Sigloch, Paul; Benzmüller, Christoph

Neuro-Symbolic Verification of LLM Outputs for Data-Sensitive Domains (extended preprint)

Neuro-Symbolic Verification of LLM Outputs for Data-Sensitive Domains (extended preprint)

Paul Sigloch[1] and Christoph Benzmüller[1,2]

[1] University of Bamberg, Bamberg, Germany
[2] Free University of Berlin, Berlin, Germany
paul.sigloch@uni-bamberg.de, christoph.benzmueller@uni-bamberg.de

Abstract. LLMs deployed in high-stakes domains face fundamental reliability challenges: hallucinations, inconsistencies, and privacy vulnerabilities introduce unacceptable risks where errors carry legal, financial, or safety consequences. This paper presents a hybrid verification architecture combining formal symbolic methods with neural semantic analysis to provide complementary guarantees for LLM-generated content.
This architecture employs logical reasoning for input verification, leveraging completeness properties to provide decidable guarantees on structured requirements. For output validation, embedding-based semantic similarity detects contextual hallucinations where formal methods lack expressiveness. This separation is realized in a parallel, actor-based pipeline, addressing limitations of prompt-based self-verification approaches, which inherit the distributional biases that produce hallucinations.
The proposed architecture and type-aware verification method are validated with HAIMEDA, a real-world medical device damage assessment reporting system developed through Action Design Research. Evaluation shows hallucination detection rates of over 83% for structured entities and 72% for semantic fabrications, with a 30% reduction in report creation time, demonstrating that neuro-symbolic architectures can provide principled safeguards for LLM deployment in data-sensitive domains.

Keywords: Neuro-symbolic AI · LLM verification · Formal verification · Trustworthy AI · Hybrid AI · Actor model · AI system architecture

1 Introduction

Large language models (LLMs) have demonstrated strong abilities in a variety of tasks, although their usage within high-risk areas remains challenging. In professional environments where information accuracy is paramount, such as medical diagnosis, legal reviews, and regulatory affairs, the limitations of purely neural approaches pose significant risks. Hallucinations, or generating credible but false information, occur at rates of 3% to 10% even in state-of-the-art models [19,21]. For domains where errors carry legal, financial, or safety consequences, such shortcomings are unacceptable.

Beyond accuracy concerns, many data-sensitive workflows require local processing to satisfy regulatory, institutional, or client-imposed constraints on data

transmission [12,22]. These issues point towards a limitation of general-purpose LLMs, regardless of size, which lack the ability to preserve information integrity for specialized settings. The effectiveness provided by approaches like retrieval-augmented generation (RAG) or chain-of-thought prompt design cannot provide the necessary guarantees for specialized settings like high-stakes applications. A more principled approach requires integrating the complementary strengths of different AI paradigms.

Hybrid AI systems address this gap by combining sub-symbolic methods, specifically neural networks that excel at pattern recognition, natural language processing, and handling novel inputs, with symbolic methods that provide explicit reasoning, rule enforcement, and verifiable guarantees [14,26]. This integration enables architectures where LLM-generated content can be systematically verified against domain constraints, factual databases, and logical consistency requirements before reaching end users.

Despite growing recognition of hybrid AI's potential, practitioners still face an architectural integration gap. Existing research has produced valuable individual techniques, including neuro-symbolic reasoning systems, verification frameworks, and domain-specific architectures, but validated end-to-end verification architectures for data-sensitive domains remain scarce. Questions of how to combine symbolic and sub-symbolic components to preserve information integrity at system level still lack systematic answers [4,33].

This paper makes two technical contributions for verifiable LLM deployment in data-sensitive domains. First, we propose a verification architecture for LLM-assisted generation in data-sensitive domains, combining tableaux-based input validation, actor-based fault isolation, and a type-aware post-generation verification method that uses deterministic symbolic checks for structured claims and semantic similarity scoring for free-text claims. Second, we validate this architecture through HAIMEDA, a locally deployed medical device assessment reporting system that demonstrates strong detection of unsupported content and measurable improvements to workflow.

2 Related Work

Research involving hybrid verification for content produced using LLMs pertains to three areas: neuro-symbolic architectures, methods of hallucination detection, and output verification systems. In addition, implementation-oriented studies of AI systems are useful for understanding how such architectures are engineered in practice.

Neuro-Symbolic AI Architectures. Neuro-symbolic integration combines neural learning with symbolic reasoning [14]. Key frameworks include Logic Tensor Networks [3], DeepProbLog [25], and Scallop [24]. Though these models are useful in the area of algorithmic integration, they essentially imply architecture modifications or training in a way that limits their adaptability to pre-trained LLMs.

LLM Hallucination Detection. Detecting factual errors in LLM outputs remains challenging. Self-consistency methods [37] sample multiple responses and measure agreement, while retrieval-augmented verification [13] cross-references outputs against external knowledge bases. However, these methods inherit distributional biases from the models they verify [19,21], limiting reliability in high-stakes domains. Knowledge editing techniques [18] attempt post-training correction but struggle with complex or context-dependent hallucinations.

Output Verification and Guardrails. Runtime verification tools provide practical safeguards: Guardrails AI [9] and NeMo Guardrails [30] implement programmable validation, while constrained decoding [38] enforces structural compliance. These address format and policy checks but offer limited semantic verification against source content. Domain-specific systems like TrustKG [8] and Teriyaki [5] demonstrate verification in specialized contexts but remain point solutions without generalizable architectural patterns.

Overall, the current state of the art has established efficient methods for neuro-symbolic reasoning, mitigation of hallucinations, and runtime protection. However, a comprehensive architectural approach combining formal verification and semantic validation for LLM deployment in data-sensitive applications remains absent.

3 Verification Architecture and Design Rationale

Deploying LLMs in data-sensitive domains raises verification challenges that guided the design of the proposed architecture. In workflows for report generation in data-sensitive domains, input validation is a consistency problem over interacting constraints. Output verification requires mechanisms for both symbolic and neural analysis, ideally with fault isolation to prevent cascading failures. Privacy considerations in regulated domains often necessitate local deployment rather than cloud-based APIs. Our architecture addresses these challenges by combining tableaux-based verification for input constraints, actor-model concurrency for fault-isolated parallel processing, and neuro-symbolic integration for output verification.

The architecture is guided by four design principles: local-first processing for data sovereignty [22], modular separation of symbolic and neural components [29], actor-based fault isolation with functional immutability for robustness [2,16,20], and type-aware claim verification, i.e., deterministic symbolic checks for structured claims and neural semantic checks for free-text claims [14,33].

3.1 Tableaux Logic for Input Verification

Input validation in data-sensitive workflows is a consistency problem over interacting constraints. In the medical-device assessment workflow instantiated later in HAIMEDA, the report is assembled section by section, and these sections differ substantially in their validation demands: some are largely narrative, some follow

a technical schema, some contain legally consequential assessments, and some primarily document supporting evidence. Mandatory metadata and contradiction states must therefore be checked before any LLM call, but the exact requirement profile varies with the target section.

