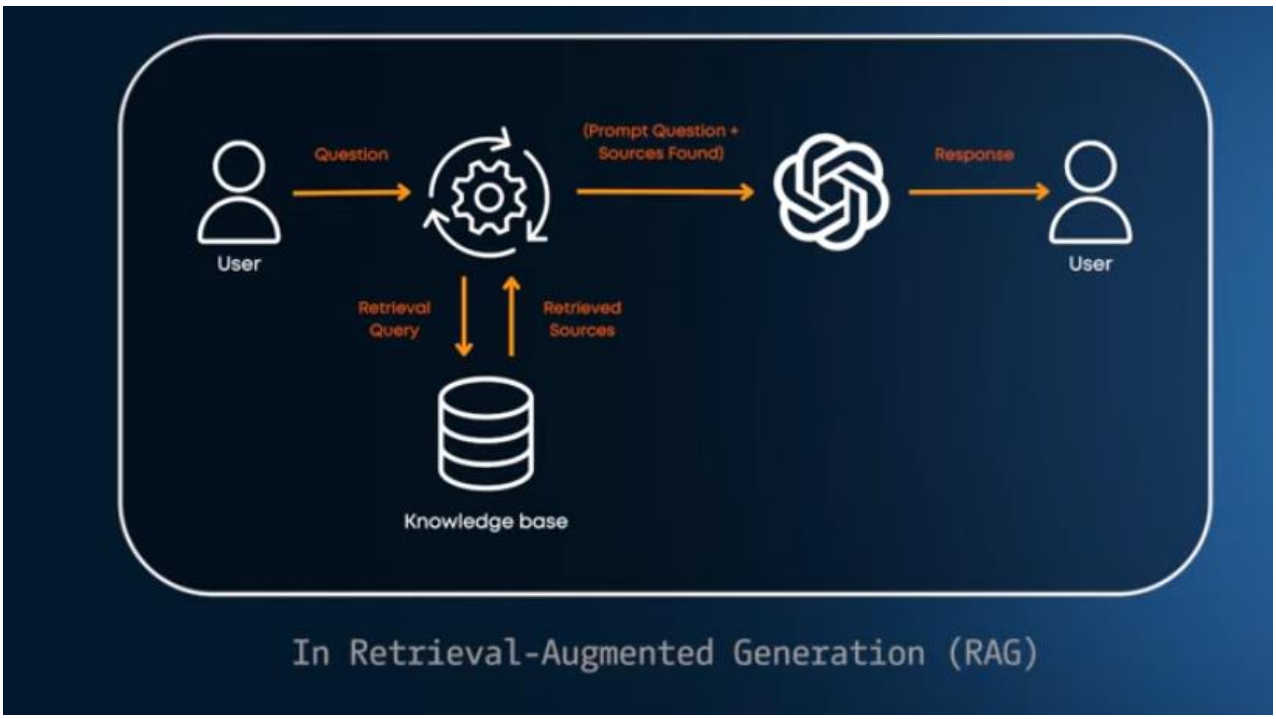
Key Metrics and Evaluation Methods for RAG

What's AI by Louis-François ...
68.7K subscribers

Join Subscribe 358 Share Ask Download ...

13,165 views Nov 21, 2024 #ai #rag #llm
Build Your First Scalable Product with LLMs: https://academy.towardsai.net/courses...
Master LLMs and Get Industry-ready Now: https://academy.towardsai.net/?ref=1f...
Our ebook: https://academy.towardsai.net/courses...

Video 2/10 of the "From Beginner to Advanced LLM Developer" course by Towards AI (linked above).
The most practical and in-depth LLM Developer course out there (~90 lessons) for software developers, machine learning engineers, data scientists, aspiring founders or AI/Computer Science students. We've gathered everything we worked on building products and AI systems and put them into one super practical industry-focused course. Right now, this means working with Python, OpenAI, Llama 3, Gemini, Perplexity, LlamaIndex, Gradio, and many other amazing tools (we are unaffiliated and will introduce all the best LLM tool options). It also means learning many new non-technical skills and habits unique to the world of LLMs.



EVALUATION METRICS FOR RAG SYSTEMS

Retrieval Metrics

- Precision
- Recall
- Hit Rate
- Mean Reciprocal Rank (MRR)
- Normalized Discounted Cumulative Gain (NDCG)

Generation Metrics

- Faithfulness
- Answer Relevancy
- Answer Correctness

RAG Model

+

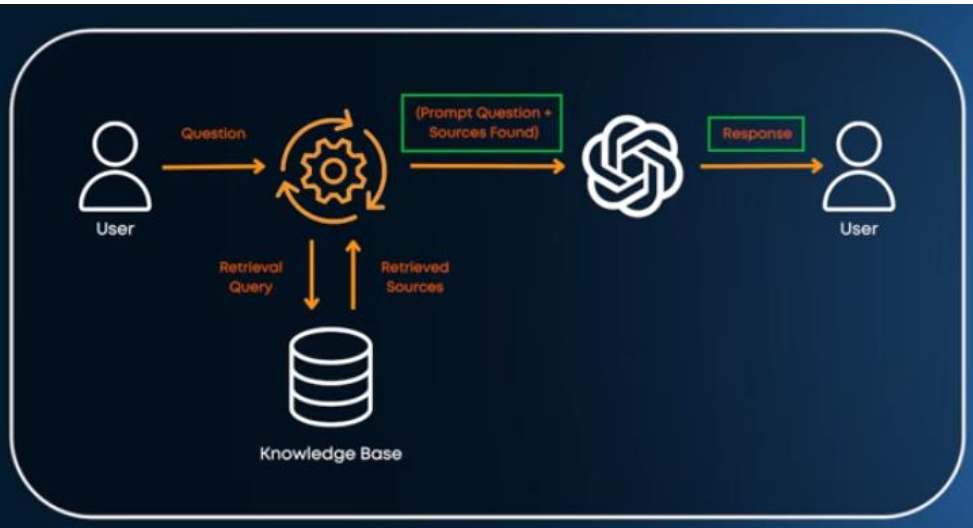
retrieval mechanisms

(i.e. Google search, vector search, graph search, hybrid search)

generative models

(i.e.. GPT-4)

RESPONSE QUALITY **RELEVANCE**



In Retrieval-Augmented Generation (RAG)

