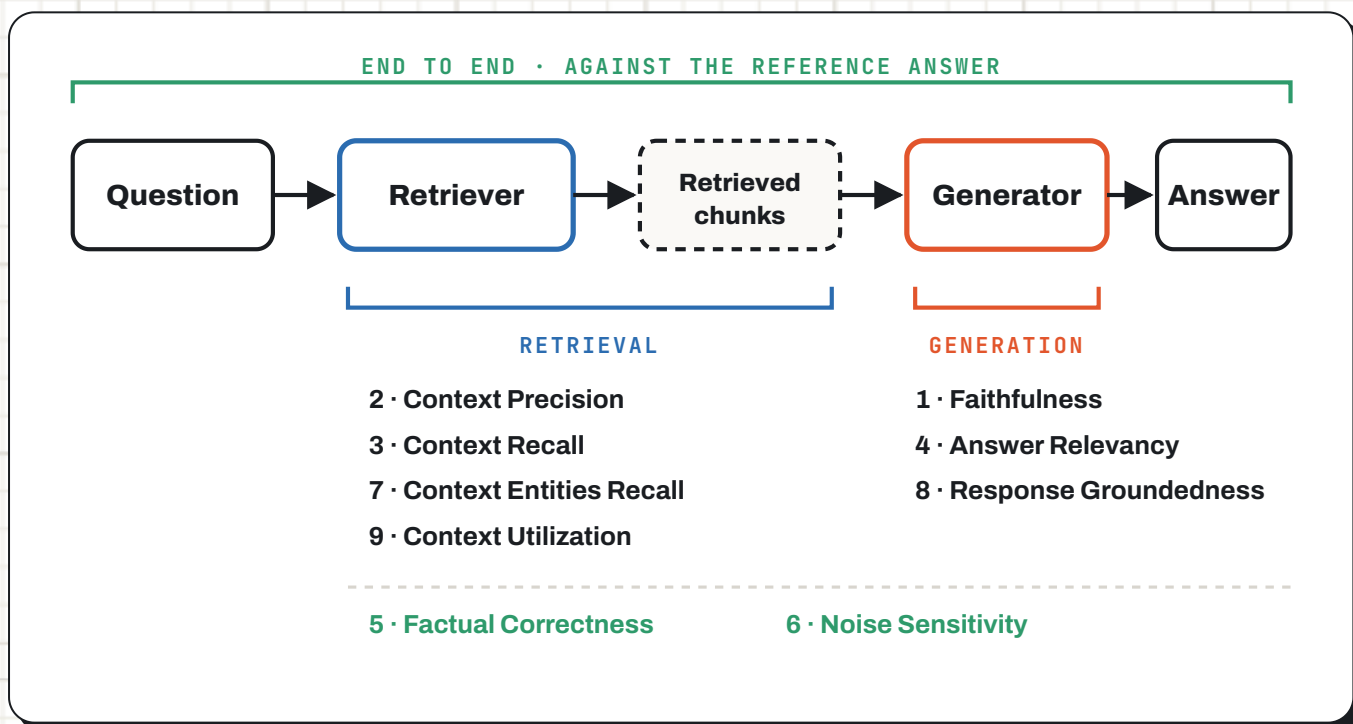


9 RAG Evaluation Metrics

HOW TO MEASURE EACH ONE

What each metric checks, the exact formula, one worked example scored end to end, and what to fix when a number is low.



Four inputs. Nine scores.

WHAT EACH METRIC NEEDS

user_input

The question the user asked.

retrieved_contexts

The chunks your retriever returned, in rank order.

response

The answer your RAG system produced.

reference

The ideal answer, written by a person. Your ground truth.

#	METRIC	JUDGES	NEEDS	BETTER
01	Faithfulness	GENERATION	question · contexts · response	↑
02	Context Precision	RETRIEVAL	question · contexts · reference	↑
03	Context Recall	RETRIEVAL	question · contexts · reference	↑
04	Answer Relevancy	GENERATION	question · response	↑
05	Factual Correctness	END TO END	response · reference	↑
06	Noise Sensitivity	END TO END	question · contexts · response · reference	↓
07	Context Entities Recall	RETRIEVAL	contexts · reference	↑
08	Response Groundedness	GENERATION	contexts · response	↑
09	Context Utilization	RETRIEVAL	question · contexts · response	↑

No reference answers yet? Start with 1, 4, 8 and 9. They need no ground truth, so you can run them on live traffic today. Add the other five once you have a labelled test set.

All scores run 0 to 1. Every metric except Noise Sensitivity is better when higher. Most use an LLM as the judge, so pick one judge model and keep it fixed when you compare runs.