This variability calls for a validation mechanism that can express both cumulative obligations and alternative sufficiency paths within a single declarative scheme. Tableaux methods are well suited to this role because they decompose formulas into conjunctive (α) obligations that must hold together and disjunctive (β) branches that represent admissible alternatives. For propositional and decidable fragments, this yields sound and complete procedures [32,36].

The architecture encodes validation as a configurable hierarchy of declarative conditions, so higher-order conditions can reason over outcomes of lower-order ones. Let $\mathcal{C}_{all}$ denote the universe of all condition identifiers in the current chapter profile. *Core conditions* $\mathcal{C} \subseteq \mathcal{C}_{all}$ evaluate atomic predicates programmatically (e.g., metadata presence or minimal title quality). *Meta conditions* and *aggregate conditions* form the higher-order set $\mathcal{H} = \mathcal{C}_{all} \setminus \mathcal{C}$: meta conditions express set-theoretic relationships over condition groups, while aggregate conditions compute statistics over condition attributes.

Let $\mathcal{C}_{sat} \subseteq \mathcal{C}_{all}$ denote the conditions observed to hold, $\mathcal{C}_{req} \subseteq \mathcal{C}$ the required core conditions, and $\mathcal{C}_{should\_sat}$ the conditions expected to hold according to the ruleset. The *positive set* $\mathcal{C}_{pos}$ contains conditions whose observed and expected satisfaction states agree, while $\mathcal{C}_{neg} = \mathcal{C}_{all} \setminus \mathcal{C}_{pos}$ captures polarity mismatches; BuildPositiveSet computes this alignment from $\mathcal{C}_{sat}$ and the ruleset's expectation declarations. Finally, $\mathcal{C}_{eval}$ tracks which conditions have been processed, enabling prerequisite-ordered evaluation: a higher-order condition $h \in \mathcal{H}$ is evaluated only once all conditions it depends on appear in $\mathcal{C}_{eval}$.

A mandatory-consistency gate then checks $(\mathcal{C}_{req} \cap \mathcal{C}_{eval}) \subseteq \mathcal{C}_{pos}$, meaning that all evaluated required conditions must be positively aligned, while an aggregate rule bounds total polarity mismatch via $|\mathcal{C}_{neg}| \leq \tau$. These rules exceed flat `if-else` validation because their outcome depends on relationships between dynamically constructed condition sets.

The domain is therefore naturally set-theoretic: the validation state is represented as a universe $\mathcal{U}$ of named condition-ID sets ($\mathcal{C}$, $\mathcal{C}_{sat}$, $\mathcal{C}_{req}$, $\mathcal{C}_{eval}$, $\mathcal{C}_{pos}$, $\mathcal{C}_{all}$). Here, $\mathcal{C}_{all} = \mathcal{C} \cup \mathcal{H}$ provides the reference universe for complement-based reasoning, while derived mismatch sets such as $\mathcal{C}_{neg}$ are computed from it as needed. Tableaux decomposition maps directly to set operations: α-expansion behaves as intersection, β-expansion as union, and negation as complement over $\mathcal{C}_{all}$ [36]. This yields deterministic, auditable gatekeeping before generation while preserving declarative domain configuration. Algorithm 1 summarizes the engine. The mandatory-consistency gate above acts as a hard stop (generation blocked if unmet), whereas the aggregate polarity-mismatch rule can produce warning feedback without blocking, illustrating graded responses within the same engine.

Algorithm 1 Tableaux-Based Input Validation.

Require: Input data I, Ruleset R with core conditions $\mathcal{C}$ and higher-order conditions $\mathcal{H}$

// Phase 1: Programmatic evaluation of core conditions

$\mathcal{C}_{sat} \leftarrow \{c \in \mathcal{C} \mid \textsc{Eval}(c, I) = \top\}$
$\mathcal{C}_{req} \leftarrow \{c \in \mathcal{C} \mid c.\mathit{required} = \top\}$
$\mathcal{C}_{eval} \leftarrow \textsc{InitializeEvaluatedSet}(R, \mathcal{C})$

// Phase 2: Tableaux reasoning over condition relationships

$\mathcal{C}_{pos} \leftarrow \textsc{BuildPositiveSet}(\mathcal{C}_{sat}, R)$
$\mathcal{C}_{all} \leftarrow \mathcal{C} \cup \mathcal{H}$
$\mathcal{U} \leftarrow \{\mathcal{C}, \mathcal{C}_{all}, \mathcal{C}_{sat}, \mathcal{C}_{req}, \mathcal{C}_{eval}, \mathcal{C}_{pos}\}$ ▷ Universe of sets
for $h \in \mathcal{H}$ **do**
  if $\textsc{Prerequisites}(h) \not\subseteq \mathcal{C}_{eval}$ **then continue**
  $\varphi_h \leftarrow \textsc{BuildFormula}(h, \mathcal{U})$ ▷ e.g., $\mathcal{C}_{req} \subseteq \mathcal{C}_{pos}$
  $[\![\varphi_h]\!] \leftarrow \textsc{TableauxSolve}(\varphi_h, \mathcal{U})$ ▷ α/β-decomposition
  if $[\![\varphi_h]\!] \neq \emptyset$ **then**
    $\mathcal{C}_{sat} \leftarrow \mathcal{C}_{sat} \cup \{h\}$
  end if
  $\mathcal{C}_{eval} \leftarrow \mathcal{C}_{eval} \cup \{h\}$ ▷ record processed rule
end for

// Phase 3: Trigger actions based on satisfaction state

for action a with trigger condition τ_a and event $e \in \{\mathit{sat}, \mathit{unsat}\}$ **do**
  if $(\tau_a \in \mathcal{C}_{sat}) = (e = \mathit{sat})$ **then** ▷ activate iff actual satisfaction state matches declared trigger event
    $\textsc{Execute}(a)$
  end if
end for
return $(\mathcal{C}_{req} \cap \mathcal{C}_{eval}) \subseteq \mathcal{C}_{pos}$, collected feedback

For each higher-order condition h, BuildFormula maps the declarative rule to a set-theoretic formula φ_h over the universe $\mathcal{U}$. TableauxSolve then evaluates φ_h by decomposing conjunctions, disjunctions, and negations into the corresponding intersection, union, and complement operations over the condition sets. The resulting denotation $[\![\varphi_h]\!]$ is non-empty if and only if the rule is satisfied under $\mathcal{U}$. Trigger actions then fire when the observed satisfaction state matches the action's declared event. Algorithm 1 shows a domain-specific instance of the domain-independent engine. Proof order, condition hierarchy, and trigger actions are supplied declaratively.

Example 1: Context-Sufficiency Gate. Before invoking the LLM to generate a damage narrative, the system checks whether the input provides sufficient grounding to avoid hallucinated device specifics. Two structurally distinct evidence combinations are accepted. A meta condition c_{ready} encodes this as

$$\varphi_{c_{\text{ready}}} \equiv \big(c_{\text{dev\_type}} \wedge c_{\text{serial}} \wedge c_{\text{category}}\big) \vee \big(c_{\text{category}} \wedge c_{\text{damage\_desc}} \wedge c_{\text{party}}\big),$$

where the left disjunct captures full technical identification and the right captures legally sufficient narrative grounding. The tableau opens one branch per disjunct. If the serial number is absent, the first branch fails. However, the second branch can still satisfy c_{ready}. Generation is blocked only if both branches fail. This prevents the need to enumerate all admissible field combinations procedurally.

