RAGVUE: A Diagnostic View for Explainable and Automated Evaluation of Retrieval-Augmented Generation

Keerthana Murugaraj[1], Salima Lamsiyah[1], Martin Theobald[1]
[1]University of Luxembourg, Department of Computer Science (DCS),
Faculty of Science, Technology and Medicine (FSTM), Esch-sur-Alzette, Luxembourg
Correspondence: keerthana.murugaraj@uni.lu

Abstract

Evaluating Retrieval-Augmented Generation (RAG) systems remains a challenging task: existing metrics often collapse heterogeneous behaviors into single scores and provide little insight into whether errors arise from retrieval, reasoning, or grounding. In this paper, we introduce RAGVUE, a diagnostic and explainable framework for automated, reference-free evaluation of RAG pipelines. RAGVUE decomposes RAG behavior into retrieval quality, answer relevance and completeness, strict claim-level faithfulness, and judge calibration. Each metric includes a structured explanation, making the evaluation process transparent. Our framework supports both manual metric selection and fully automated agentic evaluation. It also provides a Python API, CLI, and a local Streamlit interface for interactive usage. In comparative experiments, RAGVUE surfaces fine-grained failures that existing tools such as RAGAS often overlook. Our demonstration showcases the full RAGVUE workflow and illustrates how it can be integrated into research pipelines and practical RAG development. The source code as well as detailed instructions on its usage are publicly available on Github [1].

1 Introduction

Retrieval-Augmented Generation (RAG) combines a pretrained (parametric) language model with an external retriever that supplies relevant documents at inference time (Lewis et al., 2020; Guu et al., 2020). By conditioning generation on retrieved passages, RAG systems effectively tackle knowledge-intensive tasks while making their evidence explicit and easier to maintain than finetuning internal model weights (Lewis et al., 2020; Izacard et al., 2023). This paradigm has rapidly become a default solution for building search assistants, analytical tools, customer-support bots, and domain-specific copilots across high-stakes settings such as finance, healthcare, and law (Song et al., 2024; Rosenthal et al., 2025). Recent benchmarks further stress-test RAG in multi-hop and multi-turn scenarios (e.g., StrategyQA (Geva et al., 2021), mtRAG (Katsis et al., 2025), CLAPnq (Rosenthal et al., 2025)), underscoring the need for robust and fine-grained evaluation of the full RAG pipeline.

Evaluating RAG is harder than evaluating a standalone language model because errors can arise from retrieval (irrelevant or missing evidence), generation (off-topic, incomplete, or incoherent answers), or grounding (unsupported or contradictory claims despite retrieved context) (Es et al., 2024; Saad-Falcon et al., 2024; Ru et al., 2024). Recent surveys argue that global "end-to-end" scores obscure these components and advocate decomposing the evaluation into retrieval quality, answer quality, and evidence support (Yu et al., 2024; Gan et al., 2025). They also emphasize a crucial distinction between *faithfulness* to retrieved evidence and *factual correctness* with respect to world knowledge: a response may be factually true but unsupported by its citations, or fully grounded in outdated or erroneous evidence (Min et al., 2023; Sorodoc et al., 2025). Temporal drift (Ouyang et al., 2025), unanswerability (Peng et al., 2025), and privacy or policy violations in retrieved content (Zeng et al., 2025; Song et al., 2024) introduce additional evaluation axes that simple accuracy-style metrics cannot capture.

Human annotation and gold references are expensive and brittle under domain shift (Saad-Falcon et al., 2024; Rosenthal et al., 2025), motivating reference-free *LLM-as-a-judge* methods that are widely used in NLG evaluation (Wang et al., 2023; Kocmi and Federmann, 2023; Zheng et al., 2023). Despite progress (e.g., G-Eval (Liu et al., 2023), AutoCalibrate (Liu et al., 2024), SelfCheckGPT (Manakul et al., 2023)), LLM judges remain prompt-sensitive, unstable, and prone to self-preference bias (Panickssery et al., 2024; Schroeder

[1] https://github.com/KeerthanaMurugaraj/RAGVue

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics
Volume 3: System Demonstrations, pages 512–526

and Wood-Doughty, 2024; Liu et al., 2025). Existing RAG-focused evaluators, including RAGAS (Es et al., 2024), ARES (Saad-Falcon et al., 2024), RAGChecker (Ru et al., 2024), and RAG-Zeval (Li et al., 2025) have expanded coverage. However, two core gaps persist: metrics often collapse heterogeneous behaviors into non-diagnostic scalar scores, and grounding checks remain permissive, missing fine-grained factual errors (Es et al., 2024; Niu et al., 2024; Song et al., 2024).

To address these limitations, we introduce RAGVUE, a reference-free, explainable evaluation framework that offers *diagnostic results* rather than purely numerical assessments. RAGVUE decomposes RAG performance into retrieval quality, answer quality, and factual grounding (Yu et al., 2024; Gan et al., 2025), enforcing *strict faithfulness* by crediting only claim-level evidence explicitly supported in the retrieved context. This yields a more conservative alternative to semantic-inference metrics (Es et al., 2024; Min et al., 2023; Niu et al., 2024; Zhu et al., 2025). RAGVUE additionally introduces a *judge-calibration* score quantifying agreement across LLM evaluators, making stability issues in LLM-as-a-judge setups explicit (Liu et al., 2025; Schroeder and Wood-Doughty, 2024; Panickssery et al., 2024). The framework supports both *manual* metric selection and an *agentic* mode, in which an internal orchestrator automatically chooses and aggregates metrics. Moreover, we provide a Python API, command-line interface (CLI), and a Streamlit-based user interface (UI) for a seamless integration into research workflows. Finally, on a multihop StrategyQA-derived benchmark (Geva et al., 2021), RAGVUE reveals fine-grained failures that approaches based on scalar metrics, such as RAGAS (Es et al., 2024), fail to capture.

2 Related Work

RAG & Evaluation Challenges. Retrieval-Augmented Generation (RAG) integrates external evidence into LLMs to reduce hallucinations and improve grounding (Lewis et al., 2020; Kocmi and Federmann, 2023; Wang et al., 2023). Early models such as REALM (Guu et al., 2020) and RAG (Lewis et al., 2020) showed strong gains on knowledge-intensive tasks, followed by advances in retrieval and generation (e.g., late-interaction retrievers (Khattab et al., 2021) and few-shot RAG tuning (Izacard et al., 2023)). However, the pipeline-based nature of RAG introduces unique evaluation challenges. Performance must be assessed across components, including retriever relevance and evidence coverage, generator quality, and grounding faithfulness (Saad-Falcon et al., 2024). Recent surveys argue that end-to-end scores obscure these dimensions and call for separate evaluation of retrieval quality and grounding fidelity (Yu et al., 2024; Gan et al., 2025). Despite having access to documents, RAG models still tend to hallucinate or rely on outdated evidence, motivating benchmarks for temporal drift and attribution (Ouyang et al., 2025). Moreover, faithfulness and factual correctness may diverge: a response can be true but unsupported, or well-grounded yet incorrect (Khattab et al., 2021). We follow this line by separately evaluating retrieval, answer quality, and grounding with strict evidence criteria.

