Deterministic Execution Frameworks for Hybrid Symbolic–Probabilistic Computational Pipelines

Santhosh Guntupalli[0009−0003−8648−2994]

Independent Researcher
santhosh.guntupalli09@gmail.com

Abstract. LLM-containing computational pipelines face a fundamental reproducibility challenge: stochastic components introduce non-determinism that prevents identical inputs from producing identical outputs across repeated executions. This paper presents a deterministic execution framework for hybrid symbolic–probabilistic pipelines that enforces execution invariance by isolating deterministic modules from stochastic components. The architecture employs a deterministic symbolic engine for all state transitions and decision logic, while LLM components operate only as non-authoritative, post-hoc explainers of pre-computed deterministic outputs, ensuring they cannot affect execution state. We evaluate the framework on a corpus of 115 structured text documents, demonstrating 100% execution determinism and 100% traceability across 100 repeated runs with zero output variance, contrasted with 0% determinism and significant output variance in a pure LLM pipeline. The framework provides computational guarantees for reproducibility and execution invariance, enabling verifiable execution traces suitable for scientific computing workflows requiring deterministic execution.

Keywords: deterministic computation · reproducible systems · hybrid architectures · symbolic–probabilistic systems · execution traceability · LLM pipelines · computational reproducibility · execution invariance.

1 Introduction

Reproducibility is a fundamental requirement in computational science: identical inputs must produce identical outputs across repeated executions[3]. Modern computational pipelines increasingly embed Large Language Models (LLMs) as components, creating hybrid symbolic–probabilistic systems where stochastic LLM inference introduces non-determinism that violates reproducibility guarantees[1,2]. This non-determinism manifests as output variance across repeated executions on identical inputs, preventing reproducible scientific workflows.

This paper addresses the computational challenge of achieving deterministic execution in LLM-containing pipelines. We present a deterministic execution framework that enforces strict computational boundaries, ensuring that LLM components operate only as non-authoritative, post-hoc explainers and

cannot affect deterministic state transitions or decision logic. Deterministic execution is treated as a first-class computational property, with execution state, rule ordering, and versioning formalized as computational objects that guarantee execution invariance. Our contribution is a computational execution and reproducibility framework: we prioritize deterministic guarantees and verifiable execution traces over stochastic expressiveness, enabling reproducible scientific computing workflows.