Example 2: Keyword-Triggered Contextual Obligation. When Phase 1 detects a safety-critical term in the input text, the validation logic tightens the requirements: both a failure-mode classification and a regulatory risk category must be present. If no such term is detected, this obligation disappears. A meta condition c_{safety} encodes this as a material implication:

$$\varphi_{c_{\text{safety}}} \;\equiv\; \neg\, c_{\text{crit\_kw}} \;\vee\; \big(c_{\text{failure\_mode}} \wedge c_{\text{risk\_class}}\big).$$

The first branch covers cases in which no critical keyword is present. The second branch checks if both required fields are aligned correctly. If the keyword is detected ($c_{\text{crit\_kw}} \in \mathcal{C}_{sat}$), the first branch fails and the second must succeed. The example shows how a single declarative formula can express context-dependent requirement tightening without hard-coding every keyword-triggered exception.

3.2 Actor-Based Concurrency Model

Verification pipelines in data-sensitive domains must execute multiple heterogeneous analysis streams, including symbolic rule validation, entity extraction, and neural embedding comparisons. In order to ensure that a failure in one component (e.g., a timeout during external neural inference) does not corrupt the broader symbolic validation state, these streams must execute concurrently without shared mutable state. These requirements closely align with the actor model [16], a paradigm where independent lightweight processes communicate strictly through asynchronous message passing.

The proposed architecture adopts the actor model to guarantee strong process isolation and systematic fault recovery. In traditional procedural architectures, defensive programming against unpredictable machine learning API failures or memory-bound tensor operations often leads to fragile error handling or cascading crashes. Instead, the actor model utilizes supervision hierarchies: supervisor actors monitor worker processes and automatically restart them from a known clean state upon failure [2]. This approach drastically simplifies error handling while maximizing system resilience.

To instantiate this design, the architecture leverages Elixir and the Erlang Virtual Machine (BEAM). The BEAM provides native preemptive scheduling for millions of isolated, lightweight actors, making it uniquely suited for scaling complex, multi-branched verification workloads [7]. Functional immutability within Elixir guarantees reproducible data flows across concurrent evaluation pipelines, eliminating race conditions even when hundreds of verification rules process a document simultaneously. For example, if an embedding worker fails (e.g., due to a timeout), supervision will only restart that worker, while the

symbolic verification branches will continue with their preserved state. Likewise, the disjunctive (β) branches can be explored independently and in parallel.

3.3 Hybrid Integration Strategies

The hybrid strategies presented here address a fundamental limitation: formal methods provide decidable guarantees but lack expressiveness for unconstrained natural language, while neural methods capture semantic similarity but offer only probabilistic confidence. The architectural contribution is a principled decomposition assigning each verification mode to tasks matching its guarantee type. In domains with a more formal structure, such as mathematical proofs or program code synthesis, symbolic verification may suffice. However, the patterns discussed below are relevant to domains where formal methods alone cannot fully express the verification process.

Hybrid verification systems integrate symbolic and sub-symbolic components through three patterns [14,33]: pre-processing applies symbolic validation before neural inference, post-processing verifies outputs after generation, and parallel integration executes both concurrently. Each pattern addresses a specific challenge related to verifying LLM-assisted workflows.

Pre-processing Gates. Symbolic pre-processing establishes verification gates ensuring only constraint-satisfying inputs proceed to neural generation. This addresses efficiency concerns where malformed input wastes computational resources, but more fundamentally provides decidable guarantees before committing to costly inference. As detailed in Section 3.1, tableaux-based validation enables the verification of complex, interdependent constraints with completeness guarantees unavailable from procedural validation alone.

Post-processing Verification. LLMs lack mechanisms ensuring factual grounding, producing hallucinations, i.e., content that satisfies formatting constraints but lacks source support [21]. Neural self-verification inherits distributional biases from training data [19], motivating external verification mechanisms. Hybrid post-processing uses different methods for different entity types. For structured entities, such as dates, identifiers, or numeric values, symbolic comparison provides deterministic verification. For instance, a date either matches the source or it does not. Semantic content requires neural similarity assessment, with embedding-based comparison capturing relatedness beyond lexical matching. This type-dependent strategy applies each method where it provides strongest reliability guarantees.

Illustrative case (hybrid post-processing). Assume the input contains `BF-1HTJ0` and `2024-11-03`, while the generated output contains `BF-1HTJO` and `03.11.2024`. Symbolic matching accepts the date as format-equivalent but flags the identifier mismatch deterministically. Statement-level claims are then evaluated with neural similarity metrics, and unresolved cases are forwarded as interactive corrections.

Parallel Integration and Hybrid Narrowing. Parallel integration executes symbolic and neural processing concurrently, reducing latency while combining complementary strengths. In RAG, for instance, symbolic query decomposition extracts structured entities enabling targeted metadata queries, while neural encoding captures semantic intent for similarity search. A key architectural pattern emerges: symbolic filtering narrows the search space before neural ranking. This hybrid narrowing strategy is particularly valuable for locally deployed RAG systems because smaller models operating within limited context windows are more sensitive to retrieval noise than larger, cloud-hosted alternatives. By pre-filtering irrelevant documents before computing neural similarity, hybrid narrowing provides focused context, enabling smaller models to achieve response quality similar to that of larger systems. This is a critical capability when data sovereignty requirements prevent external transmission.

Illustrative case (parallel hybrid narrowing). For a query about a specific claim file, symbolic extraction first resolves candidate report IDs and restricts corpus scope to the relevant section of the document. In a damage assessment report, individual sections differ substantially in character: introductory and contextual chapters are largely narrative, technical data sections follow a structured schema with strict field requirements, jurisdictionally relevant chapters such as expert opinion or liability conclusions carry specific legal obligations, and supporting evidence chapters may consist of attached proofs or measurement records. This heterogeneity means that without structural filtering, retrieval may return context from the wrong section type. For example, a technically correct sentence from a data sheet is not valid grounding for a legal assessment chapter. Symbolic pre-filtering by section type narrows the candidate pool to structurally appropriate documents before neural ranking, reducing noise and token usage while ensuring that the retrieved context matches both the content and the formal register of the target section.

Architectural Requirements. Computing semantic similarity requires vector representations. Vector databases provide an efficient approximate nearest-neighbor search, which is useful for comparing generated statements against reference corpora. The choice of vector storage stems from the need for semantic rather than lexical matching, as keyword searches cannot identify paraphrased or conceptually related content. Multilingual sentence encoders [31] extend this capability across languages. Local model serving addresses privacy constraints; local inference architectures ensure complete data sovereignty and accept computational constraints that favor smaller, fine-tuned models. Thus, the trade-off between model capability and deployment flexibility becomes an architectural decision rather than an operational one.

