               | 0.077              |
| 4000           | 4000              | 0.946              |
| 40000          | 40000             | 10.379             |

Empirical deployment expectations are testable: holding model and span lengths fixed, doubling uncached links should roughly double assessment work; batching may reduce wall-clock latency while increasing resource demand. Source churn should reduce cache benefit and increase invalidation traffic. Measure these effects with burst loads, document updates, bounded queues, and separate latency percentiles. The present measurements assess neither semantic throughput nor organizational review capacity.

## VI COMPARATIVE EVALUATION PROTOCOL

### A Frozen-output and end-to-end designs

Figure 2 distinguishes two questions. Frozen-output evaluation generates recommendations once, stores their claims and retrieved evidence, and applies every validator to the same immutable inputs. It estimates detection and routing differences without generator changes. End-to-end evaluation reruns the full workflow, allowing bounded retrieval, correction, and abstention under each condition. It measures completed valid decisions and total cost; its gains can arise from changed generation as well as validation. The feasibility study above covers deterministic routing only and does not execute either LLM experiment.



Routing feasibility pilot is separate from both LLM experiments

Fig. 2. Frozen-output detection comparison versus end-to-end workflow comparison.

### B Baselines and protocol summary

The planned comparison includes ordinary RAG with existing authorization, citation-support checking, the obligation contract, and a semantically equivalent rule or graph implementation. Match generator versions, corpus access, initial requests, budgets, and decision schemas where relevant. A weak proxy cannot establish superiority over a fully configured compliance baseline. Table IV summarizes

independent labels, splits, and outcomes. Semantic extraction and source authority must be scored separately so routing correctness does not conceal upstream errors.

**Table IV. Planned comparative protocol**

| Stage               | Control or separation                | Primary outputs                                 |
|---------------------|--------------------------------------|-------------------------------------------------|
| Dataset             | Split by policy and scenario family  | Support, applicability, required-premise labels |
| Frozen output       | Identical saved answers and evidence | Missed defects, false rejection, coverage       |
| End to end          | Matched requests; bounded retries    | Valid completion, total cost, latency           |
| Rule or graph       | Same obligations and semantics       | Agreement, mapping effort, invalidation cost    |
| Stress and ablation | Churn, bursts; remove one check      | Stale acceptance, queue and component effects   |

At least two domain annotators should independently label support, applicability, mandatory premises, and acceptable decisions, with adjudication and reported agreement. Separate training, threshold validation, and held-out testing by policy and scenario families. Freeze model versions, prompts, evidence caps, thresholds, and retry budgets before testing. Sample size should follow required precision and clustered design; the 36 authored feasibility fixtures cannot justify that size.

Report unsupported mandatory premises among accepted decisions, accepted decisions missing prerequisites, false review among valid cases, acceptance coverage, and valid completion per attempted request. Include cost, token use, end-to-end latency percentiles, and active reviewer minutes only when measured in a consented human study. Paired differences need uncertainty intervals clustered by scenario family. Ablations remove applicability, freshness, conflict handling, or binding individually; keep downstream authorization fixed. A deny-everything system must not appear successful because it has no accepted errors.

## VII DISCUSSION AND CONCLUSION

Calibration must be evaluated against independent labels rather than self-reported confidence [15]. Risk management and auditing determine schema ownership and accountable exceptions [16], [17]; controlled engineering and regression tests support maintenance of the validation service [18]. These practices complement the ledger but do not certify it.

The study clarifies why supported text can remain insufficient for enterprise action. The obligation ledger separates output factuality from required decision evidence and makes omissions, conflicts, repairs, and acceptance lifetime explicit. Rule engines and graph constraints can realize these semantics; the contribution is the evidence-to-obligation representation and its operational contract, not a new logic. The deterministic demonstration establishes feasibility of the routing core and provides reproducible counts and scan measurements.

The main threats to validity are authored-case bias, repeated scenario structures, preassigned support labels, a fixed four-obligation schema, simplified precedence, and the absence of retrieval, learned verification, independent domain review, and live integration. In production, a mistaken schema or

compromised registry can defeat an otherwise correct checker. A conservative conflict policy can also increase review burden. These limits require semantic and human evaluation before any claim of improved decision safety or efficiency.