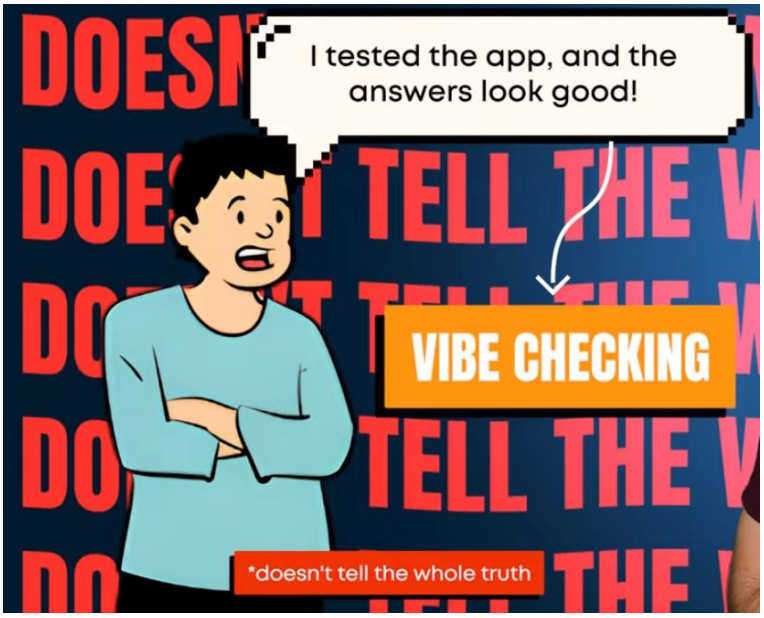
Most useful for,

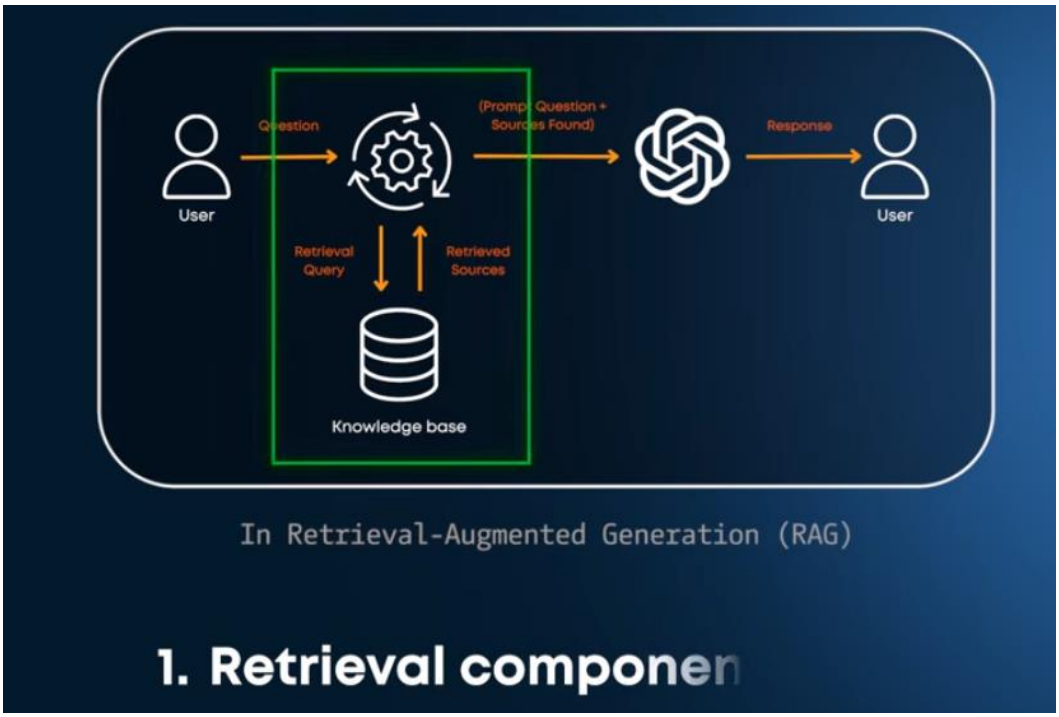
- Private data
- Advanced topics the LLM might not have seen during its training

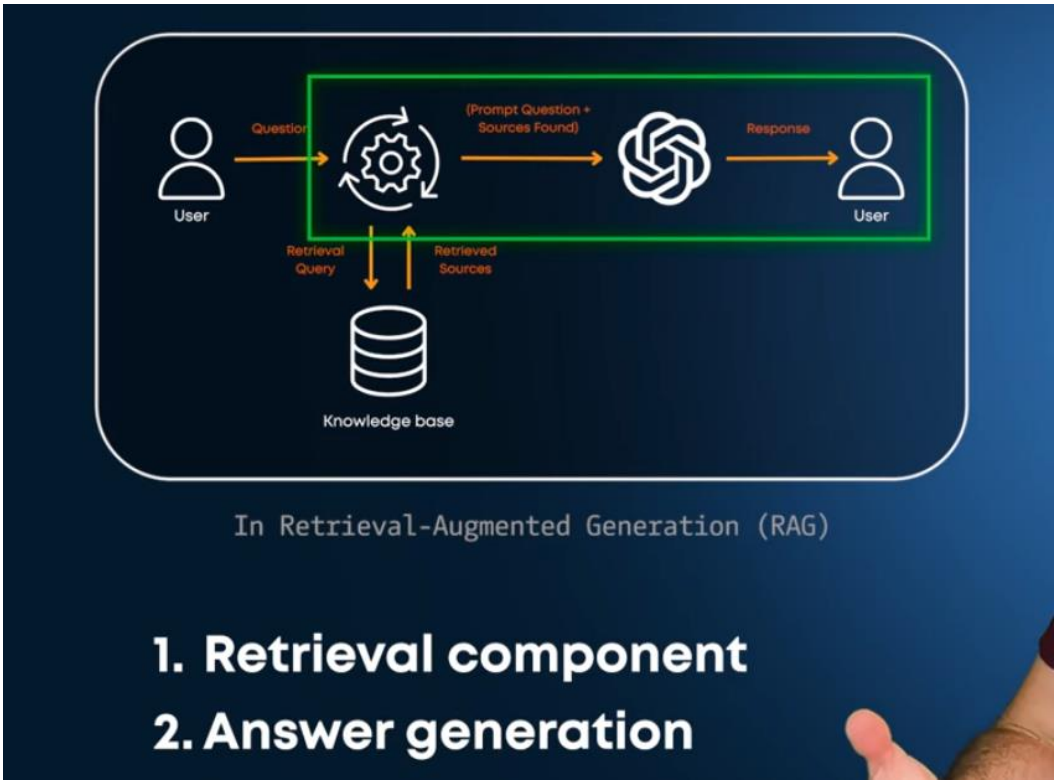
WHY IS EVALUATION IMPORTANT?