The refund-bot test case

USED ON EVERY METRIC

QUESTION	How long do I have to get a refund on an annual plan?
REFERENCE	Annual plans can be refunded within 30 days of purchase.
RETRIEVED (RANK ORDER)	C1 Office hours are 9am–6pm, Monday to Friday. IRRELEVANT C2 Annual plans can be refunded within 30 days of purchase. RELEVANT C3 Archived 2022 FAQ: every refund carries a \$10 processing fee. STALE
RESPONSE	You can get a refund within 30 days of purchase [A]. Note there is a \$10 processing fee [B].

Claim **A** is right. Claim **B** is wrong: the fee was dropped, but the archived FAQ is still in the index. Keep an eye on which metrics catch that.

RETRIEVAL DID WE FETCH THE RIGHT CHUNKS, IN THE RIGHT ORDER?

02

Context Precision

0.50

Asks: are the relevant chunks ranked at the top?

EXAMPLE

Context Precision@K = $\sum (\text{Precision}@k \times v_k) \div \text{relevant chunks in top K}$

$v_k = 1$ if chunk k is relevant to the reference, else 0

IN THE EXAMPLE

Relevance by rank is [0, 1, 0]. Precision@2 = 1/2, one relevant chunk, so 0.50. Put C2 first and it becomes 1.00.

LOW? FIX THE RANKING

Add a reranker, or try hybrid search (keywords plus vectors).

03

Context Recall

1.00

Asks: did we retrieve everything the answer needs?

EXAMPLE

Context Recall = $\text{reference claims found in the contexts} \div \text{all reference claims}$

IN THE EXAMPLE

The reference has one claim (30 days). C2 contains it: 1 ÷ 1 = 1.00.

LOW? FIX COVERAGE

Revisit chunk size and overlap, raise top-k, add hybrid search, and check the document was indexed at all.

07 Context Entities Recall

1.00

EXAMPLE

Asks: do the retrieved chunks contain the names, numbers and dates the answer depends on?

Context Entities Recall = $|\text{reference entities} \cap \text{context entities}| \div |\text{reference entities}|$

IN THE EXAMPLE

Reference entities {annual plan, 30 days} both appear in C2: $2 \div 2 = 1.00$.

LOW? FIX FACT LOOKUP

Use keyword or entity-aware search and metadata filters. Pure vector search often misses exact names and numbers.

09 Context Utilization

0.58

EXAMPLE

Asks: how much of what we retrieved did the answer actually use? This is Context Precision with no reference needed.

Same formula as Context Precision, but $v_k = 1$ if chunk k was useful for the response (not the reference).

IN THE EXAMPLE

The response used C2 and C3: $[0, 1, 1]$. $(1/2 + 2/3) \div 2 = 0.58$.

LOW? CUT THE NOISE

Retrieve fewer chunks or rerank. No ground truth needed, so it works on live traffic.

01 Faithfulness

1.00

EXAMPLE

Asks: is every claim in the answer supported by the retrieved chunks?

Faithfulness = $\text{response claims supported by the contexts} \div \text{all response claims}$
An LLM splits the response into claims, then checks each one against the contexts.

IN THE EXAMPLE

A is backed by C2, B is backed by C3: $2 \div 2 = 1.00$. **Yet B is wrong.** Faithful means "stuck to the context", not "correct".

LOW? FIX GROUNDING

Tell the model to answer only from the context, ask for citations, and lower the temperature.

04 Answer Relevancy

≈0.85

ILLUSTRATIVE

Asks: does the answer address the question, without gaps or padding? It does not check truth.

Answer Relevancy = mean cosine similarity(question, q_i) over $i = 1..N$

An LLM writes N questions the response would answer (default $N = 3$), then compares them to the real one.

IN THE EXAMPLE

The fee sentence answers a question nobody asked, which pulls the score down a little. The exact value depends on your embedding model.

LOW? FIX FOCUS

Prompt the model to answer the question first and only. Evasive "I'm not sure" answers score 0.

08 Response Groundedness

1.00

EXAMPLE

Asks: taken as a whole, is the response supported by the contexts?

An LLM judge rates support as 0 (none), 1 (partial) or 2 (full). Ragas asks with two different prompts, averages the ratings, and scales the result to 0-1.

IN THE EXAMPLE

Every sentence traces to C2 or C3, so both ratings are 2 and the score is 1.00. Same blind spot as Faithfulness.

LOW? SAME FIXES AS FAITHFULNESS

Use it as the quick overall check. Faithfulness tells you which claim failed.