The reproducibility supplement contains pilot.py, all 36 serialized fixtures, per-case outcomes, environment details, timing method, and the script SHA-256. Running the standard-library script reproduces routing counts and generates fresh timing values. No proprietary data or external model service is required. A full evaluation should add realistic contexts and independent labels, following the two experiment designs rather than treating internal conformance as empirical validation of enterprise AI.

## REFERENCES

- [1] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.
- [2] Z. Ji et al., "Survey of Hallucination in Natural Language Generation," *ACM Computing Surveys*, vol. 55, no. 12, art. 248, 2023. doi: 10.1145/3571730.
- [3] T. Gao, H. Yen, J. Yu, and D. Chen, "Enabling Large Language Models to Generate Text with Citations," *Proc. EMNLP*, pp. 6465–6488, 2023. doi: 10.18653/v1/2023.emnlp-main.398.
- [4] S. Min et al., "FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation," *Proc. EMNLP*, pp. 12076–12100, 2023. doi: 10.18653/v1/2023.emnlp-main.741.
- [5] S. Es, J. James, L. Espinosa Anke, and S. Schockaert, "RAGAs: Automated Evaluation of Retrieval Augmented Generation," *Proc. EACL: System Demonstrations*, pp. 150–158, 2024. doi: 10.18653/v1/2024.eacl-demo.16.
- [6] J. Saad-Falcon, O. Khattab, C. Potts, and M. Zaharia, "ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems," *Proc. NAACL-HLT*, pp. 338–354, 2024. doi: 10.18653/v1/2024.naacl-long.20.
- [7] L. Hagström et al., "A Reality Check on Context Utilisation for Retrieval-Augmented Generation," *Proc. ACL*, pp. 19691–19730, 2025. doi: 10.18653/v1/2025.acl-long.968.
- [8] Z. Ge, Y. Wu, D. W. K. Chin, R. K.-W. Lee, and R. Cao, "Resolving Conflicting Evidence in Automated Fact-Checking: A Study on Retrieval-Augmented LLMs," *Proc. IJCAI*, pp. 9656–9664, 2025. doi: 10.24963/ijcai.2025/1073.
- [9] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in Information Systems Research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004. doi: 10.2307/25148625.
- [10] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," *J. Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007. doi: 10.2753/MIS0742-1222240302.
- [11] Open Policy Agent, "Policy Language," documentation, <https://www.openpolicyagent.org/docs/policy-language> (accessed October 18, 2026).
- [12] H. Knublauch and D. Kontokostas, Eds., *Shapes Constraint Language (SHACL)*, W3C Recommendation, July 20, 2017. <https://www.w3.org/TR/2017/REC-shacl-20170720/>.
- [13] E. DeBenedetti, J. Zhang, M. Balunović, L. Beurer-Kellner, M. Fischer, and F. Tramèr, "AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents," *arXiv:2406.13352*, 2024. doi: 10.48550/arXiv.2406.13352. Cited arXiv version.
- [14] OWASP, "LLM Prompt Injection Prevention Cheat Sheet," [https://cheatsheetseries.owasp.org/cheatsheets/LLM\\_Prompt\\_Injection\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/LLM_Prompt_Injection_Prevention_Cheat_Sheet.html) (accessed October 18, 2026).

- [15] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," Proc. ICML, PMLR, vol. 70, pp. 1321–1330, 2017.
- [16] NIST, Artificial Intelligence Risk Management Framework (AI RMF 1.0), NIST AI 100-1, 2023. doi: 10.6028/NIST.AI.100-1.
- [17] I. D. Raji et al., "Closing the AI accountability gap: Defining an end-to-end framework for internal algorithmic auditing," Proc. FAT\*, pp. 33–44, 2020. doi: 10.1145/3351095.3372873.
- [18] S. Amershi et al., "Software Engineering for Machine Learning: A Case Study," Proc. ICSE-SEIP, pp. 291–300, 2019. doi: 10.1109/ICSE-SEIP.2019.00042.