3.4 Privacy-Preserving Architecture

Data-sensitive domains impose privacy requirements that constrain architectural options [12]. Medical assessments, legal analyses, and financial documents often

contain personally identifiable information (PII) and other regulated content, motivating a privacy-by-design approach [6]. Although cloud-based LLM APIs can in principle be made GDPR-compliant through data-processing agreements, this addresses only one layer of the constraint space. In professionally regulated deployments, additional legal, institutional, or contractual constraints may still preclude external data transmission. HAIMEDA is one such case, discussed in Section 4.

The architecture implements three strategies. Local-first processing [22] ensures sensitive content never leaves the deployment environment: all inference, embedding, and verification execute locally, accepting computational constraints for complete data sovereignty. Data flow isolation maintains sensitivity boundaries: verification results propagate as structured assessments (scores, flags) rather than raw content. Vector embeddings constitute lossy transformations providing information-theoretic protection for indexed collections. Sanitization pipelines enable model improvement: entity detection replaces sensitive elements (names, identifiers, locations) with semantically consistent placeholders before external processing.

These mechanisms integrate through the actor model's process isolation. Sensitive processing occurs within dedicated supervision subtrees with restricted communication, ensuring failures cannot leak protected information. Audit logging captures decisions without recording sensitive content, supporting regulatory traceability.

3.5 Domain-Specific Model Adaptation

Privacy constraints favoring local deployment create a tension: larger models offering superior general capability require computational resources exceeding typical deployment environments, while smaller models fit local hardware but may underperform on specialized tasks. Domain-specific fine-tuning resolves this tension by adapting smaller models to achieve competitive performance within constrained domains [1].

This adaptation strategy provides architectural benefits beyond reduced computational requirements. Training on domain-specific corpora aligns the model's behavior with professional conventions, terminology, and output expectations, which general-purpose models handle inconsistently. When sensitive training data precludes cloud-based fine-tuning services, local training preserves digital sovereignty throughout the model lifecycle, addressing sovereign cloud requirements in regulated environments, rather than just during inference. Parameter-efficient techniques such as Low-Rank Adaptation (LoRA) [17] enable iterative refinement cycles on local hardware, supporting systematic improvement through repeated training without external dependencies.

4 HAIMEDA: Hybrid Verification System

The preceding architectural principles require empirical validation within a domain with genuine constraints. Medical device damage assessment is one such domain, combining PII, health-related data, and legally consequential documentation. HAIMEDA (Hybrid AI for Medical Device Assessment) instantiates the proposed architecture for this setting through fine-tuned language models and complementary symbolic and sub-symbolic verification components. The system employs hybrid verification to detect hallucinations and ensure information integrity without relying on LLM self-assessment. Development followed an iterative refinement process through Action Design Research (ADR) in collaboration with a domain expert [35]. Implementation resources and data-availability constraints are summarized in Appendix A.2.

In this setting, the deployment architecture is constrained not only by privacy regulation but also by legal and institutional requirements. Although cloud-based LLM APIs can in principle be made GDPR-compliant through data processing agreements (DPAs) with providers [12], this addresses only one layer of the constraint space. In the HAIMEDA context, the domain expert operates as a publicly appointed and sworn expert witness (*öffentlich bestellter und vereidigter Sachverständiger*) under German law, a status that creates confidentiality obligations toward parties whose data appears in assessment cases, including patients, device operators, and insurers. Together with requirements arising from the Medical Devices Regulation [11], its German implementing act [15], and contractual restrictions imposed by commissioning insurers and public authorities, these constraints preclude external transmission of case-sensitive materials. HAIMEDA was therefore designed for local-first processing as a legal and contractual deployment requirement rather than a mere privacy preference.

4.1 System Architecture

HAIMEDA implements the design rationale from Section 3 through a modular architecture coordinated by a central orchestration layer. The implementation realizes actor-model concurrency through Elixir on the Erlang VM. Neural components integrate via language bridging to Python, maintaining functional purity in orchestration while accessing machine learning libraries. Three primary modules compose the system: the Report Authoring Module (RAM) provides the interface for fine-tuned language models, handling report generation through structured templates with iterative revision incorporating verification feedback; the Research and Investigation Module (RIM) implements hybrid narrowing for domain knowledge retrieval, detailed in Section 4.4; and the Information Integrity Verification Module (IIVM) realizes multi-stage verification through complementary symbolic and neural components, detailed in Sections 4.2 and 4.3. Figure 1 illustrates module coordination during report creation.

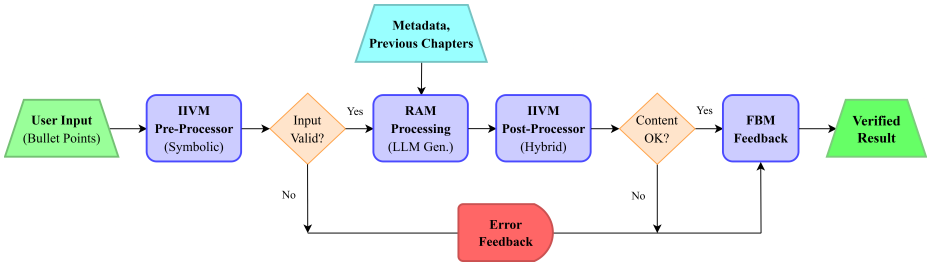


Fig. 1. The report chapter creation workflow in HAIMEDA. Symbolic pre-processing validates the input and its result flows to the orchestration layer, which either gates or permits LLM generation; the post-processor then operates independently on the generated output. The Feedback Module (FBM) communicates validation results and errors to the user.

4.2 Symbolic Pre-processing

The pre-processing pipeline instantiates the tableaux-based validation (Section 3.1) for the medical device domain. By externalizing domain rules into a declarative configuration, HAIMEDA maps abstract verification logic to concrete documentation requirements, such as product types, client identifiers, or mandatory metadata, without modifying the underlying engine. Applying Algorithm 1, critical violations securely terminate the generation process before any LLM inference occurs, whereas minor omissions resolve into warnings that permit continuation while surfacing structured feedback to the domain expert.

4.3 Hybrid Post-processing

Post-processing verifies generated content against source material to detect hallucinations, instantiating the hybrid strategy from Section 3.3 by applying symbolic and neural methods where each is most reliable (see Figure 2, panel a). The pipeline starts with deterministic entity extraction from input and generated output, where entity types include dates, numeric values, identifiers, key phrases, and complex statements. To improve robustness across formatting differences, each entity is represented in multiple variants.

Verification strategy varies by entity type, following the hybrid approach from Section 3.3. Structured entities undergo symbolic comparison through pattern matching, which provides deterministic verification. Semantic content, on the other hand, requires a neural similarity assessment using multilingual sentence embeddings. Statement matching therefore uses a five-metric similarity scheme combining TF-IDF similarity, domain-specific semantic similarity, normalized Euclidean similarity, token overlap, and keyword overlap; their weighted aggregation yields a combined score, while inter-metric agreement yields a confidence score. Bidirectional comparison detects two failure modes: missing information (input entities absent from output) and potential hallucinations (output entities unsupported by input), where detected discrepancies trigger interactive corrections.