RAG Evaluation Frameworks & Benchmarks. Recent work has introduced automatic evaluators for RAG. RAGAS (Es et al., 2024) provides reference-free metrics for context relevance, answer coherence, and coarse groundedness via static prompt-based queries. ARES (Saad-Falcon et al., 2024) increases robustness by fine-tuning smaller LMs on human labels, offering explicit relevance and faithfulness scores with confidence estimates. RAGChecker (Ru et al., 2024) adds diagnostic checks for passage usage and claim-level grounding. Benchmarks such as RAGTruth (Niu et al., 2024) and MEMERAG (Cruz Blandón et al., 2025) target hallucinations and multilingual settings, while HoH (Ouyang et al., 2025) and Unanswerability-Eval (Peng et al., 2025) test temporal drift and unanswerable queries. Furthermore, LLM-as-a-judge approaches are widely adopted (Wang et al., 2023; Zheng et al., 2023), including G-Eval (Liu et al., 2023), AutoCalibrate (Liu et al., 2024), and FactScore (Min et al., 2023), but remain prompt-sensitive and biased toward model families (Panickssery et al., 2024; Liu et al., 2025). More reliable methods such as JudgeLM (Liu et al., 2025) and RAG-Zeval (Li et al., 2025) seek stability through consensus and reasoning-based ranking. Building on these insights, RAGVUE uses reference-free LLM judges while addressing key limitations by decomposing scores (retrieval vs. coverage; answer relevance vs. completeness), enforcing strict claim-level faithfulness, and providing fine-grained explanations and stability checks. It also includes an *agentic*

evaluation mode that automatically selects and orchestrates metrics, producing structured summaries ready for debugging and comparison.

3 RAGVUE Framework Overview

This section introduces our RAGVUE evaluation framework. We first outline its core metrics, then describe its two operational modes, and conclude with its programmatic and interactive UIs.

3.1 RAGVUE Metrics

We describe seven RAGVUE metrics across three dimensions: (1) retrieval, (2) answer quality, and (3) grounding and stability, with a summary provided in Appendix A (Table 1).

3.1.1 Retrieval Relevance

This metric measures whether the retrieved contexts (C) are actually useful for answering the question (Q). For each context chunk, an LLM judge assigns a relevance score r_i in $[0, 1]$ using a predefined range[2]. A chunk (c_i) is counted as relevant if its score exceeds this threshold[3], and the final score is computed as:

$$\text{RetrievalRelevance} = \frac{\#\{\, c_i \geq \tau \,\}}{N} \qquad (1)$$

where N is the number of retrieved chunks. This precision-style formulation is simple, cost-efficient, and provides actionable diagnostic insight into retrieval quality. Importantly, it evaluates the usefulness of retrieved documents directly from the question alone, without requiring reference answers.

3.1.2 Retrieval Coverage

This metric measures whether the retrieved contexts (C) collectively contain the evidence needed to answer the question (Q), without requiring reference contexts. We first derive a small set of atomic aspects from the question alone and reuse the same aspects across metrics for consistency. Let $\mathcal{A}$ denote this set of aspects and let $\mathcal{R}_{\text{cov}} \subseteq \mathcal{A}$ be the subset of aspects supported by at least one retrieved document. The corresponding score is:

$$\text{RetrievalCoverage} = \frac{|\mathcal{R}_{\text{cov}}|}{|\mathcal{A}|} \qquad (2)$$

This recall-style metric indicates whether the retriever has surfaced enough evidence to cover all parts of the question.

3.1.3 Clarity

This metric evaluates the linguistic quality of the generated answer (A), assessing grammar, fluency, logical flow, conciseness, and overall readability. A single LLM call returns a score in $[0, 1]$ along with a brief explanation and suggested improvements. Short answers are also checked for naturalness and readability. Overall, this metric provides a compact indication of how clearly the answer is written.

3.1.4 Answer Relevance

Answer Relevance measures how well the generated answer (A) aligns with the user's question (Q) intent. The metric considers only the question and the generated answer, and assigns a score in $[0, 1]$ based on topical focus and whether the answer meaningfully addresses what the question is asking. It ignores factual correctness and stylistic quality. High scores thus indicate that the answer stays on-topic and captures the main intent, while lower scores reflect partial, generic, or off-topic content. The judge additionally returns short lists of missing or off-topic parts to provide interpretable signals about alignment.

3.1.5 Answer Completeness

Answer Completeness measures how well the answer covers the different aspects implied by the question. Using the same aspect set $\mathcal{A}$ (from 3.1.2), the metric checks each aspect against the answer and identifies the subset $\mathcal{A}_{\text{cov}} \subseteq \mathcal{A}$, i.e., the aspects that the answer explicitly addresses (optionally with short supporting snippets). The final score is computed as:

$$\text{AnswerCompleteness} = \frac{|\mathcal{A}_{\text{cov}}|}{|\mathcal{A}|} \qquad (3)$$

This reference-free metric captures how thoroughly the answer resolves the information needs expressed by the question.

3.1.6 Strict Faithfulness

To assess factual grounding, we introduce a *single-pass* faithfulness metric that checks whether the retrieved context supports each factual claim in the generated answer. Using a single LLM call, the evaluator (i) decomposes the answer into minimal atomic claims and (ii) labels each claim as *supported*, *partially hallucinated*, or *fully hallucinated*.

[2]Default ranges: 1.0–0.9 for direct answer-containing evidence; 0.8–0.7 for highly useful content; 0.6–0.3 for weakly related background; 0.2–0.0 for irrelevant text.

[3]The threshold is set to $\tau = 0.7$ to include only evidence judged as highly useful.

Unlike multi-stage pipelines that require multiple LLM calls and subsequent heuristic aggregation, RAGVUE encodes the verification logic directly within one prompt.

The strictness is intentionally applied to *high-risk factual anchors*, i.e., *key entities* (such as people, locations, organizations) and *temporal expressions* (such as years and dates). We enforce exact agreement on these anchors to capture frequent RAG failure modes such as entity substitution and incorrect temporal details. Claims with missing, unsupported, or contradictory anchors are treated as hallucinated. This design favors conservative factual checking over stylistic flexibility, i.e., paraphrases are acceptable as long as entity and temporal anchors are consistent. The final score is computed as:

$$\text{StrictFaithfulness} = \frac{|\mathcal{C}_{\text{supported}}|}{|\mathcal{C}_{\text{supported}}| + |\mathcal{C}_{\text{hallucinated}}|} \tag{4}$$

where $\mathcal{C}_{\text{supported}}$ denotes claims fully grounded in the retrieved context and $\mathcal{C}_{\text{hallucinated}}$ denotes claims marked as partially or fully hallucinated. The resulting score is transparent and directly traceable to claim-level decisions, allowing users to quickly identify which parts of an answer are evidence-backed.

3.1.7 Generic Calibration

LLM-based evaluators are sensitive to sampling noise, decoding temperature, and choice of the model, and they may also reflect systematic biases inherited from the judge models. In practice, many RAG evaluation pipelines implicitly assume that a single judge configuration is both stable and trustworthy. RAGVUE does not attempt to remove or solve judge bias. Instead, we make judge reliability observable by introducing a *generic calibration metric* that quantifies agreement across multiple judge configurations and can be applied to any RAGVUE metric. For each evaluation case, we run the same underlying metric under several $(\text{model}, \text{temperature})$ configurations and obtain a set of scores $s_1, \ldots, s_k$. We define calibration agreement as:

$$\text{Calibration} = 1 - (\max_i s_i - \min_i s_i) \tag{5}$$

which assigns high values when judges behave consistently and low values when their outputs diverge. Importantly, high calibration indicates *stability* across judge settings, but it does not guarantee correctness, fairness, or absence of systematic bias. Similarly, a low calibration score indicate that the evaluation outcome is brittle and should be treated with caution.

Beyond reporting an aggregate agreement score, RAGVUE surfaces per-judge outputs and explanations, allowing users to inspect which configurations disagree and why. This design supports practical safeguards such as (i) preferring conclusions that are consistent across multiple judges, (ii) marking low-calibration cases for manual review, and (iii) optionally expanding the judge set to include more diverse models to reduce dependence on any single judge's output. Overall, calibration in RAGVUE increases transparency by indicating whether an evaluation result is consistent across different judge models and temperature settings, or highly sensitive to those choices, without claiming to eliminate inherent judge bias.