END TO END

IS THE FINAL ANSWER ACTUALLY RIGHT?

05 Factual Correctness

0.67

EXAMPLE · F1

Asks: do the answer's claims match the reference answer?

TP = response claims in the reference · FP = not in it · FN = reference claims missed
Precision = $TP \div (TP + FP)$ · **Recall** = $TP \div (TP + FN)$ · default score = F1

IN THE EXAMPLE

TP = 1 (A), FP = 1 (B), FN = 0. Precision 0.50, recall 1.00, F1 = 0.67. **Caught it.**

WATCH OUT

A true claim that is missing from the reference also counts as FP, so keep your references complete.

06 Noise Sensitivity**0.50****Asks:** how often does retrieved noise lead the model into a wrong claim?

EXAMPLE · LOWER = BETTER

Noise Sensitivity = wrong claims traced to retrieved chunks ÷ all response claims
mode="relevant" (default): errors from relevant chunks · "irrelevant": from the rest

IN THE EXAMPLE

B is wrong and came from C3, an irrelevant chunk.

Irrelevant mode: $1 \div 2 = 0.50$. **Caught it, and found the source.**

HIGH? CLEAN THE INPUT

Remove stale documents, filter or rerank, and retrieve fewer chunks.

FIG. 03 · READ THE SCORES TOGETHER

The model wasn't the problem.

DIAGNOSIS

Retrieval found the answer (recall 1.00) but ranked noise above it (precision 0.50) and pulled in a stale FAQ. **The generator did its job:** it stuck to what it was given (faithfulness 1.00). **So the answer came out wrong** (factual correctness 0.67), and noise sensitivity points at the stale chunk.

Fix: delete archived docs from the index and add a reranker. Switching to a bigger LLM would change nothing here. Architecture first.

FIG. 04 · SYMPTOM → FIX

IF YOU SEE

DO THIS

Low context recall

Retriever misses info: tune chunking, raise top-k, add hybrid search.

Low precision or utilization

Noise ranks high: add a reranker, retrieve fewer chunks.

Low faithfulness or groundedness

Model invents facts: answer-only-from-context prompt, citations.

Low answer relevancy

Off-topic or padded: answer first, nothing else.

High faithfulness, low correctness

Your documents are wrong or stale. Fix the data, not the model.

All nine, one script.

CHECKED AGAINST RAGAS 0.4.3

```
# pip install "ragas>=0.4" openai          then set OPENAI_API_KEY
import asyncio, inspect
from openai import AsyncOpenAI
from ragas.llms.base import llm_factory
from ragas.embeddings.base import embedding_factory
from ragas.metrics.collections import (
    Faithfulness, ContextPrecisionWithReference, ContextRecall, AnswerRelevancy,
    FactualCorrectness, NoiseSensitivity, ContextEntityRecall,
    ResponseGroundedness, ContextUtilization)

client = AsyncOpenAI()
llm = llm_factory("gpt-4o-mini", client=client) # the judge: keep it fixed across runs
emb = embedding_factory("openai", model="text-embedding-3-small", client=client)

sample = {
    "user_input": "How long do I have to get a refund on an annual plan?",
    "retrieved_contexts": [                                # in rank order
        "Office hours are 9am-6pm, Monday to Friday.",
        "Annual plans can be refunded within 30 days of purchase.",
        "Archived 2022 FAQ: every refund carries a $10 processing fee.",
    ],
    "response": "You can get a refund within 30 days of purchase. "
                "Note there is a $10 processing fee.",
    "reference": "Annual plans can be refunded within 30 days of purchase.",
}

metrics = [Faithfulness(llm=llm), ContextPrecisionWithReference(llm=llm),
           ContextRecall(llm=llm), AnswerRelevancy(llm=llm, embeddings=emb),
           FactualCorrectness(llm=llm), NoiseSensitivity(llm=llm, mode="irrelevant"),
           ContextEntityRecall(llm=llm), ResponseGroundedness(llm=llm),
           ContextUtilization(llm=llm)]

async def main():
    for m in metrics: # each metric takes only the inputs it needs
        wants = inspect.signature(m.ascore).parameters
        result = await m.ascore(**{k: v for k, v in sample.items() if k in wants})
        print(f"{m.name:<36} {result.value:.2f}")

asyncio.run(main())
```

Your numbers will be close to the hand-worked ones, not identical: the judge LLM decides what counts as a claim, an entity or a relevant chunk. Run each test case a few times and compare averages, not single scores.

Found this useful?

Follow SKD Neuron for more explainers on RAG, AI agents and AI system design. Save this for your next interview.

@SKDNEURON
SKDNEURON.COM