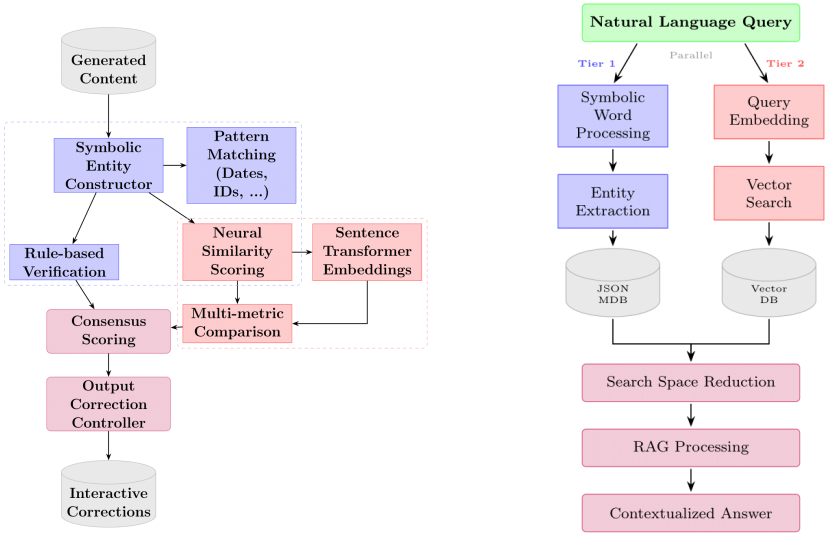
Coverage metrics quantify both modes with type-dependent weights reflecting verification confidence, assigning highest weight to symbolically verifiable structured entities and lower weight to semantically assessed content.

4.4 Hybrid Information Retrieval

RIM instantiates the parallel hybrid-narrowing strategy from Section 3.3 by executing symbolic and neural retrieval tiers concurrently (Figure 2, panel b).

The symbolic tier uses pattern-based query decomposition to extract domain terms (e.g., stakeholder references, device categories, manufacturers, and case identifiers). Extracted terms are mapped to schema associations, and confidence ranking distinguishes direct from tentative matches.

The neural tier computes query embeddings for vector similarity search within the scope constrained by the symbolic tier. When structured queries identify relevant report identifiers, only corresponding chapters enter similarity computation. Dynamic result sizing uses configurable token limits to keep context within model capacity while maximizing informational density.



(a) Post-processing verification (IIVM) **(b)** Hybrid retrieval workflow (RIM)

Fig. 2. Panel(a) presents IIVM post-processing: symbolic entity construction (blue) supports rule-based verification, neural similarity scoring (orange) evaluates pattern-matched content, and consensus scoring with output correction (red) integrates both streams to detect hallucinated or missing content. Panel(b) presents RIM retrieval, where parallel symbolic (Tier 1) and neural (Tier 2) pipelines are merged to reduce search space before RAG.

5 Evaluation

HAIMEDA was evaluated through systematic assessment of verification effectiveness, computational performance, and practical utility in professional workflows. All experiments were conducted on consumer hardware representative of deployment conditions: AMD Ryzen 9 5900X (12 cores), 64GB RAM, and NVIDIA RTX 4090 GPU with 24GB VRAM, reflecting the architecture's local-deployment objective. Practical evaluation with a domain expert revealed a 30% reduction in report creation time, yielding a System Usability Scale score of 62.5. Detailed methodology and extended results are available in Appendix A.1 and in the external technical documentation referenced in Appendix A.2. The domain expert is a publicly appointed and sworn expert witness and certified medical device engineer in Germany, with extensive professional experience in standardized damage assessment for insurance companies, courts, and public authorities. His assessments carry legally binding status; as such, data confidentiality is a statutory obligation rather than a preference, which is why local deployment is mandatory in this context rather than optional (see Section 3.4). Collaboration followed the Action Design Research process [35] and encompassed requirement elicitation, iterative prototype feedback, and final evaluation.

5.1 Experimental Configuration

Content generation employs fine-tuned variants of `Llama3-DiscoLeo-Instruct-8B`, selected after comparative evaluation against alternative base models. Fine-tuning instantiated the domain adaptation strategy from Section 3.5: parameter-efficient training via LoRA [17] on 899 historical assessment reports, conducted entirely on local hardware as data sensitivity precluded cloud-based processing. Statement verification thresholds were empirically calibrated to balance false positives and false negatives in the medical device domain. A five-grade classification system allows for sensitivity adjustments in different use cases. Threshold configurations and the classification-rule matrix are given in the appendix, while the external technical documentation contains the complete training procedures.

Practical utility was assessed through a controlled task-timing comparison following the comparative task analysis methodology of Sauro & Dumas [34]. Following a training session, report creation time was recorded across three standardized medical device damage assessments completed with HAIMEDA and three completed without it. Standardization was enforced by selecting cases of comparable scope, as defined by insurance-company assessment templates. Activities that were unaffected by the system, such as on-site device inspections, telephone inquiries with involved parties, and contact with stakeholders, were excluded from both conditions. The measured scope comprised content writing, document-based research (retrieval of reference reports and internal database queries, where the RIM module provides direct assistance), and final review. Content writing time was reduced by 50%, document-based research activities by 33%, while final review time doubled. These observations are consistent with previous studies showing that generative AI can substantially improve the

productivity of writing-intensive tasks [28], while still requiring human verification and integration effort before professional use [23]. The net 30% reduction in total productive time reflects the difference between the mean of the three assessments completed with HAIMEDA and the mean of the three completed without it.

5.2 Verification Effectiveness

The IIVM underwent systematic evaluation through 100 verification runs per component. Test suites comprised three categories: (i) correct statements extracted from authentic assessment reports serving as negative controls, (ii) deliberately introduced hallucinations created by substituting entity values and inserting semantically plausible but factually incorrect statements, and (iii) versions with critical information systematically removed. Ground truth was established through manual annotation against source documents. Table 1 summarizes the detection rates for each verification component.

Table 1. IIVM verification performance across component types, showing detection rates for hallucinated and missing content based on 100 test runs per component.

Component	Method	Hallucinated	Missing
Preprocessing	Tableaux logic	—	100%
Entities	Regex patterns	83%	90%
Phrases	Semantic rules	76%	90%
Statements	Semantic similarity[†]	72%	65%

[†]Consensus scoring using the five-metric statement-matching scheme (TF-IDF similarity, domain-specific semantic similarity, normalized Euclidean similarity, token overlap, and keyword overlap) at the moderate threshold.