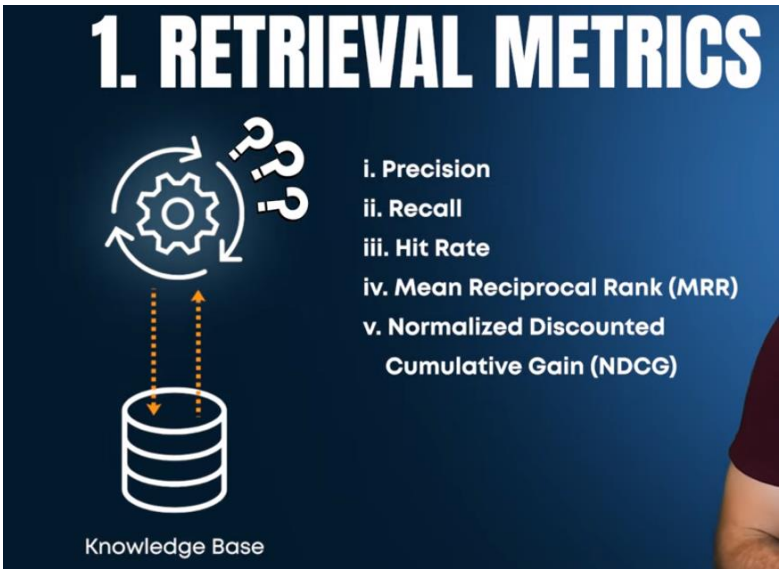
ANSWER:

Can be the difference between a nice demo and a highly useful LLM tool or product.









i. Precision

$$\text{Precision} = \frac{(\text{True Positives})}{(\text{True Positives} + \text{False Positives})}$$

- Useful to ensure highly relevant information retrieval

- Useful for critical applications like medical diagnosis

ii. Recall

$$\text{Recall} = \frac{(\text{True Positives})}{(\text{True Positives} + \text{False Negatives})}$$

- Crucial where missing key information can be costly

  i.e. Legal research

iii. Hit Rate

- Proportion of queries for which at least one relevant document is retrieved within the top few results

- System's ability to find relevant documents

iv. Mean Reciprocal Rank (MRR)

$$\text{MRR} = (1 / Q) * \Sigma\, (1 / \text{rank of the first relevant document})$$

where:

Q = total number of queries

- Useful for systems where users prioritize top results

v. Normalized Discounted Cumulative Gain (NDCG)

- Takes relevance and the ranking of retrieved documents into account

- Useful where the order of results matters

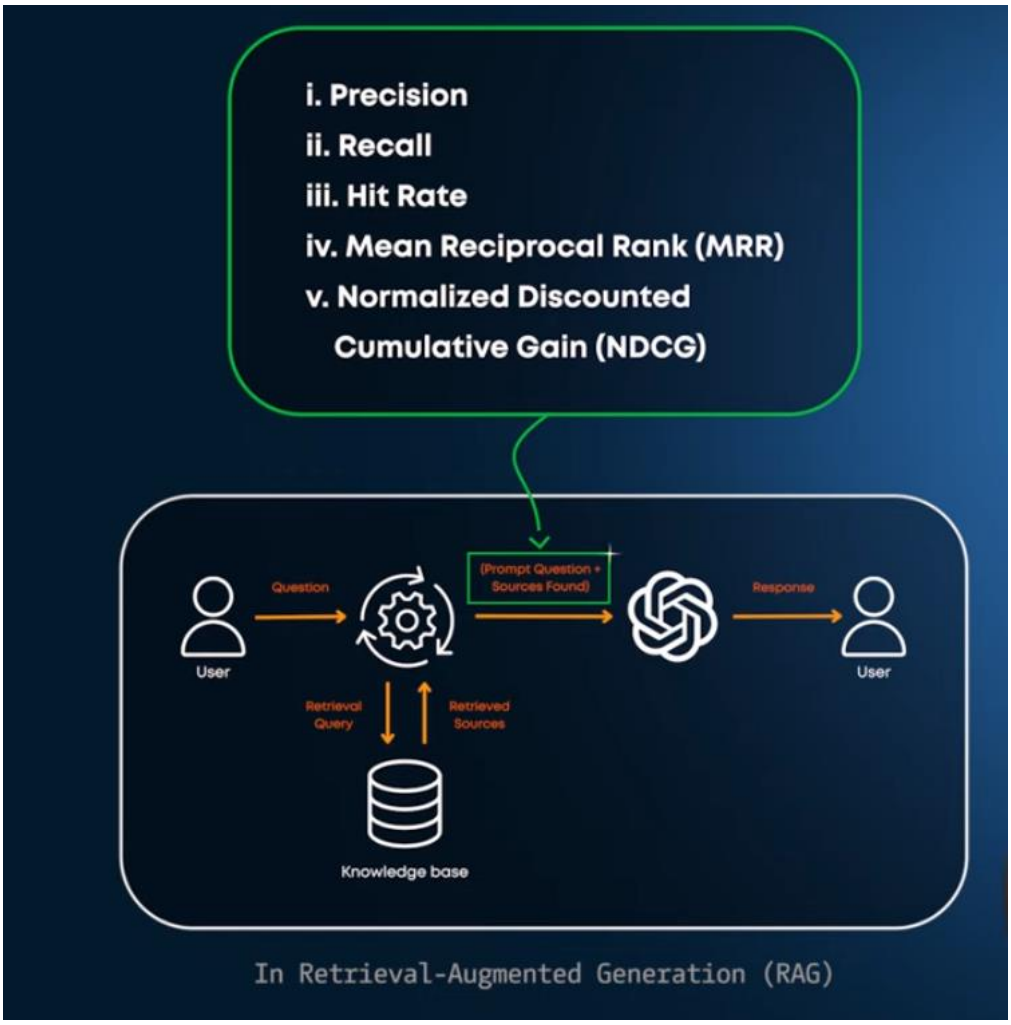
  i.e. Recommendation systems

MRR

- MRR is key for systems where top document rank matters most
- i.e. Search engines (e.g. top 1-3 Google results)

NDCG

- NDCG measures ranking quality of all documents by considering both relevance and order
- Ideal for applications like Recommendation systems where the entire list is important (e.g. top 10 lists)



2. GENERATION METRICS

i. Faithfulness

ii. Answer Relevancy

iii. Answer Correctness

i. Faithfulness

Hint:

Question: Where and when was Einstein born?