3.2 Operational Modes & Availability

Our evaluation framework supports two complementary operational modes and user interfaces, enabling flexible integration into research workflows and production pipelines.

3.2.1 Operational Modes

RAGVUE provides two modes for users. In **manual mode**, users control which metrics are executed and how results are aggregated, offering full transparency and fine-grained control. In **agentic mode**, an internal orchestration agent fully automates evaluation. The agent selects appropriate retrieval and answer-level metrics based on the presence of context, the availability of an answer, and the user query. It then executes these metrics in a single pass and synthesizes high-level scores, including an overall retrieval score (harmonic mean of relevance and coverage) and an answer-level composite score (weighted blend of strict faithfulness, relevance, completeness, and clarity).

3.2.2 Availability

RAGVUE is released under the Apache License 2.0 and can be used through multiple access modes depending on the user's preference and technical requirements.

Python API. RAGVUE can also be used directly as a Python library (Fig. 1). Users import the evaluator, load a JSONL dataset, and run all metrics with a single function call, making this mode ideal for

integration into notebooks, scripts, and automated pipelines.

```
from ragvue import evaluate, load_metrics
items = [
    {"question": "...", "answer": "...",
    "context": [...]}]
metrics = load_metrics().keys()
report = evaluate(items, metrics=list(metrics))
print(report)
```

Figure 1: RAGVUE Python API usage example.

Python/Command-Line Interface (CLI). RAGVUE provides a simple command-line interface (ragvue-cli) for terminal-based workflows, as shown in Figure 2. A lightweight Python runner (ragvue-py) is also available, as illustrated in Figure 3. Both interfaces support listing available metrics, running manual evaluations, and executing the agentic mode, enabling fast and scriptable evaluation without writing additional code.

```
# Help
ragvue-cli --help
# List all available metrics
ragvue-cli list-metrics
# Manual evaluation (choose metrics explicitly)
ragvue-cli eval
  --inputs <your_data.josnl>
  --metrics <metrics>
  --out-base report_manual
  --format "json,md,csv"
# Agentic evaluation (auto-select metrics)
ragvue-cli agentic
  --inputs <your_data.jsonl>
  --out-base report_agentic
   --format "json,md,csv"
```

Figure 2: Example usage of the RAGVUE command-line interface (`ragvue-cli`).

```
# Help
ragvue-py --help
# Manual Mode Usage
ragvue-py  --input <your_data> --metrics <metrics>
--out-base report_manual --skip-agentic
# Agentic Mode Usage
ragvue-py  --input <your_data> --metrics <metrics>
--agentic-out report_agentic --skip-manual
```

Figure 3: Example usage of the RAGVUEchat Python command-line runner (`ragvue-py`).

Local Streamlit Application. For no-code, interactive usage, we provide a Streamlit-based UI that exposes the same capabilities through an interactive browser interface. The application is run locally: users clone the repository and start the interface with a standard command such as `streamlit run streamlit_app.py`. Within this local UI, users can upload JSONL files, select operational modes, set/paste API keys for the current session, and generate formatted reports without writing code. This interface is targeted at practitioners who prefer a point-and-click workflow while keeping all data and keys on their own machine. The images of our UI are shown in Appendix E.

4 Experiments & Discussion

4.1 Dataset

We construct our evaluation dataset based on the multihop StrategyQA (Geva et al., 2021) benchmark. Each item contains a question, the reference yes/no label, the supporting facts, the decomposition steps, and the Wikipedia evidence titles. The supporting facts are cleaned and used as independent context snippets. In the next stage, we generate five answer variants for each question, such as ideal, partial, unclear, off-topic, and hallucinated. These variants capture common RAG failure modes by altering the correctness of the label, the completeness of the explanation, and the relevance or confidence of the response. Each answer is stored with its associated metadata (question ID, reference label, contexts, supporting facts, decomposition, and evidence titles), producing exactly five synthetic examples per question. For this study, we created 100 synthetic (Q, C, A) triplets from StrategyQA. The final dataset is exported in two formats: a RAGAS-compatible JSON for metric-based evaluation and a RAGVUE-ready JSONL for interactive inspection.

4.2 RAGVUE vs. RAGAS Performance

Computational Time. We first measured latency on the 100 queries described in Section 4.1. RAGAS averaged 18.26 seconds per query, while RAGVUE averaged 18.87 seconds. This represents a marginal 3.4% increase in per-item latency, which is negligible. As shown in the boxplot (Appendix B), both systems exhibit nearly identical latency distributions, including similar medians, inter-quartile ranges, and outliers. Importantly, RAGVUE provides richer diagnostics, and this enhanced granularity makes it more actionable for system debugging and improvements, rendering the small computational overhead a worthwhile

trade-off.

Quantitative Analysis. We next summarize the behavior of both evaluators using descriptive statistics over the 100 queries (Appendix C). RAGAS' faithfulness has a mean of 0.52, while answer relevance and response groundedness average at 0.24 and 0.39, respectively, indicating that RAGAS often judges answers as only weakly relevant or weakly grounded. In contrast, RAGVUE reports low average answer completeness (0.12) and moderate answer relevance (0.37), alongside consistently high clarity scores (0.70). Its retrieval metrics center around 0.50 for coverage and 0.42 for relevance, while strict-faithfulness has a mean of 0.40, reflecting a wide range of partially supported and unsupported answers.

A correlation analysis (see Appendix C) shows that the two evaluators align on broad, generation-focused behavior but diverge sharply on retrieval-focused metrics. RAGAS' faithfulness, answer relevancy, and response groundedness correlate strongly with RAGVUE's strict faithfulness and answer relevance, indicating comparable sensitivity to high-level answer correctness. However, RAGAS' retrieval-related metrics, context relevance and response groundedness, correlate only weakly or inconsistently with RAGVUE's retrieval coverage and retrieval relevance. This reveals that RAGAS often conflates insufficient retrieval with unsupported reasoning, whereas RAGVUE explicitly separates retrieval performance from generation performance.

Qualitative Analysis. Our qualitative inspection reveals several systematic failures that RAGAS does not diagnose, and some examples are presented in Appendix D. We find that RAGAS often gives scores that look reasonable but do not explain why an answer fails. When the model gives vague, unsupported, or partially relevant answers, RAGAS may still assign high or mid-range faithfulness scores because it only checks for direct contradictions and does not account for missing multi-hop reasoning, unanswered parts of the question, or unsupported conclusions. RAGVUE, on the other hand, clearly shows what went wrong: it marks claims as unsupported when the evidence does not back them, highlights when the answer ignores key aspects of the question, and indicates whether retrieval fully or only partially matched what was needed. As a result, RAGVUE makes it easy to see whether the error comes from retrieval, grounding, or the model's reasoning, which is not possible with RAGAS. Overall, the qualitative analysis shows that RAGVUE provides clearer and more actionable feedback for diagnosing RAG system failures.

Discussion. Our quantitative and qualitative analyses directly reflect the structural limitations of RAGAS. Context Relevance collapses all chunks into a single aggregated score and cannot show which passages are missing or irrelevant. RAGVUE provides per-chunk relevance without requiring reference answers, clearly exposing retrieval strengths and failures. Likewise, RAGAS' Context Recall requires a reference answer and multi-step alignment, while RAGVUE's Retrieval Coverage operates directly on the question and retrieved documents, making it usable even when references are unavailable. On the generation side, RAGAS' Answer Relevancy relies on embedding-based synthetic question generation, capturing only coarse semantic overlap. RAGVUE's Answer Relevance is intent-aware and identifies missing or off-topic elements, yielding actionable diagnostics about why an answer may be incomplete. Finally, RAGAS' Response Groundedness assigns coarse labels (0/1/2) based on inferred support in the context, but cannot reveal which question aspects were addressed or missed. RAGVUE's Answer Completeness evaluates coverage directly from the question's aspect structure, producing a fine-grained completeness signal. Strict Faithfulness exhibits the clearest contrast: RAGAS uses a two-step pipeline to extract statements from the answer and verify each with a semantic-inference prompt that tolerates semantic drift, whereas RAGVUE decomposes the answer into atomic claims and enforces exact evidence matching for key entities and temporal expressions. This yields a stricter, more deterministic assessment of factual support. Overall, these results show that, while RAGAS provides high-level semantic judgments, RAGVUE delivers finer-grained, retrieval-aware, and more diagnostically meaningful evaluation signals.