Symbolic preprocessing detected all missing required conditions (100%), establishing a deterministic validation floor for formally specifiable constraints. For structured entities, pattern-based checks detected 83% of hallucinated values and 90% of missing values. Statement-level verification reached 72% hallucination and 65% missing-statement detection at the calibrated "moderate match" operating point that was selected during calibration on domain-representative test cases. A systematic sweep across all five grades was not conducted. Overall, the results confirm complementarity: symbolic checks provide guarantees where constraints are formalizable, whereas neural similarity extends verification to unconstrained text where formal methods and LLM self-verification are less reliable [10,19,21]. Runtime reflects this trade-off: symbolic preprocessing averaged below 20 ms, while hybrid statement verification averaged 16.6 s per chapter due to embedding computation.

The gap between statement-level and entity-level detection reflects structural differences in the verification task. For structured entities, expected values are

discrete and well-formed; symbolic matching is exact and leaves no ambiguity about identity. Statement-level hallucinations, by contrast, take three characteristic forms in this domain: slightly corrupted identifiers embedded in prose (a form spanning the entity–statement boundary), chapter-type phrases absorbed during fine-tuning that are plausible for the section genre but incorrect for the specific case, and contextual elaborations not grounded in any input field. Detecting all three via embedding similarity alone is inherently harder because paraphrase distance between a plausible hallucination and a true statement can be small. Missing-statement failures are structurally the inverse: detection fails when the model omits a detail entirely, so no generated string is available for comparison; the verifier must infer absence from a low-similarity match against an expected reference segment, which is sensitive to how the expected content is represented in the reference corpus. These failure modes are amplified in deployment settings that rely on compact, locally deployable models, such as the Llama3 8B variants used in this case, where hallucination control is of particular importance [27]. This makes the pre- and post-processing guardrails proportionally more important, but also means that moving to larger local models could shift both the base hallucination rate and the verification dynamics.

5.3 Limitations and Baseline Considerations

Results depend on domain-specific rule and regex coverage, and statement-level performance depends on threshold calibration and embedding model choice. External validity is currently limited to a single domain workflow with one expert-centered deployment setting, and the practical utility evaluation is based on a small number of timed trials with a single expert, which limits statistical generalizability while still providing directional evidence in a real professional context. Runtime for neural verification may constrain interactive use in higher-throughput scenarios.

The evaluation was designed to assess integrated system utility in a real professional workflow rather than the marginal contribution of each component in isolation [35]. Accordingly, no dedicated ablation experiment was conducted; such a study remains future work. Still, Table 1 provides indirect evidence for the need for a type-aware architecture. A rule-only baseline would be restricted to structured entities and formally specifiable constraints, leaving free-text statements largely unaddressed. An embedding-only baseline would forfeit deterministic verification for identifiers, dates, and other structured values. LLM self-verification was excluded because it inherits the same distributional biases that produce the hallucinations it is asked to detect [19]. A comparison between the fine-tuned and base model on the same post-processed test suite showed only a 6% difference in hallucination rate and 1.5% in missing-entity rate, suggesting that the main value of fine-tuning lies less in integrity verification than in output conformance to domain vocabulary and professional writing style.

Automation bias, i.e., the tendency of practitioners to accept AI-generated outputs uncritically, was not formally assessed in this evaluation. The system's interactive correction mechanism, which presents discrepancies for explicit expert

review before report finalization, and the mandatory human sign-off prior to professional or legal use are intended design mitigations. Whether these mechanisms are sufficient to prevent over-reliance in sustained professional use is an open empirical question that longitudinal field studies would be needed to address.

6 Conclusion

This paper presented a neuro-symbolic verification architecture for LLM-assisted generation in data-sensitive domains, centered on a principled decomposition of verification tasks: symbolic methods for decidable constraint checking and hybrid neuro-symbolic methods for semantically unconstrained content. The main contribution is a model-agnostic architecture that grounds formal guarantees in symbolic validation rather than any single base model, with actor-based orchestration enabling fault-isolated, locally deployable quality-control pipelines.

The HAIMEDA instantiation demonstrates that this decomposition is practical in a regulated workflow: symbolic components establish deterministic guarantees for input integrity, while hybrid post-processing extends verification to semantic hallucinations under explicit accuracy–latency trade-offs. Beyond detection rates, HAIMEDA demonstrates workflow integration: bidirectional discrepancy diagnosis (missing vs. unsupported claims) is translated into interactive correction feedback, turning verification signals into actionable report revisions.

Current results remain bounded by domain rule coverage, threshold calibration, and single-domain evaluation. Future work should evaluate transfer across legal and financial settings, compare alternative constraint engines and similarity backends via ablations, study calibration strategies that improve semantic recall without unacceptable precision loss, and assess whether larger locally deployable models reduce the verification burden or leave the need for guardrail architectures largely unchanged.

References

1. Aralimatti, R., Shakhadri, S.A.G., Kruthika KR, Angadi, K.B.: Fine-tuning small language models for domain-specific ai: An edge ai perspective (2025), https://arxiv.org/abs/2503.01933
2. Armstrong, J.L.: Making reliable distributed systems in the presence of software errors. In: Ph.D. Thesis, Royal Institute of Technology, Stockholm, Sweden (2003), https://api.semanticscholar.org/CorpusID:28795665
3. Badreddine, S., d'Avila Garcez, A., Serafini, L., Spranger, M.: Logic tensor networks. Artificial Intelligence **303**, 103649 (2022). https://doi.org/https://doi.org/10.1016/j.artint.2021.103649, https://www.sciencedirect.com/science/article/pii/S0004370221002009
4. Besold, T.R., d'Avila Garcez, A., Bader, S., Bowman, H., Domingos, P., Hitzler, P., Kühnberger, K.U., Lamb, L.C., Lima, P.M.V., de Penning, L., Pinkas, G., Poon, H., Zaverucha, G.: Neural-symbolic learning and reasoning: A survey and interpretation. In: Hitzler, P., Sarker, M.K. (eds.) Neuro-Symbolic Artificial Intelligence: The State of the Art, Frontiers in Artificial Intelligence and Applications, vol. 342, pp. 1–51. IOS Press (2021). https://doi.org/10.3233/FAIA210348