Context: Albert Einstein (born 14 March 1879) was a German-born theoretical physicist, widely held to be one of the greatest and most influential scientists of all time.

High faithfulness answer: Einstein was born in Germany on 14th March 1879.

Low faithfulness answer: Einstein was born in Germany on 20th March 1879.

- Measures the integrity of the answer
- Ensure responses reflect accurate, relevant document information without errors

ii. Answer Relevancy

Hint:

Question: Where is France and what is it's capital?
Low relevance answer: France is in western Europe.
High relevance answer: France is in western Europe and Paris is its capital.

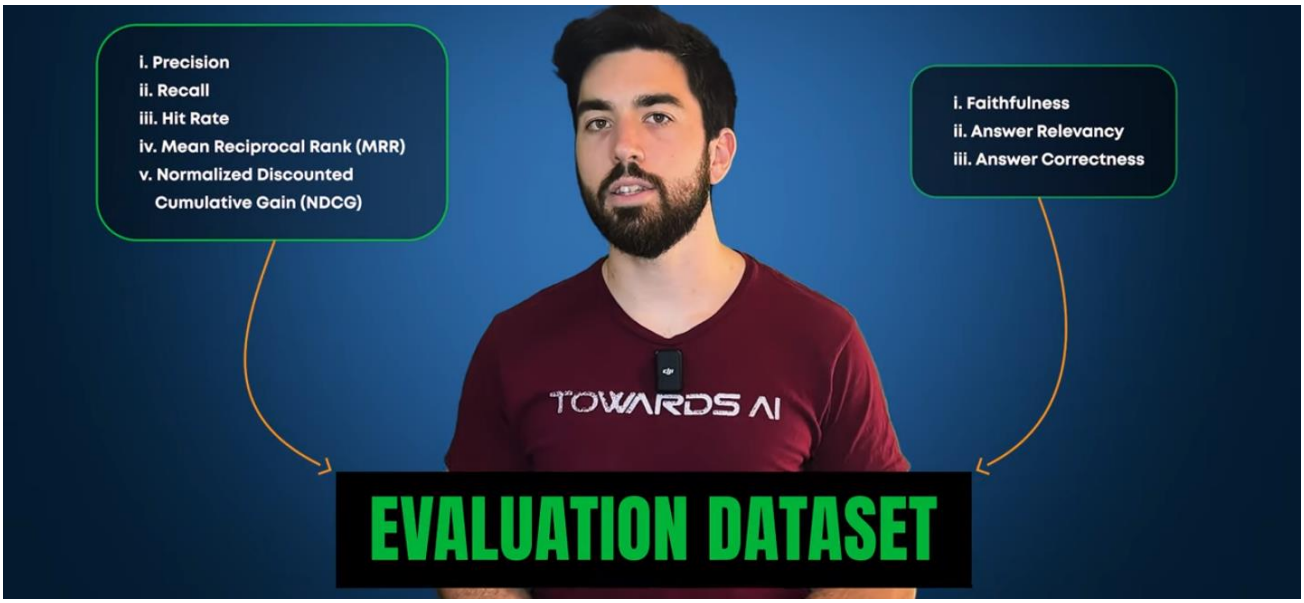
- Evaluate answer's relevance to original query
- Ensure system generates pertinent responses