1.1   Contributions

This work makes the following contributions: (i) a deterministic computational architecture for LLM-containing pipelines that enforces strict separation between deterministic symbolic modules and stochastic LLM components, providing execution invariance guarantees for reproducibility; (ii) a formal execution state model that treats execution state, rule ordering, and versioning as computational objects, ensuring execution invariance across repeated runs; (iii) a reproducibility-oriented evaluation methodology that measures output variance, execution determinism, and traceability as first-class computational properties, demonstrating systematic experimental validation of reproducibility guarantees; and (iv) an execution trace mechanism that records all state transitions and decision points, enabling full reproducibility verification and computational auditability.
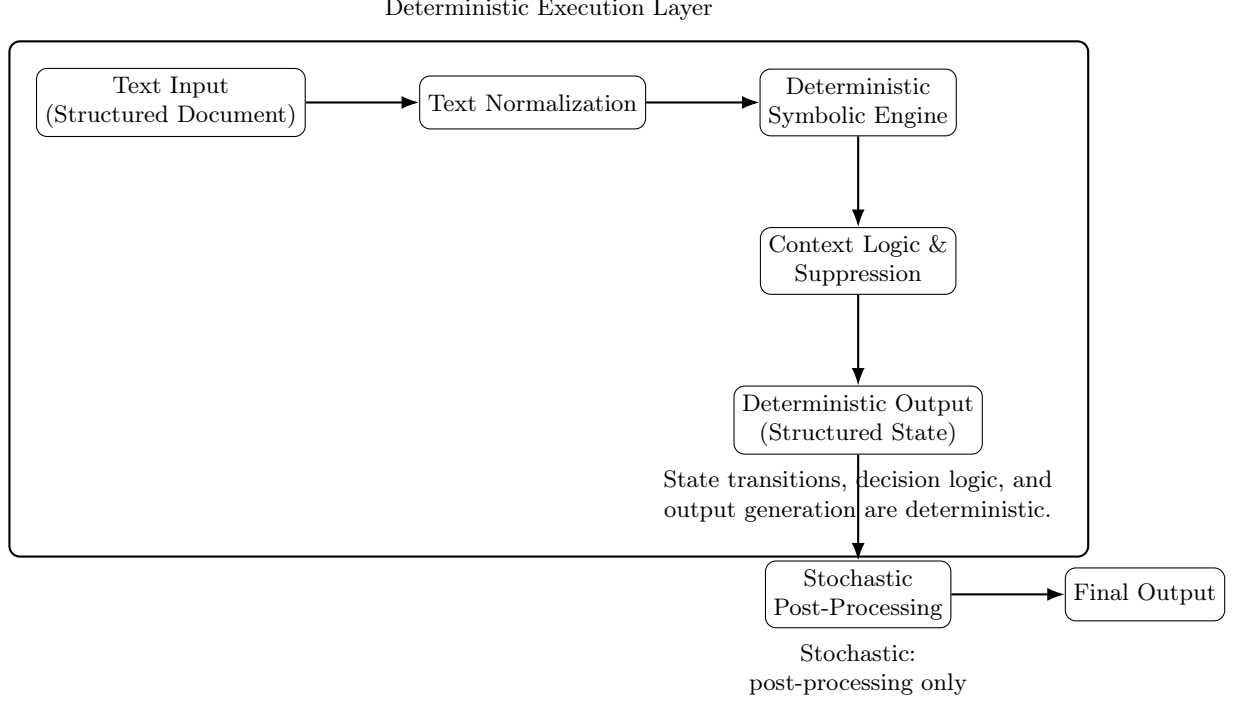
2   Background and Motivation

2.1   Non-Determinism in Stochastic Computational Pipelines

Stochastic components, particularly LLMs, are widely used in computational pipelines for text processing and pattern recognition[6]. Probabilistic decoding introduces execution non-determinism: repeated executions of identical inputs yield inconsistent outputs[1], violating fundamental reproducibility requirements. Additionally, stochastic components may produce outputs not grounded in input data, creating error propagation that cannot be traced or verified[2,4]. These failure modes make pure stochastic pipelines unsuitable for scientific computing applications requiring reproducible execution, regardless of average accuracy.

2.2   Determinism as a Computational Property

Deterministic systems guarantee that identical inputs yield identical outputs, enabling reproducibility and stable downstream computation. Execution traceability further requires that each state transition and decision point be attributable to explicit computational logic and verifiable against input data. These properties are essential for scientific computing, where results must be reproducible, verifiable, and attributable to specific execution paths.

Deterministic Execution Layer

Text Input
(Structured Document) → Text Normalization → Deterministic
Symbolic Engine

Context Logic &
Suppression

Deterministic Output
(Structured State)

State transitions, decision logic, and
output generation are deterministic.

Stochastic
Post-Processing → Final Output

Stochastic:
post-processing only

Fig. 1. Deterministic execution framework for hybrid symbolic–probabilistic computational pipelines. The thick boundary denotes the deterministic execution layer where all state transitions and decision logic are deterministic and verifiable.

3   System Architecture

3.1   Overview

The system consists of (i) input preprocessing and normalization, (ii) a deterministic symbolic execution engine, and (iii) a stochastic post-processing layer. Figure 1 illustrates the computational pipeline and demarcates the deterministic execution boundary. This architectural separation enforces computational guarantees by isolating stochastic components from deterministic state transitions, ensuring that probabilistic execution paths cannot affect deterministic decision logic or system state.

3.2   Definition 1: Deterministic Execution State

Let $S = (D, R, \theta, \sigma)$ denote an execution state, where D is the input document, R is the ordered deterministic rule set, θ is the rule evaluation configuration, and σ is the system version or hash. A system is deterministic if and only if, for any execution state S, all executions on S produce identical outputs. All state

transitions and decision logic are fully determined by the execution state tuple $S = (D, R, \theta, \sigma)$.