5. Capitanelli, A., Mastrogiovanni, F.: A framework for neurosymbolic robot action planning using large language models. Frontiers in Neurorobotics **18** (2024). https://doi.org/10.3389/fnbot.2024.1342786, https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2024.1342786
6. Cavoukian, A.: Privacy by design: The 7 foundational principles. Information and Privacy Commissioner of Ontario, Canada (2009), https://www.ipc.on.ca/wp-content/uploads/resources/7foundationalprinciples.pdf
7. Cesarini, F., Vinoski, S.: Designing for Scalability with Erlang/OTP. O'Reilly Media, Inc., Sebastopol, CA, 1st edn. (may 2016), first Release: 2016-05-11
8. Chudasama, Y., Huang, H., Purohit, D., Vidal, M.E.: Toward interpretable hybrid ai: Integrating knowledge graphs and symbolic reasoning in medicine. IEEE Access **13**, 39489–39509 (2025). https://doi.org/10.1109/ACCESS.2025.3529133
9. Dong, Y., Mu, R., Jin, G., Qi, Y., Hu, J., Zhao, X., Meng, J., Ruan, W., Huang, X.: Building guardrails for large language models (2024), https://arxiv.org/abs/2402.01822
10. Dreyfus, H.L.: What computers still can't do: a critique of artificial reason. MIT Press, Cambridge, MA, USA (1992)
11. European Parliament and Council of the European Union: Regulation (eu) 2017/745 of the european parliament and of the council of 5 april 2017 on medical devices, amending directive 2001/83/ec, regulation (ec) no 178/2002 and regulation (ec) no 1223/2009 and repealing council directives 90/385/eec and 93/42/eec (text with eea relevance) (2017), http://data.europa.eu/eli/reg/2017/745/2017-05-05, consolidated version
12. European Union: Regulation (EU) 2016/679 of the European Parliament and of the Council (General Data Protection Regulation) (2016), http://data.europa.eu/eli/reg/2016/679/oj, oJ L 119, 4.5.2016
13. Gao, L., Dai, Z., Pasupat, P., Chen, A., Chaganty, A.T., Fan, Y., Zhao, V., Lao, N., Lee, H., Juan, D.C., Guu, K.: RARR: Researching and revising what language models say, using language models. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 16477–16508. Association for Computational Linguistics, Toronto, Canada (Jul 2023). https://doi.org/10.18653/v1/2023.acl-long.910, https://aclanthology.org/2023.acl-long.910/
14. Garcez, A.d., Lamb, L.C.: Neurosymbolic AI: The 3rd wave. Artificial Intelligence Review **56**(11), 12387–12406 (2023). https://doi.org/10.1007/s10462-023-10448-w, https://doi.org/10.1007/s10462-023-10448-w
15. German Federal Parliament: Medizinprodukterecht-Durchführungsgesetz (mpdg) (2021), https://www.gesetze-im-internet.de/mpdg/, bGBl. I S. 833, as amended by Article 15 of the Act of 20 December 2023 (BGBl. 2023 I Nr. 408)
16. Hewitt, C., Bishop, P., Steiger, R.: A universal modular actor formalism for artificial intelligence. In: Proceedings of the 3rd International Joint Conference on Artificial Intelligence. p. 235–245. IJCAI'73, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1973)
17. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models (2021)
18. Huang, B., Chen, C., Xu, X., Payani, A., Shu, K.: Can knowledge editing really correct hallucinations? (2025), https://arxiv.org/abs/2410.16251
19. Huang, J., Chen, X., Mishra, S., Zheng, H.S., Yu, A.W., Song, X., Zhou, D.: Large language models cannot self-correct reasoning yet (2024), https://arxiv.org/abs/2310.01798

20. Hughes, J.: Why functional programming matters. The Computer Journal **32**(2), 98–107 (Jan 1989). https://doi.org/10.1093/comjnl/32.2.98
21. Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y.J., Madotto, A., Fung, P.: Survey of hallucination in natural language generation. ACM Computing Surveys **55**(12) (Mar 2023). https://doi.org/10.1145/3571730, https://doi.org/10.1145/3571730
22. Kleppmann, M., Wiggins, A., van Hardenberg, P., McGranaghan, M.: Local-first software: you own your data, in spite of the cloud. In: Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software. pp. 154–178. Onward! 2019, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3359591.3359737, https://doi.org/10.1145/3359591.3359737
23. Lee, H.P.H., Sarkar, A., Tankelevitch, L., Drosos, I., Rintel, S., Banks, R., Wilson, N.: The impact of generative ai on critical thinking: Self-reported reductions in cognitive effort and confidence effects from a survey of knowledge workers. In: Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems. CHI '25, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3706598.3713778, https://doi.org/10.1145/3706598.3713778
24. Li, Z., Huang, J., Naik, M.: Scallop: A language for neurosymbolic programming (2023)
25. Manhaeve, R., Dumančić, S., Kimmig, A., Demeester, T., De Raedt, L.: Neural probabilistic logic programming in deepproblog. Artificial Intelligence **298**, 103504 (2021). https://doi.org/https://doi.org/10.1016/j.artint.2021.103504, https://www.sciencedirect.com/science/article/pii/S0004370221000552
26. Marcus, G., Davis, E.: Rebooting AI: Building Artificial Intelligence We Can Trust. Pantheon Books, USA (2019)
27. Nadeau, D., Kroutikov, M., McNeil, K., Baribeau, S.: Benchmarking llama2, mistral, gemma and gpt for factuality, toxicity, bias and propensity for hallucinations (2024), https://arxiv.org/abs/2404.09785
28. Noy, S., Zhang, W.: Experimental evidence on the productivity effects of generative artificial intelligence. Science **381**(6654), 187–192 (2023). https://doi.org/10.1126/science.adh2586, https://www.science.org/doi/abs/10.1126/science.adh2586
29. Parnas, D.L.: On the criteria to be used in decomposing systems into modules. Commun. ACM **15**(12), 1053–1058 (Dec 1972). https://doi.org/10.1145/361598.361623, https://doi.org/10.1145/361598.361623
30. Rebedea, T., Dinu, R., Sreedhar, M.N., Parisien, C., Cohen, J.: NeMo guardrails: A toolkit for controllable and safe LLM applications with programmable rails. In: Feng, Y., Lefever, E. (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 431–445. Association for Computational Linguistics, Singapore (Dec 2023). https://doi.org/10.18653/v1/2023.emnlp-demo.40, https://aclanthology.org/2023.emnlp-demo.40/
31. Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 3982–3992. Association for Computational Linguistics, Hong Kong, China (Nov 2019). https://doi.org/10.18653/v1/D19-1410, https://aclanthology.org/D19-1410/
32. de Rijke, M.: Handbook of tableau methods, Marcello D'Agostino, Dov M. Gabbay, Reiner Hähnle, and Joachim Posegga, eds. Journal of Logic, Language and Information **10**(4), 518–523 (Dec 2001). https://doi.org/10.1023/A:1017520120752, https://doi.org/10.1023/A:1017520120752

33. Sarker, M.K., Zhou, L., Eberhart, A., Hitzler, P.: Neuro-symbolic artificial intelligence: Current trends. AI Communications **34**(3), 197–209 (2021). https://doi.org/10.3233/AIC-210084
34. Sauro, J., Dumas, J.S.: Comparison of three one-question, post-task usability questionnaires. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 1599–1608. CHI '09, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1518701.1518946, https://doi.org/10.1145/1518701.1518946
35. Sein, M.K., Henfridsson, O., Purao, S., Rossi, M., Lindgren, R.: Action design research. MIS Quarterly **35**(1), 37–56 (2011), http://www.jstor.org/stable/23043488
36. Smullyan, R.M.: First-Order Logic. Dover, 1 edn. (1995)
37. Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., Zhou, D.: Self-consistency improves chain of thought reasoning in language models (2023), https://arxiv.org/abs/2203.11171
38. Willard, B.T., Louf, R.: Efficient guided generation for large language models (2023), https://arxiv.org/abs/2307.09702

A Appendix

This appendix provides the technical details critical to reproducibility that support the main paper, including classification thresholds, coverage-scoring weights, and inference parameters. Full fine-tuning experiments, analyses, dependency manifests, throughput profiling, and usability testing are detailed in an external technical documentation available online.