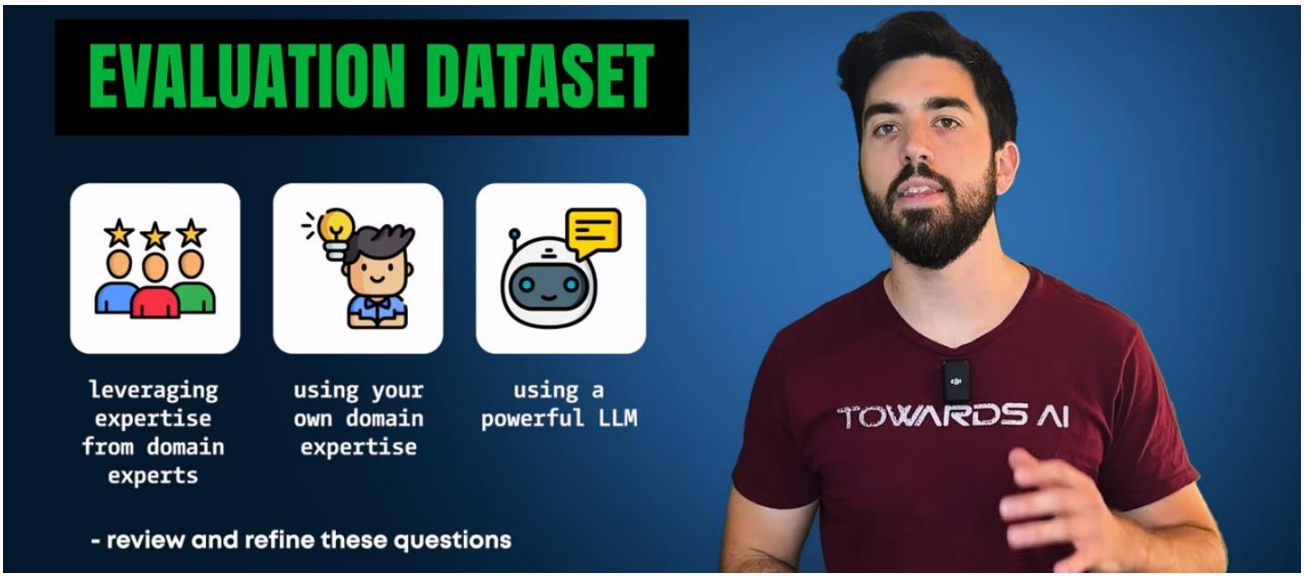
iii. Answer Correctness

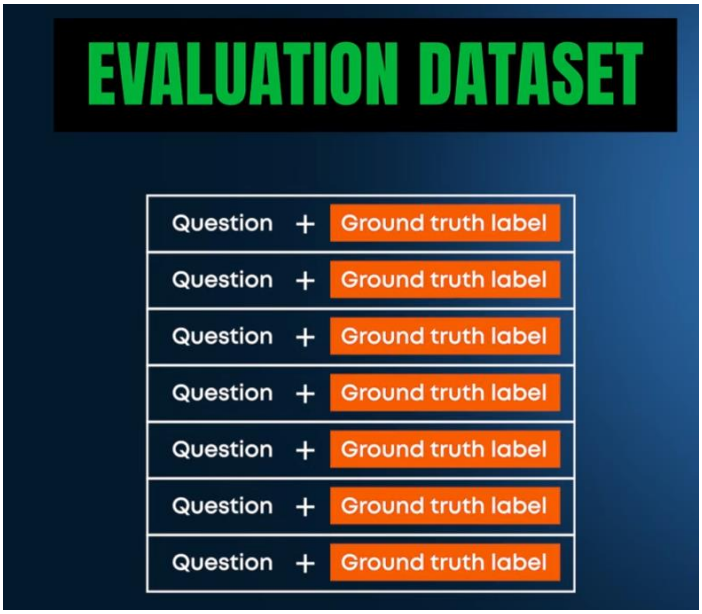
Hint:

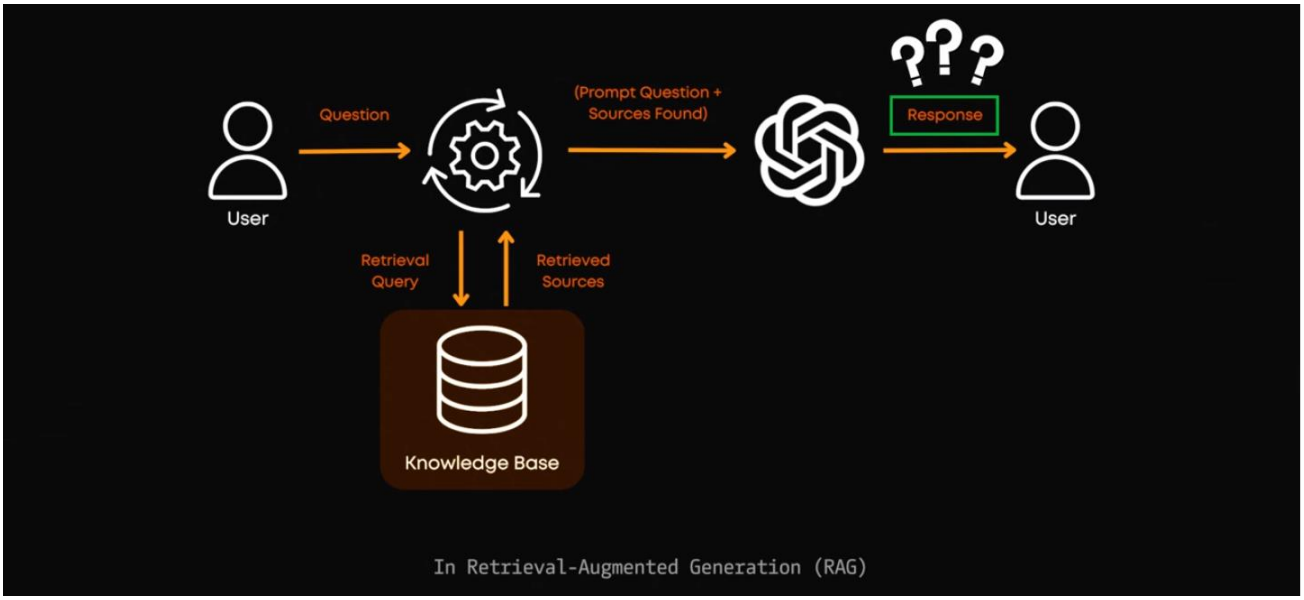
Ground truth: Einstein was born in 1879 in Germany.
High answer correctness: In 1879, Einstein was born in Germany.
Low answer correctness: Einstein was born in Spain in 1879.

- Assess if the answer aligns with a given query's reference answer
- Useful with ground truth answers for comparison