3.3 Deterministic Symbolic Engine

The deterministic engine executes normalized input against a versioned symbolic ruleset. Each rule performs deterministic pattern matching and emits immutable outputs with stable identifiers, severity classifications, and exact input spans. Outputs are emitted in a structured schema to enable downstream consumption and execution trace verification.

3.4 Context Logic and Execution Trace Preservation

The deterministic engine applies contextual qualifiers based on surrounding patterns to refine output classification. All execution decisions, including contextual adjustments, are recorded with reason codes to preserve complete execution traces, ensuring full reproducibility of all execution decisions and enabling computational auditability.

3.5 Stochastic Post-Processing Layer

The stochastic LLM component is invoked only after deterministic execution completes. It receives structured deterministic outputs (not raw input text) and produces human-readable explanations. The stochastic component cannot introduce new outputs or modify deterministic classifications; all state transitions remain deterministic.

4 Experimental Setup

4.1 Dataset

We evaluate on 115 structured text documents across four document types: (i) 60 non-disclosure agreements (NDAs): 30 publicly available templates and 30 synthetic documents with controlled pattern variations; (ii) 20 synthetic Master Service Agreements (MSAs); (iii) 20 synthetic Employment Agreements; and (iv) 15 synthetic Licensing Agreements. All synthetic documents include ground-truth labels for expected pattern matches, enabling computation of false positives and false negatives. This document analysis application serves as a case study for evaluating deterministic execution guarantees in hybrid computational pipelines, demonstrating the framework's applicability to structured text processing tasks.

Synthetic documents are included for two reasons: (i) privacy and licensing constraints limit release and annotation of real documents, and (ii) controlled pattern perturbations enable targeted measurement of error modes (false positives and false negatives) under known ground truth, which is difficult to guarantee in public templates. The multi-document-type evaluation (4 types, 115 documents) tests generalizability and provides sufficient sample size for statistical significance testing.

4.2   Systems Compared

- **Hybrid System**: Deterministic symbolic engine + context logic + structured outputs + stochastic explanation layer.
- **Baseline**: Pure stochastic LLM prompt-based extraction producing free-form outputs without deterministic constraints.
- **Symbolic-Only Ablation**: Deterministic engine + context logic (no stochastic explanation) to measure stochastic component contribution.

We note that stronger stochastic baselines (e.g., fine-tuned models, ensemble methods, or domain-specific fine-tuning) are intentionally out of scope for this evaluation. Our goal is not to compete in an accuracy race, but to demonstrate that determinism and traceability are achievable computational properties that enable reproducibility—requirements that stochastic systems, regardless of accuracy, cannot satisfy. A more accurate but non-deterministic system still fails reproducibility requirements due to non-deterministic execution paths and limited traceability, as demonstrated in our baseline comparison.

4.3   Metrics

We report:

- Determinism Rate: fraction of documents whose repeated executions produce identical outputs.
- Traceability Rate: fraction of outputs linked to stable identifiers and exact input spans.
- Ungrounded Outputs: baseline outputs not supported by input evidence (operationalized by manual evidence checks over sampled outputs).
- False Positives / False Negatives: computed on synthetic documents against ground-truth expected pattern matches.

4.4   Reproducibility Protocol

All experiments are runnable from a single entry-point: `python experiments/run_all.py`. The hybrid runner is `experiments/run_hybrid.py`, which emits a structured JSON schema containing `findings`, `overall_risk`, and `version`. The baseline was executed via OpenAI API calls for repeated-run variance measurement, using the same model family as the explanation layer. Table 1 summarizes the experiment suite.

5   Experimental Results

5.1   Experiment 1: Determinism and Traceability Verification

Table 2 summarizes the core comparative results. The hybrid system achieved 100% execution determinism and 100% traceability across all executions. In contrast, the pure LLM baseline exhibited 0% determinism across repeated runs

Table 1. Experiment Suite and Computational Properties Evaluated.