Section A.1 presents the configuration used during evaluation, while Section A.2 consolidates resource links and data-availability constraints, including the external technical documentation repository.

A.1 System Configuration and Evaluation

This section documents the operational parameters used during the HAIMEDA evaluation to ensure reproducibility of the reported results. Table 2 defines thresholds for the five-grade statement matching system, Table 3 lists entity-type weights used to compute weighted coverage scores, and Table 4 specifies the model configurations used for inference.

Verification Classification System The rule-based verification scheme uses five match grades. These thresholds were determined through testing on assessment reports in order to balance precision and recall in the medical device domain. The "moderate match" threshold (combined score $> 45\%$, confidence $> 40\%$) was chosen to be the default sensitivity level for production use.

Distance-based metrics are converted to similarity percentages using bounded normalization. Euclidean similarity is computed as

$$s_{\text{euclidean}} = \left(1 - \min\left(\frac{d_{\text{euclidean}}}{d_{\max}}, 1\right)\right) \times 100,$$

Table 2. The verification classification matrix defines grade thresholds and conditions for statement matching. Unless explicitly combined with conjunction (AND), conditions use logical disjunction (OR).

Grade	Score Min	Conf. Min	Key Conditions (OR)
Exact	90	90	(combined $\geq 95 \wedge$ overlap ≥ 90) (domain $\geq 95 \wedge$ tfidf ≥ 45) (euclidean $\geq 30 \wedge$ combined ≥ 95)
Strong	75	75	tfidf ≥ 35 domain ≥ 80 (keyword $\geq 70 \wedge$ combined ≥ 80) euclidean ≥ 45
Moderate	45	40	tfidf ≥ 20 domain ≥ 50 keyword ≥ 20 (combined $\geq 50 \wedge$ confidence ≥ 50)
Weak	25	0	No additional conditions
No match	0	0	Default classification

Note: Abbreviations and metrics used in the table.

- **Score** = combined similarity score.
- **Conf.** = confidence threshold.
- **Metrics**: TF-IDF similarity (tfidf), domain-specific semantic similarity (domain), normalized Euclidean similarity (euclidean), token overlap percentage (overlap), keyword overlap percentage (keyword); *combined* denotes the weighted aggregate score and *confidence* denotes inter-metric agreement.

where $d_{\text{euclidean}} = \|\mathbf{e}_1 - \mathbf{e}_2\|_2$ is the L2 distance between embedding vectors and $d_{\max} = 2.0$ represents a conservative upper bound for normalized embeddings. Manhattan distance uses analogous normalization with $d_{\max} = 10.0$.

Statement verification uses a parallel worker pool, in which each worker processes one input-output statement pair to determine semantic equivalence. The architecture implements adaptive scaling based on available GPU memory, allocating approximately 200 MB of VRAM per worker for the `paraphrase-multilingual-MiniLM-L12-v2` embedding model. On the evaluation hardware with 24 GB VRAM, this enables up to 100 concurrent verification workers, with automatic CPU fallback when GPU resources are limited or unavailable. Statement uniqueness detection eliminates redundant embedding computations when identical statements appear across multiple comparisons.

Coverage Scoring Weights The verification system computes coverage metrics through bidirectional entity comparison. Input coverage identifies entities from the source material that are successfully represented in the generated output. Output coverage recognizes generated entities that lack support from the source material. Coverage percentages and weighted content scores provide a quantitative assessment of verification quality.

Entity-type weights reflect the confidence in the verification based on the determinism of the underlying comparison method. Table 3 specifies the weights used in HAIMEDA's coverage scoring.

Table 3. Coverage weights by entity type based on verification confidence and comparison-method determinism.

Entity Type	Weight	Verification Method
Dates	0.5	Symbolic (pattern matching)
Identifiers	0.5	Symbolic (pattern matching)
Numeric Values	0.4	Symbolic (pattern matching)
Phrases	0.2	Hybrid (semantic rules)
Statements	0.2	Neural (embedding similarity)

Structured entities, such as dates, identifiers, and numeric values, receive the highest weights due to their symbolic verifiability. Pattern-based comparison provides deterministic verification outcomes. Semantic content, such as phrases and statements, receives lower weights because verification relies on neural similarity measures, which have inherent uncertainty. The weighted content score, S_w, is computed as

$$S_w = \frac{\sum_{t \in T} w_t \cdot c_t}{\sum_{t \in T} w_t \cdot n_t},$$

where T is the set of entity types, w_t is the weight for type t, c_t is the count of verified entities of type t, and n_t is the total count of entities of type t.

Inference Parameters Table 4 specifies model configurations for each task type. Temperature and sampling parameters were tuned to balance output quality against factual accuracy: lower temperatures for revision tasks requiring precision, higher temperatures for optimization tasks permitting stylistic flexibility. All models use either 8-bit or 16-bit quantization for deployment on consumer hardware.

Table 4. Model configuration and inference parameters for various tasks.

Task	Model	τ	top_p	top_k	max_tokens	repeat_penalty
RAG / Q&A	llama3_german_v1-b	0.2	0.5	50	6144	1.1
Text Optimization	llama3_german_v3	0.4	0.6	60	4096	1.1
Text Revision	llama3_german_v3	0.2	0.6	45	4096	1.3
Text Summarization	llama3_german_instruct	0.1	0.5	40	2048	1.2

Note: τ = temperature. RAG context is limited to 12,000 characters with nomic-embed-text embeddings.

A.2 Resources and Data Availability

Technical Documentation Full implementation details, including complete fine-tuning experiments, dependency manifests, throughput profiling, and usability instrumentation, are provided in the external technical documentation.[3]

Code Repositories The HAIMEDA application source code[4] and a companion toolkit[5] for data preparation, fine-tuning, and model integration are publicly available. Configuration files and example data have been anonymized; domain-specific content containing personally identifiable information has been removed or replaced with synthetic examples.

Supplementary Practitioner Material An external practitioner guide for broader hybrid-AI development in data-sensitive domains is available online.[6] As supplementary material, it distills practical lessons from the ADR-based development of HAIMEDA and synthesizes relevant literature into design principles, lifecycle guidance, and implementation checklists.

Data Availability The training corpus consists of confidential medical device assessment reports containing sensitive information protected under GDPR [12] and the Medical Devices Regulation [11]. This includes patient data, device serial numbers, stakeholder contact details, and proprietary manufacturer information. Neither the raw training data nor the fine-tuned model weights can be publicly released, as the models may retain sensitive information from the training corpus. The pretrained base model (`Llama3-DiscoLeo-Instruct-8B`) is publicly available.[7]

[3] https://github.com/penthooose/HAIMEDA_technical_documentation
[4] https://github.com/penthooose/HAIMEDA
[5] https://github.com/penthooose/ai_dev_toolkit
[6] https://github.com/penthooose/hybrid_ai_dev_guide
[7] https://huggingface.co/DiscoResearch/Llama3-DiscoLeo-Instruct-8B-v0.1