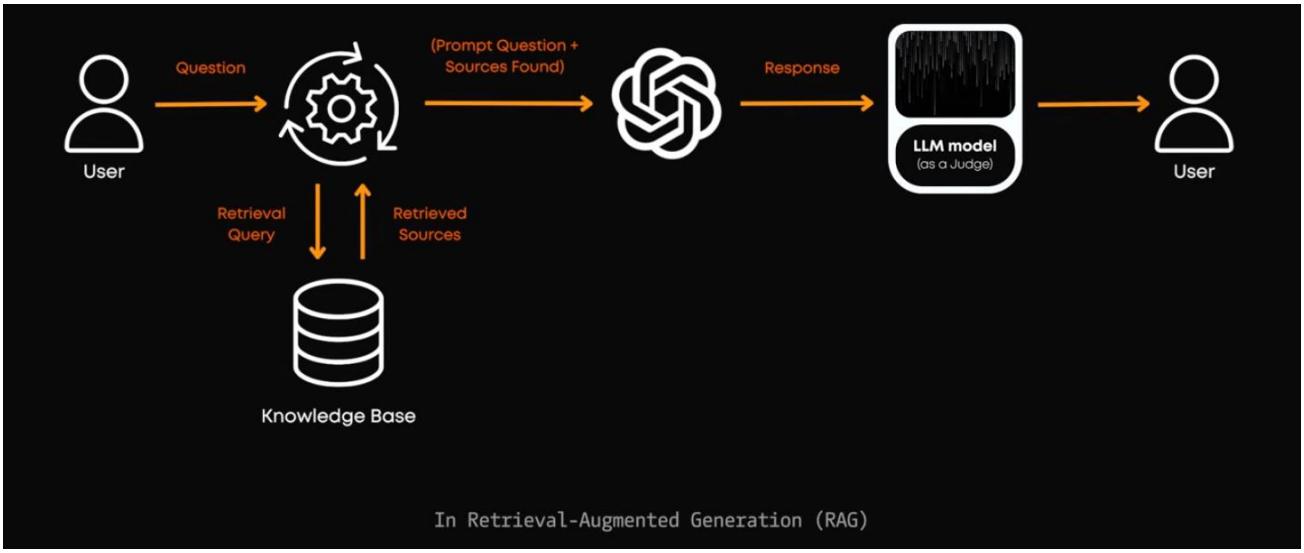


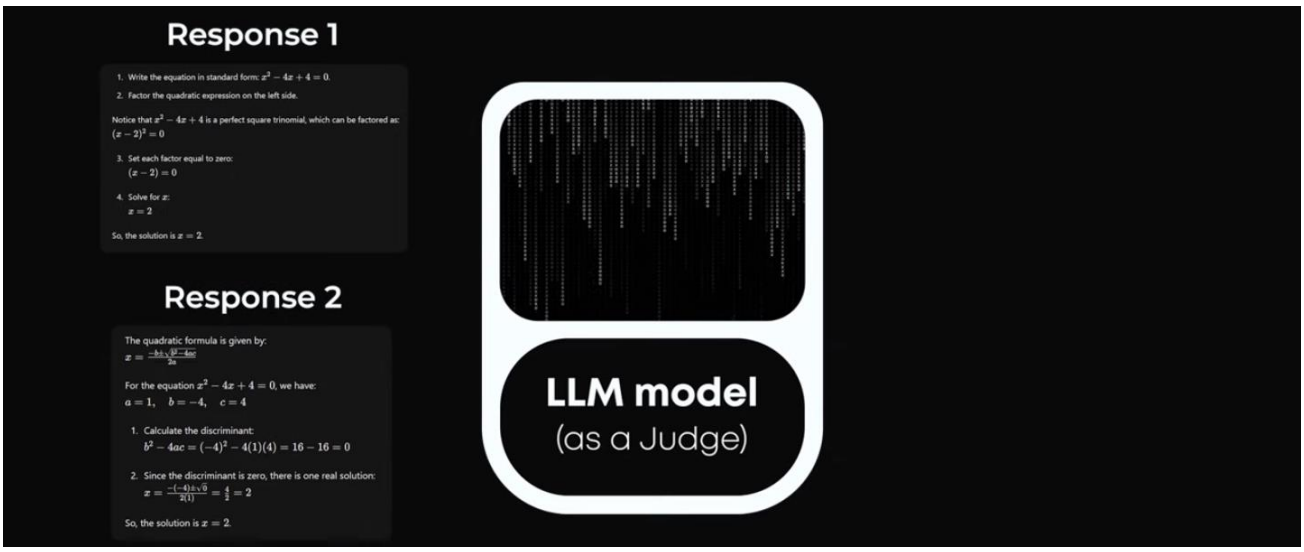




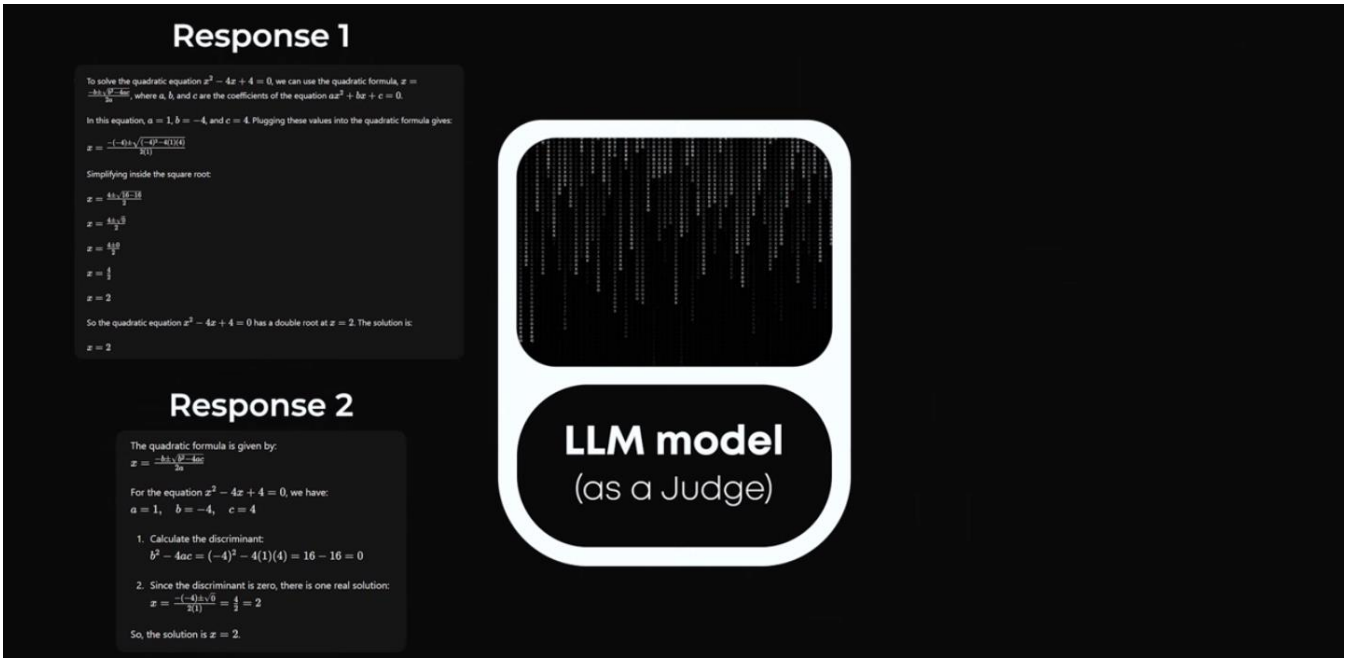