5 RAGVUE in the Loop

RAGVUE is designed to support iterative RAG development. Developers can modify individual pipeline components (e.g., retrieval settings, chunking, reranking, prompting, or abstention rules) and re-run RAGVUE on a fixed development set to compare diagnostic reports across versions. This

workflow is supported through multiple access interfaces: evaluations can be executed programmatically via the Python API, scripted through the CLI, or inspected interactively using the Streamlit UI.

To make iterations actionable, RAGVUE exposes decomposed evaluation signals with structured explanations. It separates retrieval behavior (*retrieval relevance* and *retrieval coverage*) from answer quality (*answer relevance* and *answer completeness*) and grounding (*strict claim-level faithfulness*). This decomposition is intended to help localize failures to retrieval, generation, or grounding rather than relying on a single aggregate score. For example, retrieval coverage reflects whether the retrieved set contains evidence for the required aspects of a query, retrieval relevance provides per-chunk usefulness signals, and strict faithfulness flags claims that are unsupported under evidence-matching constraints.

6 Conclusion

RAGVUE provides an automated, diagnostic, and fully reference-free evaluation framework tailored for explainable assessment of RAG systems. It separates retrieval and generation-level metrics, delivers structured explanations rather than opaque scalar scores, and exposes the underlying causes of model failures. The agentic evaluation mode makes the framework immediately usable with minimal setup, automatically selecting appropriate metrics and producing structured reports that highlight where and why a pipeline breaks. By combining fine-grained metrics with transparent reasoning traces and cross-model calibration for reliability, RAGVUE reveals whether an error stems from retrieval drift, missing evidence, unsupported reasoning, or incomplete answers, problems that traditional metrics such as RAGAS often conflate. Overall, RAGVUE functions not only as an evaluator but as a practical debugging tool for real-world RAG development, which helps users identify weaknesses and iteratively improve their RAG systems.

7 Limitations & Future Work

RAGVUE represents our first step toward a transparent and diagnostic evaluation framework for RAG. The current version delivers seven core metrics, two operational modes, and a no-code user interface, but there is still significant room for growth. As RAGVUE relies on LLM-based evaluation, careful selection of the underlying judge models is recommended to ensure stable and consistent scoring. We plan to extend RAGVUE with additional metrics, perform retrieval and grounding analysis on complex queries, and provide broader support for different LLM models. The agentic mode will become more adaptive, assisting users by detecting errors and automatically selecting appropriate metrics.

At present, RAGVUE provides diagnostic outputs and structured explanations, but it does not automatically modify the underlying RAG system. A natural extension is tighter integration into semi-automated development loops, where evaluation results guide interventions such as retriever configuration updates, reranker/threshold selection, or abstention/refusal policies when evidence is insufficient. In this setting, RAGVUE's generic calibration is particularly important: by exposing instability in LLM-based judging across model and temperature configurations, it helps users avoid acting on brittle evaluations. Over time, our goal is to develop RAGVUE into a unified evaluation pipeline with baseline models, system comparisons, and optional multimodal support.

References

María Andrea Cruz Blandón, Jayasimha Talur, Bruno Charron, Dong Liu, Saab Mansour, and Marcello Federico. 2025. MEMERAG: A multilingual end-to-end meta-evaluation benchmark for retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22577–22595, Vienna, Austria. Association for Computational Linguistics.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. Ragas: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158.

Aoran Gan, Hao Yu, Kai Zhang, Qi Liu, Wenyu Yan, Zhenya Huang, Shiwei Tong, and Guoping Hu. 2025. Retrieval augmented generation evaluation in the era of large language models: A comprehensive survey. *arXiv preprint arXiv:2504.14891*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24(1).

Yannis Katsis, Sara Rosenthal, Kshitij Fadnis, Chulaka Gunasekara, Young-Suk Lee, Lucian Popa, Vraj Shah, Huaiyu Zhu, Danish Contractor, and Marina Danilevsky. 2025. mtrag: A multi-turn conversational benchmark for evaluating retrieval-augmented generation systems. *Transactions of the Association for Computational Linguistics*, 13:784–808.

Omar Khattab, Christopher Potts, and Matei Zaharia. 2021. Relevance-guided supervision for openqa with colbert. *Transactions of the association for computational linguistics*, 9:929–944.

Tom Kocmi and Christian Federmann. 2023. Large language models are state-of-the-art evaluators of translation quality. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 193–203, Tampere, Finland. European Association for Machine Translation.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Kun Li, Yunxiang Li, Tianhua Zhang, Hongyin Luo, Xixin Wu, James R. Glass, and Helen M. Meng. 2025. RAG-zeval: Enhancing RAG responses evaluator through end-to-end reasoning and ranking-based reinforcement learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 24936–24954, Suzhou, China. Association for Computational Linguistics.

Shuliang Liu, Xinze Li, Zhenghao Liu, Yukun Yan, Cheng Yang, Zheni Zeng, Zhiyuan Liu, Maosong Sun, and Ge Yu. 2025. Judge as a judge: Improving the evaluation of retrieval-augmented generation through the judge-consistency of large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 5788–5807, Vienna, Austria. Association for Computational Linguistics.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Yuxuan Liu, Tianchi Yang, Shaohan Huang, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. 2024. Calibrating LLM-based evaluator. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 2638–2656, Torino, Italia. ELRA and ICCL.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore. Association for Computational Linguistics.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

Cheng Niu, Yuanhao Wu, Juno Zhu, Siliang Xu, Kashun Shum, Randy Zhong, Juntong Song, and Tong Zhang. 2024. Ragtruth: A hallucination corpus for developing trustworthy retrieval-augmented language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10862–10878.

Jie Ouyang, Tingyue Pan, Mingyue Cheng, Ruiran Yan, Yucong Luo, Jiaying Lin, and Qi Liu. 2025. HoH: A dynamic benchmark for evaluating the impact of outdated information on retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6036–6063, Vienna, Austria. Association for Computational Linguistics.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. 2024. Llm evaluators recognize and favor their own generations. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Xiangyu Peng, Prafulla Kumar Choubey, Caiming Xiong, and Chien-Sheng Wu. 2025. Unanswerability evaluation for retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8452–8472, Vienna, Austria. Association for Computational Linguistics.

Sara Rosenthal, Avirup Sil, Radu Florian, and Salim Roukos. 2025. CLAPnq: Cohesive long-form answers from passages in natural questions for RAG systems. *Transactions of the Association for Computational Linguistics*, 13:53–72.

Dongyu Ru, Lin Qiu, Xiangkun Hu, Tianhang Zhang, Peng Shi, Shuaichen Chang, Cheng Jiayang, Cunxiang Wang, Shichao Sun, Huanyu Li, and 1 others. 2024. Ragchecker: A fine-grained framework for diagnosing retrieval-augmented generation. *Advances in Neural Information Processing Systems*, 37:21999–22027.

Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2024. ARES: An automated evaluation framework for retrieval-augmented generation systems. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 338–354, Mexico City, Mexico. Association for Computational Linguistics.

Kayla Schroeder and Zach Wood-Doughty. 2024. Can you trust llm judgments? reliability of llm-as-a-judge. *arXiv preprint arXiv:2412.12509*.