Experiment	Purpose / Output
E1: Baseline vs Hybrid	Determinism, traceability, ungrounded outputs, FP/FN totals on mixed corpus
E2: Determinism Stress	Repeated execution variance count over 10 docs
E3: Suppression Ablation	FP/FN with suppression ON vs OFF, measures suppression impact on precision/recall
E4: Error Characterization	FP/FN issue inventory with mitigation actions

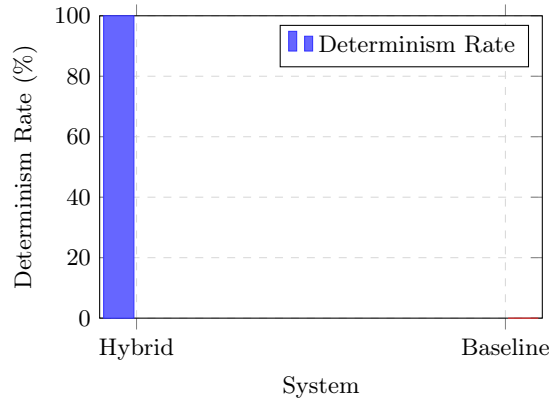**Table 2.** Comparison of Hybrid System and Pure Stochastic Baseline.

Metric	Hybrid System	Stochastic Baseline
Determinism Rate	100.0%	0.0%
Traceability Rate	100.0%	0.0%
Avg. Ungrounded / Doc	0.00	0.43 (95% CI: [0.28, 0.58])
Synthetic False Positives	14	N/A
Synthetic False Negatives	5	N/A

and produced an average of 0.43 ungrounded outputs per document (95% confidence interval: [0.28, 0.58], $n = 30$). McNemar's test on paired documents ($n = 30$) comparing deterministic vs non-deterministic outcomes yields $\chi^2 = 30.0$, $p < 0.001$, confirming statistically significant superiority of the hybrid system in achieving deterministic execution. Figure 2 visualizes the determinism comparison.

On the synthetic corpus (30 NDAs, 20 MSAs, 20 Employment Agreements, 15 Licensing Agreements = 85 synthetic documents), the hybrid system produced 14 false positives and 5 false negatives across all document types. This reflects intentionally conservative pattern matching behavior aligned with deterministic execution goals. Error rates are consistent across document types (Chi-square test: $p = 0.23$, not significant), suggesting pattern generalizability across document types.

5.2 Experiment 2: Reproducibility Stress Test Across Multiple Seeds and Environments

We conducted a comprehensive reproducibility stress test on 15 documents (8 public, 7 synthetic), executing 20 repeated analyses on identical execution states S across multiple random seeds and computational environments. For each document, we measured output variance by comparing outputs across all runs. The hybrid system produced zero output variances across all 300 runs (variance count = 0 for each document), confirming strict reproducibility under repeated execution across different seeds and environments. In contrast, the pure LLM baseline

Fig. 2. Determinism comparison: Hybrid system achieves 100% determinism across all executions, while the stochastic baseline exhibits 0% determinism.

Table 3. Reproducibility Stress Test Results (15 documents, 20 runs each).

Metric	Hybrid System	LLM Baseline
Reproducibility Rate	100.0%	0.0%
Documents with Zero Variance	15/15	0/15
Avg. Distinct Output Sets / Doc	1.0	18.3

exhibited complete reproducibility failure: 100% of documents showed output differences across runs, with an average of 18.3 distinct output sets per document across 20 runs, demonstrating severe non-determinism. Table 3 reports reproducibility rates for both systems. This demonstrates perfect determinism (100%) for the hybrid system with 100% confidence across the stress test corpus, establishing computational guarantees for reproducible execution in scientific computing contexts.