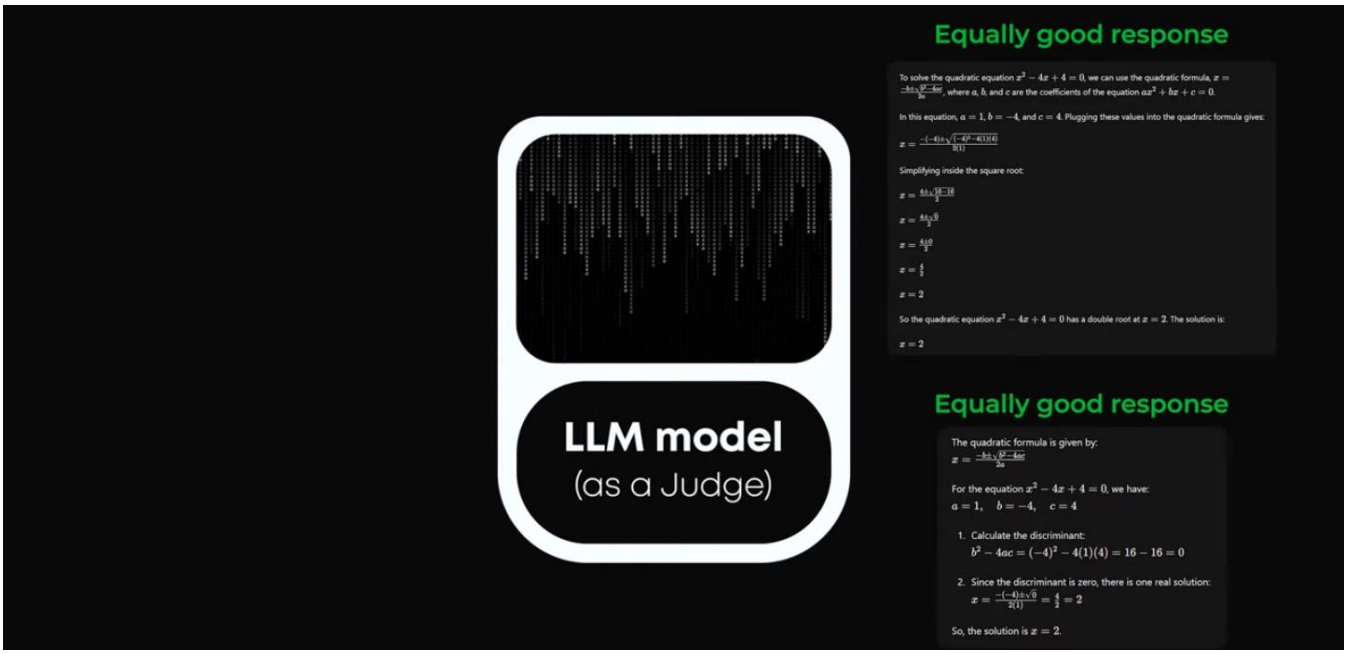
How can we judge if the response generated is good?