Maojia Song, Shang Hong Sim, Rishabh Bhardwaj, Hai Leong Chieu, Navonil Majumder, and Soujanya Poria. 2024. Measuring and enhancing trustworthiness of llms in rag through grounded attributions and learning to refuse. *arXiv preprint arXiv:2409.11242*.

Ionut Teodor Sorodoc, Leonardo F. R. Ribeiro, Rexhina Blloshmi, Christopher Davis, and Adrià de Gispert. 2025. GaRAGe: A benchmark with grounding annotations for RAG evaluation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 17030–17049, Vienna, Austria. Association for Computational Linguistics.

Jiaan Wang, Yunlong Liang, Fandong Meng, Zengkui Sun, Haoxiang Shi, Zhixu Li, Jinan Xu, Jianfeng Qu, and Jie Zhou. 2023. Is ChatGPT a good NLG evaluator? a preliminary study. In *Proceedings of the 4th New Frontiers in Summarization Workshop*, pages 1–11, Singapore. Association for Computational Linguistics.

Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. 2024. Evaluation of retrieval-augmented generation: A survey. In *CCF Conference on Big Data*, pages 102–120. Springer.

Zhirui Zeng, Jiamou Liu, Meng-Fen Chiang, Jialing He, and Zijian Zhang. 2025. S-RAG: A novel audit framework for detecting unauthorized use of personal data in RAG systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10375–10385, Vienna, Austria. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Kunlun Zhu, Yifan Luo, Dingling Xu, Yukun Yan, Zhenghao Liu, Shi Yu, Ruobing Wang, Shuo Wang, Yishan Li, Nan Zhang, Xu Han, Zhiyuan Liu, and Maosong Sun. 2025. RAGEval: Scenario specific RAG evaluation dataset generation framework. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8520–8544, Vienna, Austria. Association for Computational Linguistics.

A Summary of Metrics

Our summary of metrics is available in Table 1

B Computational Time plot

The computational time box plot is shown in Figure 4

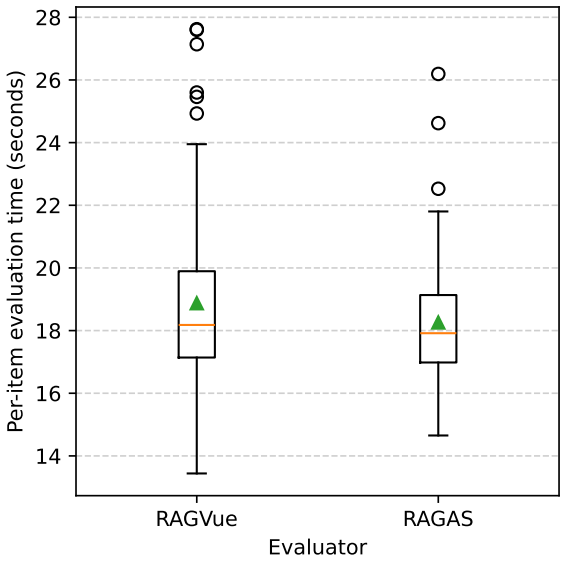


Figure 4: Distribution of per-item evaluation time for RAGVUE and RAGAS on the 100-query benchmark.

C Quantitative Analysis

The descriptive statistics are shown in Table 2, and the correlation results are provided in Table 3.

D Qualitative Case Studies - Examples

Table 4 provides item-level analyses for three representative evaluation cases drawn from our dataset. For each item, we report the behaviors of RAGAS and RAGVUE alongside the structured diagnostic signals RAGVUE generates.

Example 1. The model gives a vague answer ("No, even though there is no strong supporting evidence") that does not actually address the comparative question or follow from the retrieved facts. RAGAS still gives it a perfect faithfulness score

Metric	Inputs	What it Measures
		Retrieval Metrics
Retrieval Relevance	Q, C	Evaluates how useful each retrieved chunk is for addressing the information needs of the question, based on per-chunk relevance scoring.
Retrieval Coverage	Q, C	Assesses whether the retrieved context collectively provides sufficient coverage for all sub-aspects required to answer the question.
		Answer Metrics
Answer Relevance	Q, A	Measures how well the answer aligns with the intent and scope of the question, identifying missing, irrelevant, or off-topic content.
Answer Completeness	Q, A	Determines whether the answer fully addresses all aspects of the question without omissions.
Clarity	A	Evaluates the linguistic quality of the answer, including grammar, fluency, logical flow, coherence, and overall readability.
		Grounding & Stability Metrics
Strict Faithfulness	A, C	Evaluates how many factual claims in the answer are directly supported by the retrieved context, enforcing strict evidence alignment (entity accuracy and temporal correctness)
Calibration	Q, A, C	Examines the stability of metric by measuring variance across different judge configurations (model choice and temperature).

Table 1: Summary of the RAGVUE metrics.

System	Metric	Mean	Std
RAGAS	`faithfulness`	0.521	0.403
	`answer_relevancy`	0.240	0.307
	`context_relevance`	0.550	0.264
	`response_groundedness`	0.390	0.460
RAGVUE	`answer_completeness`	0.121	0.225
	`answer_relevance`	0.372	0.255
	`clarity`	0.698	0.100
	`retrieval_coverage`	0.503	0.279
	`retrieval_relevance`	0.420	0.316
	`strict_faithfulness`	0.400	0.492

Table 2: Descriptive statistics of RAGAS and RAGVUE metrics on the 100-query benchmark (mean and standard deviation).

(1.0) because it only checks for surface-level consistency and does not verify whether the conclusion is supported across multiple pieces of evidence, missing the needed **multi-hop reasoning**. RAGVUE instead marks the claim as fully unsupported (strict faithfulness $= 0.0$), shows that none of the key parts of the question are answered (completeness $= 0.0$), and indicates that retrieval only partly matched what was needed (retrieval coverage $= 0.33$). Together, these signals clearly reveal an unsupported reasoning error that RAGAS fails to catch.

Example 2. The system retrieves the right information, i.e, both RAGAS and RAGVUE show that the context is fully relevant. But the model still gives an incorrect answer ("Yes, even though there is no strong supporting evidence…"), which is not backed by the retrieved facts. RAGAS gives a mid-range faithfulness score (0.5) without explaining where the mistake comes from. RAGVUE makes this clear: it marks the claim as completely unsupported (strict faithfulness $= 0.0$), shows that the answer covers none of the required points (completeness $= 0.0$), and confirms that retrieval was correct. This directly identifies the problem as a *generation error*, not a retrieval issue.

Example 3. The model gives a vague answer ("probably true") even though the retrieved evidence does not actually say which of the two (dog or grey seal) would respond first. RAGAS gives the answer a mid-range faithfulness score (≈ 0.67) because it does not directly contradict any single fact, but this does not explain what went wrong. RAGVUE makes the issue clear: it marks the answer as fully unsupported (strict faithfulness $= 0.0$), shows that the model did not address the key parts of the question (completeness $= 0.0$), and indicates that only part of the retrieved information was relevant. As a result, RAGVUE pinpoints that the failure also comes from the model's reasoning, not only from retrieval, which is something RAGAS cannot show.

Across all three examples, RAGVUE provides structured diagnostics that clearly distinguish whether errors stem from retrieval, grounding, or reasoning. In contrast, RAGAS offers only scalar scores, which obscure these distinctions in practice.

Table 3: Spearman correlation between RAGAS metrics and RAGVUE metrics on the 100-query benchmark.