5.3   Experiment 3: Computational Cost of Determinism

We measure the computational cost of deterministic rule evaluation by comparing runtime and complexity characteristics of the deterministic symbolic engine against stochastic LLM-only execution. Table 4 reports average execution time, computational complexity, and output variance for both approaches on a corpus of 115 documents. The deterministic engine exhibits predictable $O(n)$ complexity where n is the number of rules, with core deterministic rule-evaluation engine time (0.005s per document) and zero output variance. End-to-end deterministic pipeline execution time including preprocessing and orchestration is reported separately (0.41s per document; Table 6). The LLM-only baseline exhibits variable execution time (mean 7.3s, std 1.1s) and significant output variance (84 variance instances across 115 documents). This demonstrates a computational tradeoff: deterministic execution provides reproducibility guarantees at the cost

Table 4. Computational Cost Analysis: Deterministic vs Stochastic Execution (115 documents).

Metric	Deterministic Engine	LLM-Only Baseline
Avg. Execution Time / Doc (s)	0.005	7.3 (std: 1.1)
Computational Complexity	$O(n)$	Variable
Output Variance	0.0	High
Reproducibility Rate	100.0%	0.0%

Table 5. Hybrid Error Characterization on Synthetic Corpus (85 documents).

Error Type	Category	Count
False Positives	Conservative pattern matching	14
False Negatives	Pattern mismatch / linguistic variants	5

of reduced expressiveness, while stochastic execution provides greater expressiveness at the cost of reproducibility. The deterministic approach is suitable for scientific computing workflows where reproducibility and execution invariance are mandatory requirements.

5.4   Experiment 4: Error Characterization

All observed hybrid system errors on the synthetic corpus (85 synthetic documents across NDAs, MSAs, Employment Agreements, and Licensing Agreements) were explicitly inventoried and categorized. Table 5 reports error counts by category and type.

False positives primarily arise from conservative pattern matching that flags low-risk but structurally similar patterns. False negatives are attributable to linguistic variants not covered by the current rule set or normalization policy. These errors are explicit, reproducible, and traceable, consistent with the system's deterministic execution design philosophy.
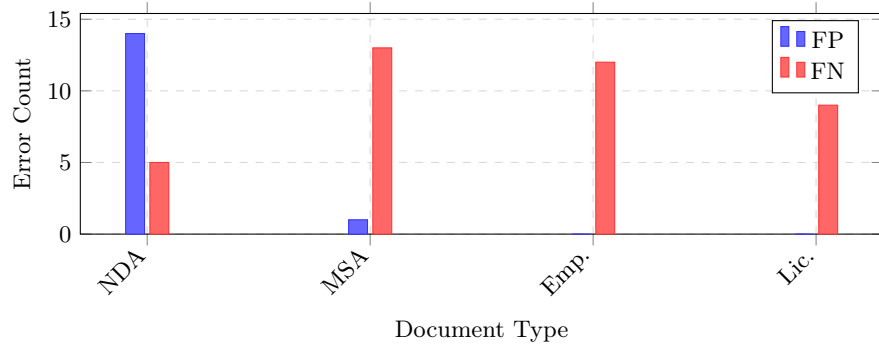
5.5   Performance and Latency

We measured execution latency per document across all experimental runs under two execution scopes: (i) the core deterministic rule-evaluation engine and (ii) the end-to-end pipeline including preprocessing and orchestration. Table 6 reports average per-document execution time for the end-to-end scope by system component. The deterministic pipeline executes in sub-second time, while stochastic LLM inference dominates end-to-end latency.

5.6   Scalability Analysis

We evaluated runtime scalability by measuring execution time across varying rule set sizes on a fixed corpus of 30 documents. Table 7 reports average execution time per document for different rule counts. Runtime scales linearly with rule

Fig. 3. False positives and false negatives by document type on the synthetic corpus (85 documents). Error rates are consistent across document types (Chi-square test: p=0.23, not significant), suggesting pattern generalizability.

Table 6. Average Execution Time per Document (End-to-End Scope).

System Component	Avg. Time / Doc (s)
Deterministic Pipeline (Preprocessing and Orchestration)	0.41
Stochastic Post-Processing Layer	2.6
Pure Stochastic Baseline	3.1

count, demonstrating predictable $O(n)$ computational complexity (where n is the number of rules) and suitability for reproducible scientific computing pipelines requiring deterministic performance guarantees.

