RAGAS: Context Precision Metric

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 score, at bottom = LOW score

What You Provide

user_input: The question asked
retrieved_contexts: Documents returned by retriever (in order)
reference: Ground truth answer

Step 1: Capture Data from Your RAG Pipeline

```
# In your RAG chatbot code
docs = vector_db.similarity_search(query)
retrieved_contexts = [doc.page_content for doc in docs]

# Save to logs (preserve order!)
save_to_logs({
    "question": query,
    "retrieved_contexts": retrieved_contexts,
    "reference": ground_truth_answer
})
```

Step 2: Test with RAGAS

```
from ragas.metrics import LLMContextPrecisionWithoutReference
from ragas.llms import llm_factory
from openai import AsyncOpenAI

client = AsyncOpenAI()
llm = llm_factory("gpt-4o-mini", client=client)
scorer = LLMContextPrecisionWithoutReference(llm=llm)

test_case = get_from_logs()

result = await scorer.ascore(
    user_input=test_case["question"],
    retrieved_contexts=test_case["retrieved_contexts"]
)
print(result)
```

Example 1: HIGH Score (Good Ranking)

Question: How to cancel my flight?

Retrieved Docs:
1. Cancellation policy (RELEVANT)
2. Baggage info (irrelevant)
3. Meal options (irrelevant)

Result: Relevant doc at position 1 → Score: ~1.0 (HIGH)

Example 2: LOW Score (Bad Ranking)

Question: How to cancel my flight?

Retrieved Docs:
1. Meal options (irrelevant)
2. Baggage info (irrelevant)
3. Cancellation policy (RELEVANT)

Result: Relevant doc at position 3 → Score: ~0.3 (LOW)

Key Insight

Context Precision evaluates your RETRIEVER, not the LLM. If relevant documents are buried at the bottom, your vector search needs improvement. Order matters!

Reference

https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/context_precision/