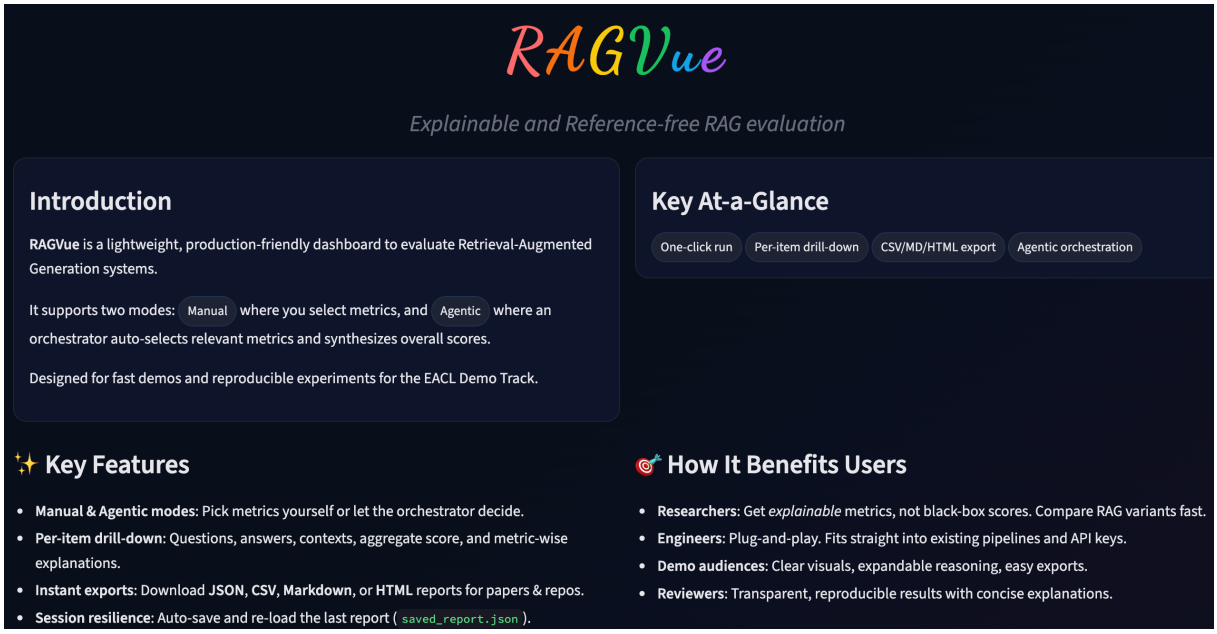
	RAGVUE metrics					
RAGAS	ans_comp.	ans_rel	clarity	ret_cov	ret_rel	strict_faith
faithfulness	0.553	0.668	0.094	0.298	0.025	0.739
answer_relevancy	0.704	0.644	0.172	0.193	0.053	0.958
context_relevance	0.082	0.062	0.171	0.006	0.708	0.035
response_groundedness	0.373	0.174	0.071	0.071	0.124	0.940

Table 4: Qualitative examples comparing RAGAS and RAGVUE on real evaluation outputs.

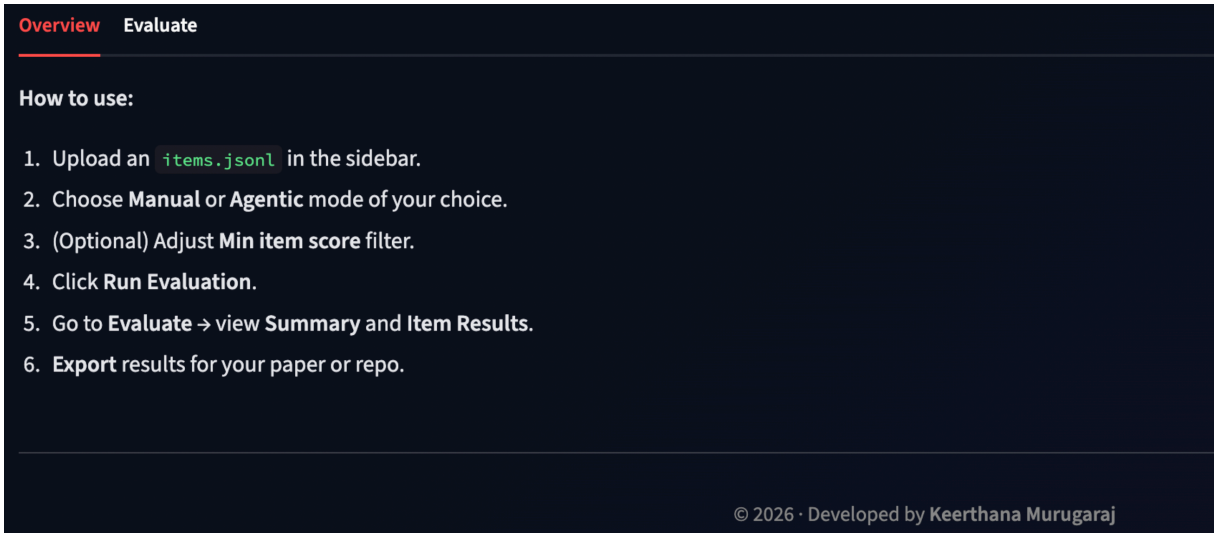
Question	**Model Answer**	**RAGAS Scores**	**RAGVUE Scores**
Are more people today related to Genghis Khan than Julius Caesar?	"No, even though there is no strong supporting evidence..."	Faithfulness: 1.0 Answer rel.: 0.0 Context rel.: 0.5 Groundedness: 0.0	Strict faith.: 0.0 Answer rel.: 0.40 Completeness: 0.00 Ret. cov.: 0.33 Ret. rel.: 0.33
Could the members of *The Police* perform lawful arrests?	"Yes, even though there is no strong supporting evidence..."	Faithfulness: 0.5 Answer rel.: 0.0 Context rel.: 1.0 Groundedness: 0.0	Strict faith.: 0.0 Answer rel.: 0.40 Completeness: 0.00 Ret. cov.: 0.33 Ret. rel.: 1.00
Would a dog respond to a bell before a grey seal?	"It is hard to say... probably true. The evidence is not entirely clear."	Faithfulness: 0.67 Answer rel.: 0.0 Context rel.: 0.5 Groundedness: 0.0	Strict faith.: 0.0 Answer rel.: 0.40 Completeness: 0.00 Ret. cov.: 0.50 Ret. rel.: 0.33

E Streamlit UI Images

The following figures illustrate the full RAGVUE Streamlit interface and its functionality. Figure 5a-5b provides the introduction page and the overview tab, which guide users through the workflow and usage instructions. Figure 6a–6d presents the core configuration components, including API key setup, data selection, manual and agentic mode configuration, and optional filters and report-saving tools. Figure 7a-7b shows the evaluation tab, which contains both the global summary across all processed cases and the detailed report for an individual (Q, A, C) example. Finally, Figure 8a-8b demonstrates the behavior of the agentic orchestrator for different input formats, highlighting its ability to select appropriate metrics based on the available fields.

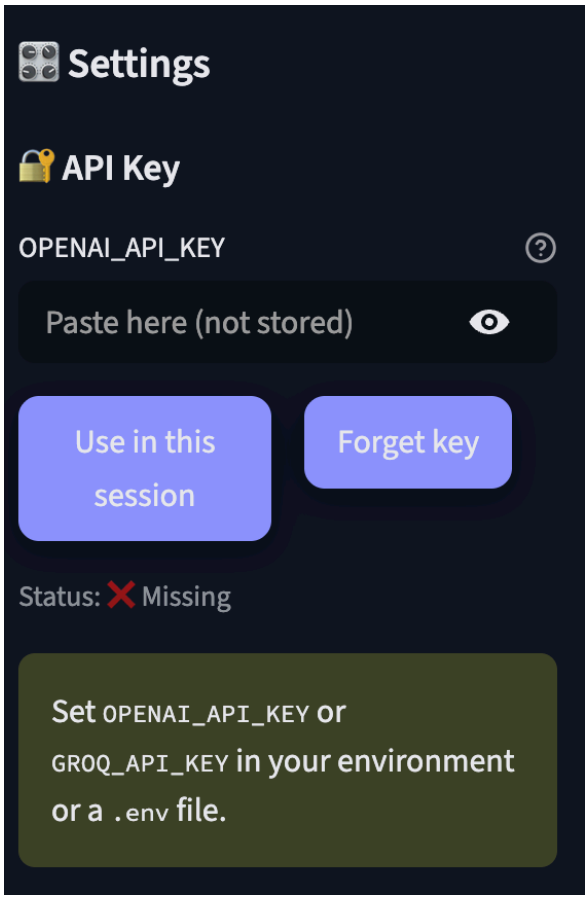


(a) Introduction page.