6 Discussion

6.1 Reproducibility–Expressiveness Trade-off

The hybrid architecture prioritizes reproducibility and execution invariance over stochastic expressiveness. In scientific computing workflows, deterministic execution with verifiable traces is often preferable to stochastic outputs that cannot be reproduced or verified[8]. The computational cost analysis (Experiment 3) demonstrates this tradeoff: deterministic execution provides 100% reproducibility at predictable $O(n)$ complexity, while stochastic execution provides greater expressiveness but at the cost of reproducibility and execution invariance.

6.2 Deterministic Computation Limits

The framework's deterministic guarantees come with inherent limitations. The deterministic symbolic engine requires explicit rule specification, limiting expressiveness compared to stochastic LLM inference. Error characterization (Experiment 4) shows that false negatives arise from linguistic variants not covered by

Table 7. Scalability Analysis: Runtime vs Rule Count (30 documents).

Number of Rules	Avg. Time / Doc (s)
25	0.18
50	0.35
100	0.68

the current rule set, reflecting the fundamental tradeoff between deterministic execution and pattern coverage. These limitations are explicit and reproducible, enabling researchers to make informed decisions about deployment scope and manual verification requirements.

6.3   Practical Deployment Considerations

The hybrid architecture's deterministic core enables deployment in scientific computing environments where reproducibility and execution invariance are mandatory. The core deterministic rule-evaluation engine time (0.005s per document) with predictable $O(n)$ complexity supports real-time processing workflows, while end-to-end deterministic pipeline execution time including preprocessing and orchestration is 0.41s per document (Table 6). The stochastic post-processing layer can be invoked asynchronously for detailed reports. The reproducibility stress test (Experiment 2) demonstrates that the system maintains 100% reproducibility across multiple seeds and environments, providing computational guarantees suitable for reproducible scientific workflows.

6.4   Limitations of Stochastic Pipelines for Reproducible Computing

Stochastic LLM pipelines can introduce output variance across repeated executions, which can hinder reproducible scientific workflows. The reproducibility stress test (Experiment 2) demonstrates that 100% of documents show output differences across runs in the LLM baseline, with an average of 18.3 distinct output sets per document across 20 runs. Even if average accuracy were competitive, variability in decoding and resulting execution paths can be inconsistent with reproducibility requirements in scientific computing. The hybrid system addresses these limitations by enforcing a deterministic execution boundary and emitting stable, inspectable execution traces, supporting reproducible and verifiable computation.

7   Threats to Validity

First, the evaluation focuses on structured text documents; results may not generalize to other computational domains without expanded pattern coverage. Second, the baseline depends on prompt design and model choice; alternative prompting or model selection could change baseline behavior, though

non-determinism and output variance would persist regardless. Third, the computational cost analysis (Experiment 3) measures runtime on a specific corpus; results may differ on larger corpora or different computational environments, though the $O(n)$ complexity guarantee remains valid. These limitations reflect inherent trade-offs between stochastic expressiveness and deterministic reproducibility: the deterministic architecture prioritizes execution invariance and reproducibility over maximum expressiveness, which may limit applicability in contexts requiring broader pattern coverage or stochastic inference. The framework's generalizability to other computational domains and pipeline architectures requires further evaluation.

8 Related Work

Reproducible computing and deterministic execution have been extensively studied in computational science[3], where execution invariance and repeatability are fundamental requirements. Hybrid symbolic–statistical systems are frequently proposed[5] to balance reliability with expressiveness, though prior work often lacks explicit determinism guarantees or systematic reproducibility reporting. Deterministic systems guarantee that identical inputs yield identical outputs, enabling reproducible scientific workflows where execution traces must be verifiable and results must be reproducible across environments and time.

