RAGAS Metrics Guide

Complete guide to evaluating RAG applications

Overview

RAGAS (Retrieval Augmented Generation Assessment) provides metrics to evaluate both the retriever and the LLM in your RAG pipeline.

Retriever Metrics: Context Precision, Context Recall, Context Entities Recall
LLM Metrics: Answer Relevancy, Faithfulness, Noise Sensitivity

1. Answer Relevancy

What is it?

Measures if the chatbot's answer actually addresses the user's question.
Score: 0-1 (higher = better)

How it Works

1. LLM generates 3 questions from the answer
2. Compares with original question using cosine similarity
3. Averages scores = final score

What You Provide

user_input: The question asked
response: Chatbot's answer

Code Example

```
from ragas.metrics import AnswerRelevancy
scorer = AnswerRelevancy(llm=llm)
result = await scorer.ascore(
    user_input="What is your refund policy?",
    response="Refunds available within 30 days."
)
```

Example: HIGH Score

Q: How to cancel flight? | A: Cancel via app under My Trips.

Generated Qs match original → Score: 0.92

Example: LOW Score

Q: How to cancel flight? | A: Flynas offers meals on flights.

Generated Qs don't match → Score: 0.15
Reference: RAGAS Docs - Answer Relevancy

2. Faithfulness

What is it?

Measures if the answer is factually correct based on retrieved context.
Score: 0-1 (higher = more faithful)

How it Works

1. Breaks answer into individual statements
2. Checks if each statement is supported by context
3. Score = (supported statements) / (total statements)

What You Provide

response: Chatbot's answer
retrieved_contexts: Documents the RAG retrieved

Code Example

```
from ragas.metrics import Faithfulness
scorer = Faithfulness(llm=llm)
result = await scorer.ascore(
    response=test_case["response"],
    retrieved_contexts=test_case["retrieved_contexts"]
)
```

Example: HIGH Score

Context: Free cancel within 24hrs | Answer: Cancel free within 24hrs.

All statements supported → Score: 1.0

Example: LOW Score (Hallucination)

Context: Free cancel within 24hrs | Answer: Cancel free anytime.

Statement NOT in context → Score: 0.0
Reference: RAGAS Docs - Faithfulness

3. Context Precision

What is it?

Measures if the retriever ranks relevant documents at the top.
Score: 0-1 (higher = better ranking)

How it Works

1. Checks each retrieved document for relevance
2. Calculates precision at each position
3. Relevant docs at top = HIGH, at bottom = LOW

What You Provide

user_input: The question
retrieved_contexts: Documents in order
reference: Ground truth answer

Code Example

```
from ragas.metrics import LLMContextPrecisionWithoutReference
scorer = LLMContextPrecisionWithoutReference(llm=llm)
result = await scorer.ascore(
    user_input=test_case["question"],
    retrieved_contexts=test_case["retrieved_contexts"]
)
```

Example: HIGH Score

Retrieved: [Cancellation policy (relevant), Baggage, Meals]

Relevant doc at position 1 → Score: ~1.0

Example: LOW Score

Retrieved: [Meals, Baggage, Cancellation policy (relevant)]

Relevant doc at position 3 → Score: ~0.3
Reference: RAGAS Docs - Context Precision

4. Context Recall

What is it?

Measures if the retriever fetched ALL relevant documents needed.
Score: 0-1 (higher = more complete)

How it Works

1. Breaks reference into individual claims
2. Checks if each claim is in retrieved contexts
3. Score = (claims found) / (total claims)

What You Provide

retrieved_contexts: Documents retrieved
reference: Ground truth answer (required!)

Code Example

```
from ragas.metrics import LLMContextRecall
scorer = LLMContextRecall(llm=llm)
result = await scorer.ascore(
    retrieved_contexts=test_case["retrieved_contexts"],
    reference=test_case["reference"]
)
```

Example: HIGH Score

Reference has 2 claims, both found in retrieved docs.

2/2 claims found → Score: 1.0

Example: LOW Score

Reference has 2 claims, only 1 found in retrieved docs.

1/2 claims found → Score: 0.5

Precision vs Recall

Precision: Are relevant docs at TOP? | **Recall:** Did we get ALL relevant docs?

Reference: RAGAS Docs - Context Recall

5. Context Entities Recall

What is it?

Measures if retriever fetched docs with all important entities (names, dates, places).
Score: 0-1 (higher = more entities)

How it Works

1. Extracts entities from reference (names, dates, places)
2. Checks how many appear in retrieved contexts
3. Score = (entities found) / (total entities)

What You Provide

retrieved_contexts: Documents retrieved
reference: Ground truth with entities

Code Example

```
from ragas.metrics import ContextEntityRecall
scorer = ContextEntityRecall(llm=llm)
result = await scorer.ascore(
    retrieved_contexts=test_case["retrieved_contexts"],
    reference=test_case["reference"]
)
```

Example: HIGH Score

Reference: [Flynas, Flight 123, Riyadh, 10:00 AM] = 4 entities

All 4 found in docs → Score: 1.0

Example: LOW Score

Reference: [Flynas, Flight 123, Riyadh, 10:00 AM] = 4 entities

Only Flynas found → Score: 0.25
Reference: RAGAS Docs - Context Entities Recall

6. Noise Sensitivity

What is it?

Measures how often LLM makes errors when given noisy/irrelevant docs.
Score: 0-1 (LOWER = better, opposite of others!)

How it Works

1. Checks each claim in LLM response
2. Verifies if correct against ground truth
3. Score = (incorrect claims) / (total claims)

What You Provide

user_input: Question
response: LLM answer
reference: Ground truth
retrieved_contexts: Docs including noise

Code Example

```
from ragas.metrics import NoiseSensitivity
scorer = NoiseSensitivity(llm=llm)
result = await scorer.ascore(
    user_input=test_case["question"],
    response=test_case["response"],
    reference=test_case["reference"],
    retrieved_contexts=test_case["retrieved_contexts"]
)
```

Example: LOW Score (GOOD)

Retrieved: [Baggage policy, Meal menu (noise)] | Answer: Only about baggage.

LLM ignored noise → Score: 0.0 (GOOD)

Example: HIGH Score (BAD)

Retrieved: [Baggage policy, Meal menu (noise)] | Answer: Baggage + meals.

LLM confused by noise → Score: 0.5 (BAD)
Reference: RAGAS Docs - Noise Sensitivity

Quick Reference Summary

Metric Comparison

Answer Relevancy - Does answer address the question? (Tests: LLM) **Faithfulness** - Is answer factually correct from context? (Tests: LLM) **Context Precision** - Are relevant docs ranked at top? (Tests: Retriever) **Context Recall** - Did we fetch ALL relevant docs? (Tests: Retriever) **Context Entities Recall** - Did we fetch all entities? (Tests: Retriever) **Noise Sensitivity** - Does LLM resist bad docs? (Tests: LLM, Lower=Better)

What Each Metric Needs

Answer Relevancy: user_input, response **Faithfulness:** response, retrieved_contexts **Context Precision:** user_input, retrieved_contexts, reference **Context Recall:** retrieved_contexts, reference **Context Entities Recall:** retrieved_contexts, reference **Noise Sensitivity:** user_input, response, reference, retrieved_contexts

Key Insight

To use most metrics, you need to capture **retrieved_contexts** from your RAG pipeline. Modify your chatbot code to log the documents fetched during retrieval.