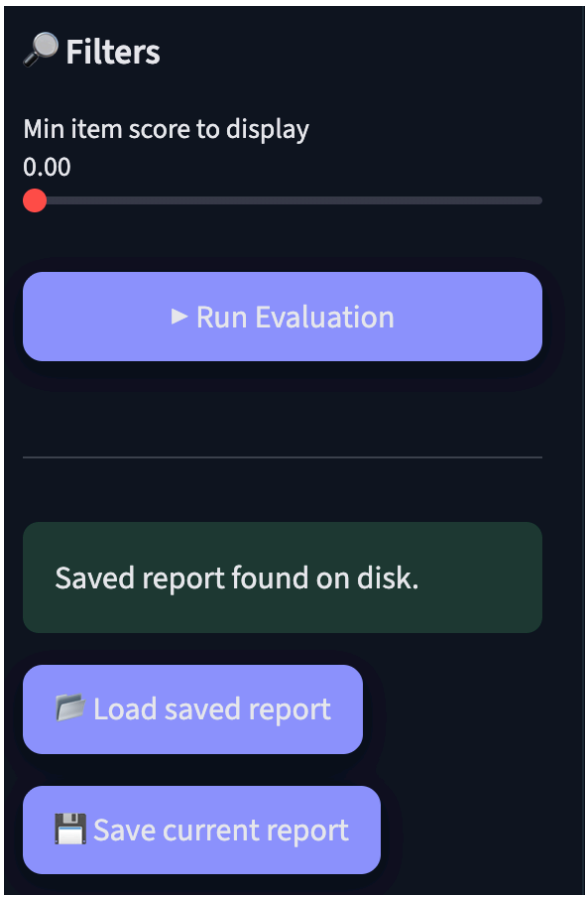


(b) Overview tab: usage instructions.

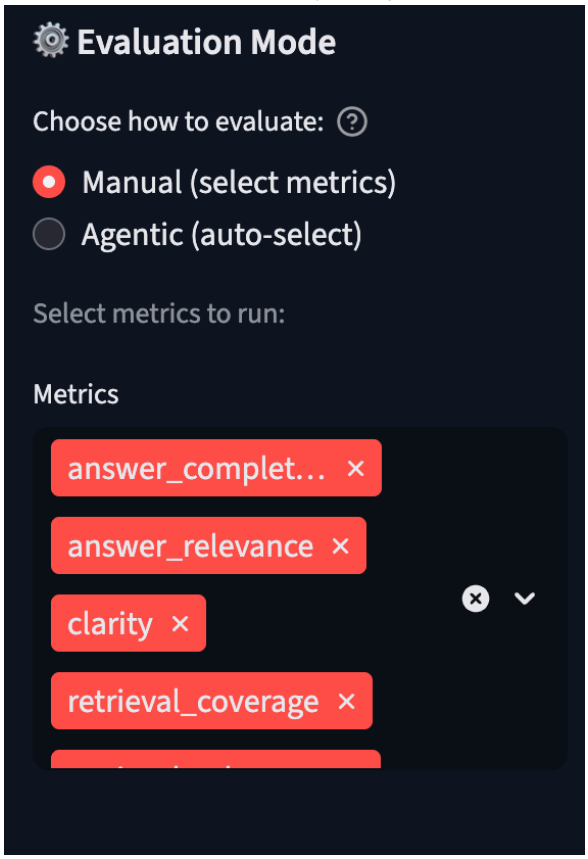
Figure 5: RAGVUE Streamlit UI: introduction page & overview tab.



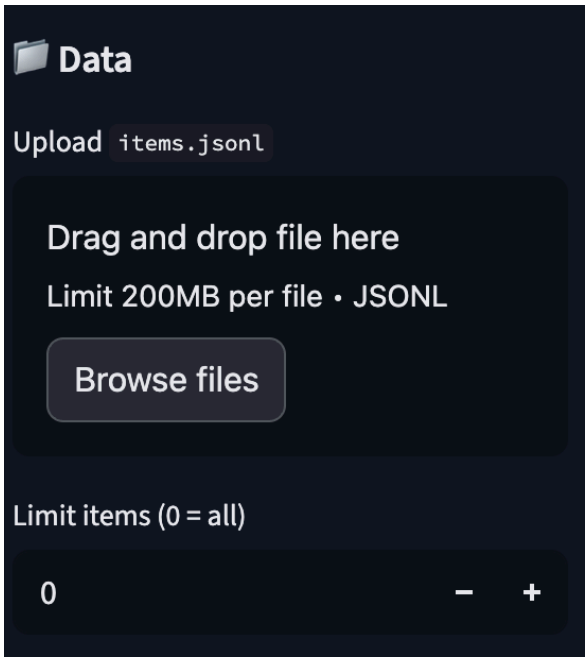
(a) API and key settings.



(b) Filters and saving options.

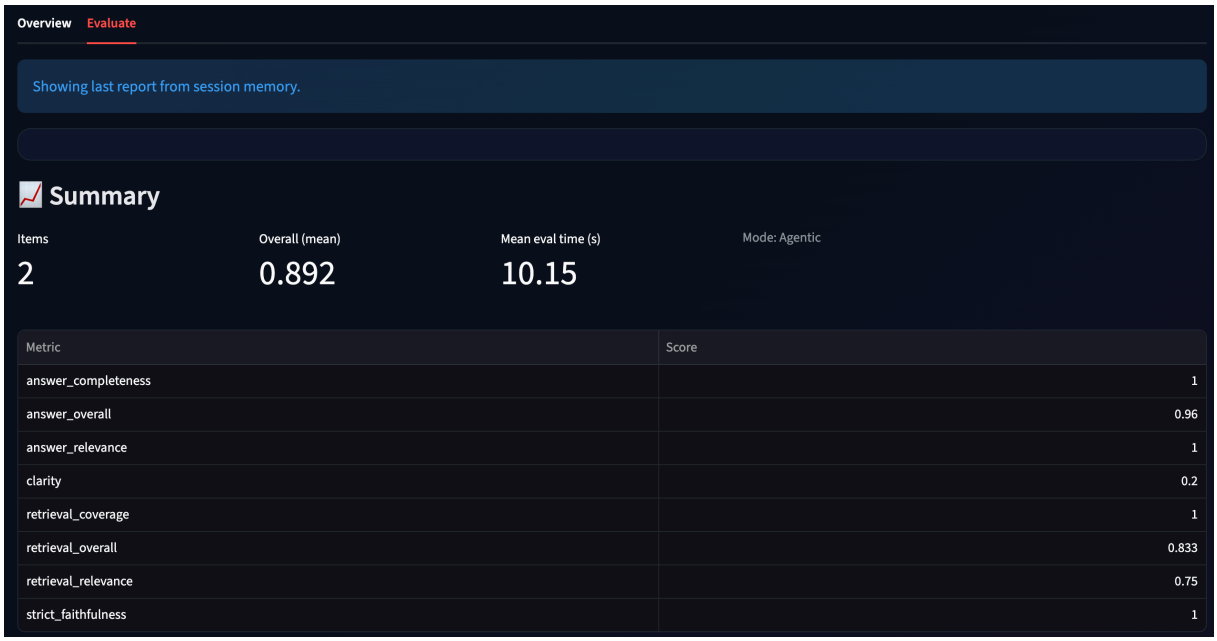


(c) Manual/agentic mode configuration.