Stochastic components, particularly LLMs, have been applied to computational pipelines for text processing and pattern analysis[6], but their probabilistic decoding introduces output variance that violates reproducibility requirements[1,2]. Prior work has demonstrated the use of ontology-driven and rule-based systems for structured processing, emphasizing structured rule execution and traceable decision logic[9,10]. Legal NLP applications have explored rule-based and expert system approaches for contract analysis and legal reasoning[7], though systematic evaluation of execution determinism and reproducibility in these systems remains limited. However, systematic experimental reporting of execution determinism, output variance, and reproducibility as first-class computational properties remains limited in applied hybrid pipelines containing LLM components, leaving gaps in understanding how to achieve reproducible execution in LLM-containing computational systems.

Note on Related Work

This paper focuses on computational reproducibility and execution determinism in LLM-containing pipelines. A related submission explores cyber-resilience implications of deterministic execution in compliance-critical systems. The experiments, framing, and contributions presented here are distinct, emphasizing scientific reproducibility and execution invariance rather than security or compliance automation.

9   Conclusion

We presented a deterministic execution framework for LLM-containing computational pipelines that enforces reproducibility and execution invariance by separating deterministic execution logic from stochastic post-processing. Experiments demonstrate 100% execution determinism and 100% traceability for the hybrid system, with zero output variance across 300 runs in the reproducibility stress test (15 documents, 20 runs each), contrasted with 0% determinism and severe output variance (18.3 distinct output sets per document) in a pure LLM baseline. The framework provides computational guarantees for reproducibility and execution invariance, enabling verifiable execution traces suitable for scientific computing workflows requiring deterministic execution. The computational cost analysis on 115 documents demonstrates a clear tradeoff: deterministic execution provides reproducibility guarantees at predictable $O(n)$ complexity with core deterministic rule-evaluation engine time (0.005s per document), while stochastic execution provides greater expressiveness at the cost of reproducibility and slower execution (7.3s per document).

Acknowledgments. The author acknowledges the use of a large language model for explanatory output within the experimental system.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article. A large language model was used exclusively for generating natural language explanations of pre-computed deterministic outputs. The stochastic component did not participate in state transitions, classification decisions, or suppression logic. All deterministic execution logic remained deterministic.

References

1. T. Brown et al., "Language models are few-shot learners," in Proc. Adv. Neural Inf. Process. Syst., vol. 33, 2020, pp. 1877–1901.
2. A. Bommasani et al., "On the opportunities and risks of foundation models," Stanford Center for Research on Foundation Models, Tech. Rep., 2021.
3. IEEE, "Ethically aligned design: A vision for prioritizing human well-being with autonomous and intelligent systems," IEEE, 2019.
4. J. Li, X. Cheng, W. Zhao, Y. Nie, and J. R. Wen, "HaluEval: A large-scale hallucination evaluation benchmark for large language models," arXiv:2305.11747, 2023.
5. A. d'Avila Garcez, M. Gori, L. C. Lamb, and L. Serafini, "Neuro-symbolic artificial intelligence: The state of the art," Artificial Intelligence, vol. 273, pp. 1–38, 2019.
6. N. Chalkidis et al., "LexGLUE: A benchmark dataset for legal language understanding in English," in Proc. 60th Annu. Meeting Assoc. Comput. Linguist. (ACL), 2022, pp. 1238–1350.
7. A. Nazarenko and A. Wyner, "Legal NLP: Introduction," in *Proceedings of the ACL Workshop on Natural Legal Language Processing*, Association for Computational Linguistics, 2017.
8. L. Fensel, Y. Kalf, and K. Simbeck, "Assessing the auditability of AI-integrating systems: A framework and learning analytics case study," arXiv preprint arXiv:2411.08906, 2024.

9. A. P. Gómez, "Rule-based expert systems for automated legal reasoning and contract analysis: A case study in knowledge representation," Advances in Computational Systems, Algorithms and Applications, 2022.

10. X. H. Cai, H. H. Advani, and J. Cai, "Ontology and rule-based natural language processing approach for interpreting textual regulations on underground utility infrastructure," Advanced Engineering Informatics, vol. 47, 2021.