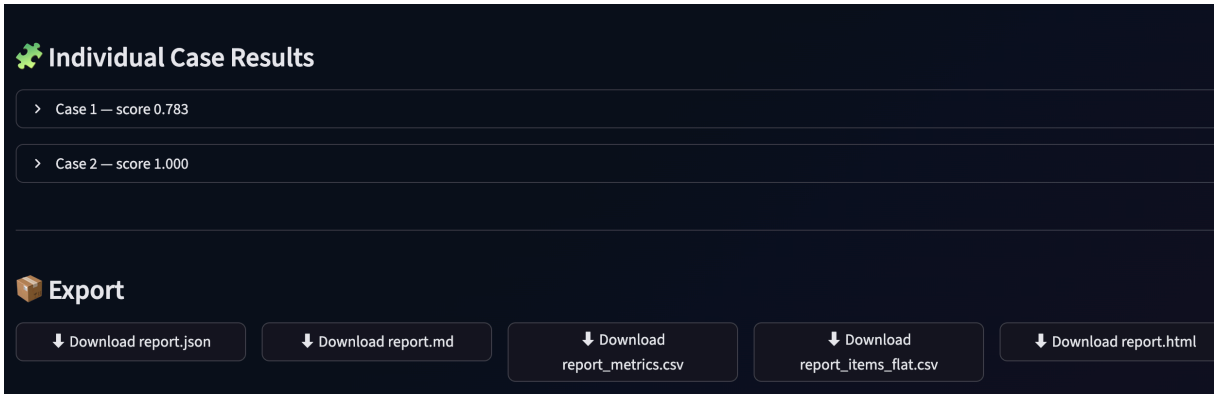


(d) Data upload option.

Figure 6: RAGVUE Streamlit user interface: API configuration, filter and save settings, manual/agentic mode selection, data upload options.

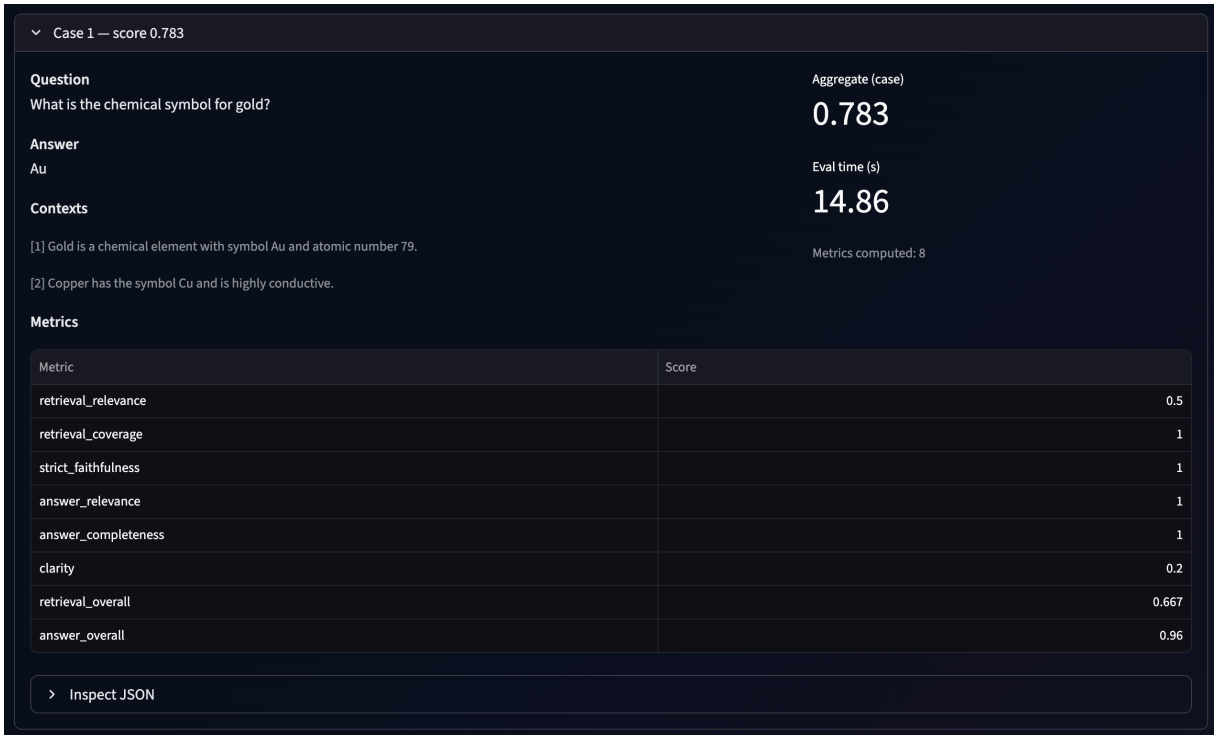


(a) Evaluate tab showing the global summary across all cases.

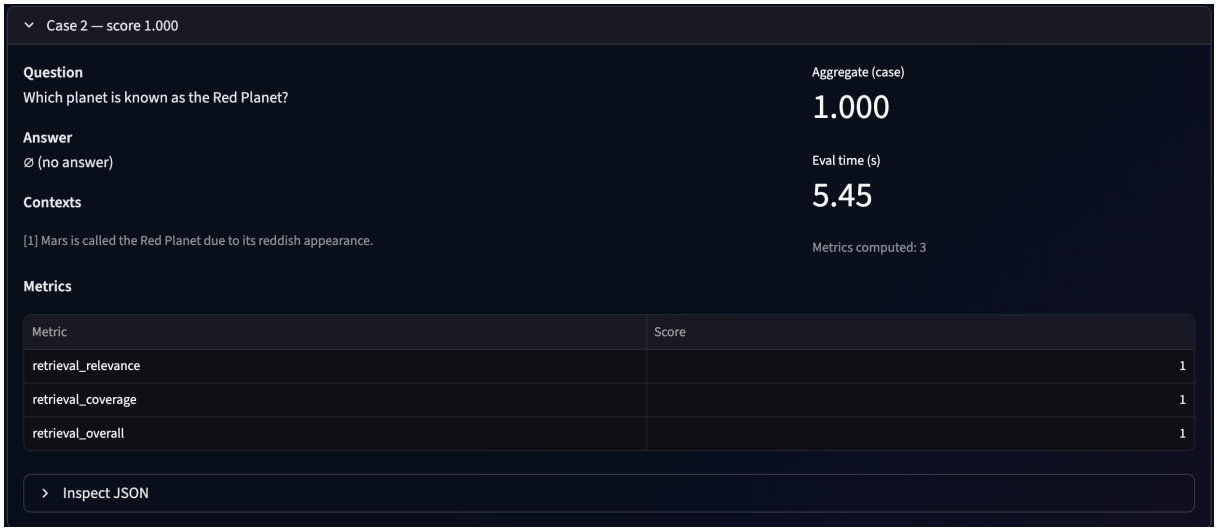


(b) Detailed individual case report for a single (Q, A, C) instance.

Figure 7: RAGVUE Streamlit user interface: evaluation summary view and individual case report.



(a) Agentic mode applied to (Q, A, C) triplets. The orchestrator correctly selects all relevant metrics.



(b) Agentic mode applied to (Q, C) triplets. The orchestrator selects only retrieval metrics and skips answer metrics.

Figure 8: Agentic mode behavior across different input configurations.