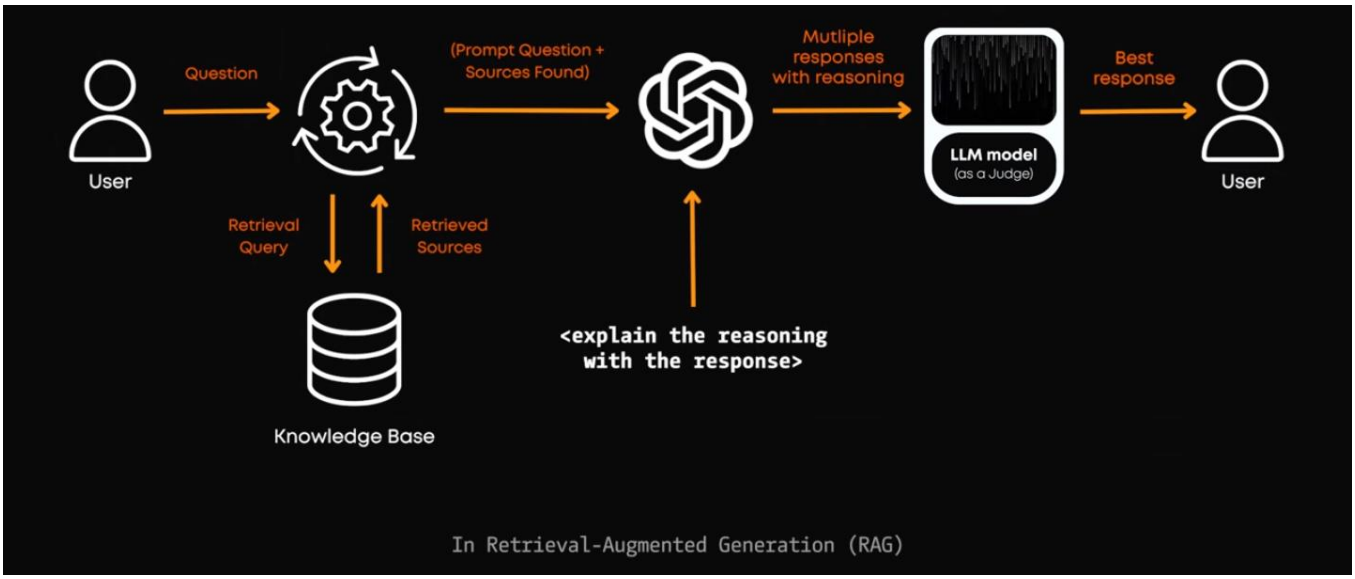


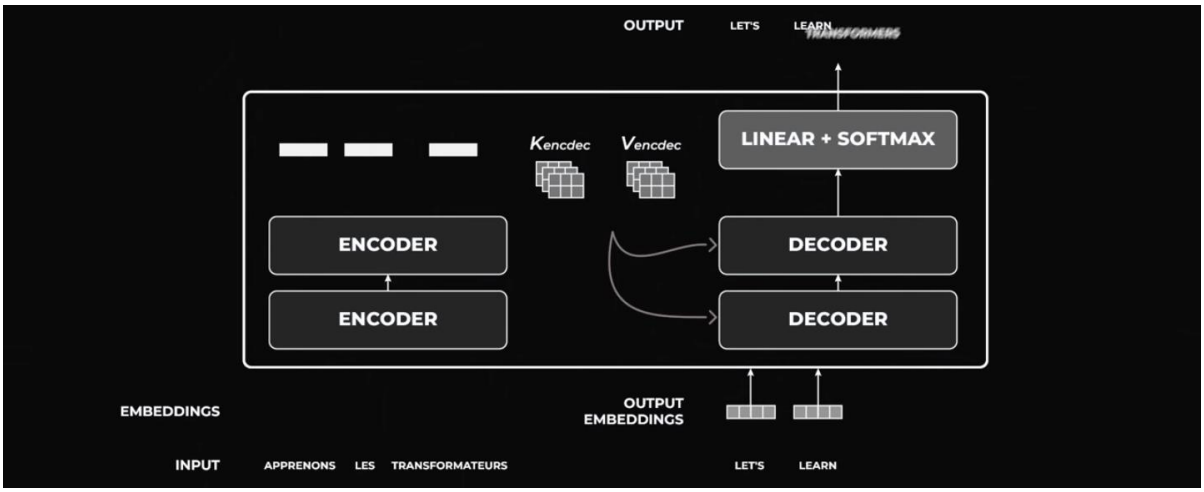
Don’t forget to randomize the order of the 2 responses you give to the LLM judge to reduce bias for the first response

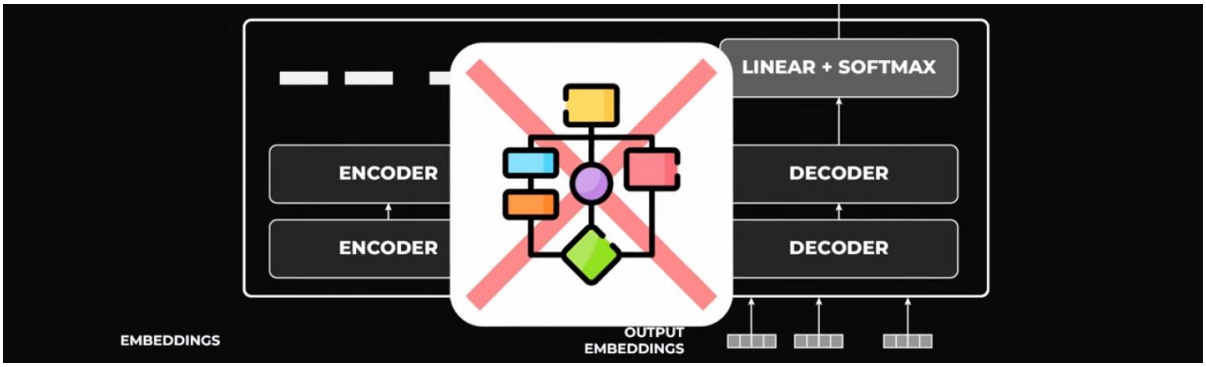




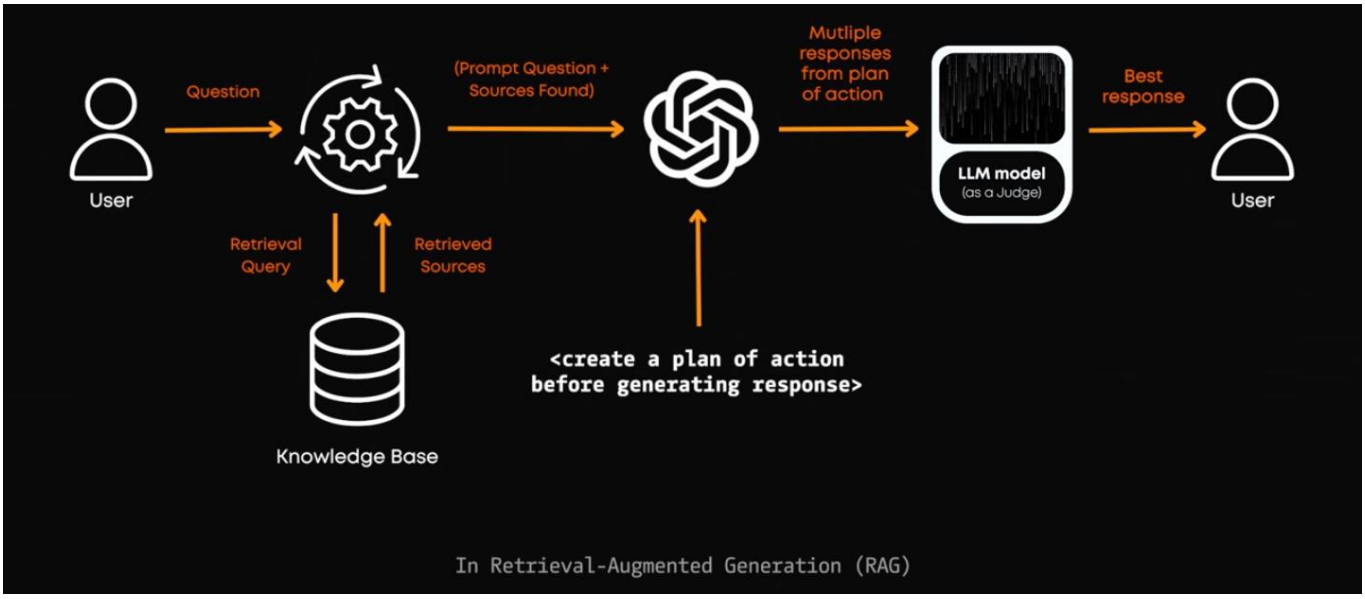
Give the LLM judge the option to declare both options are equally good to make it more nuance

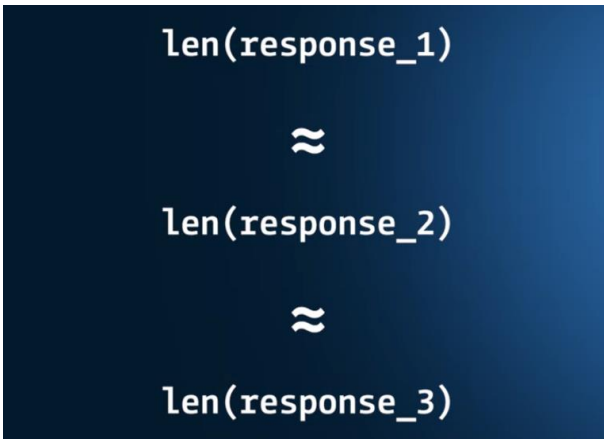




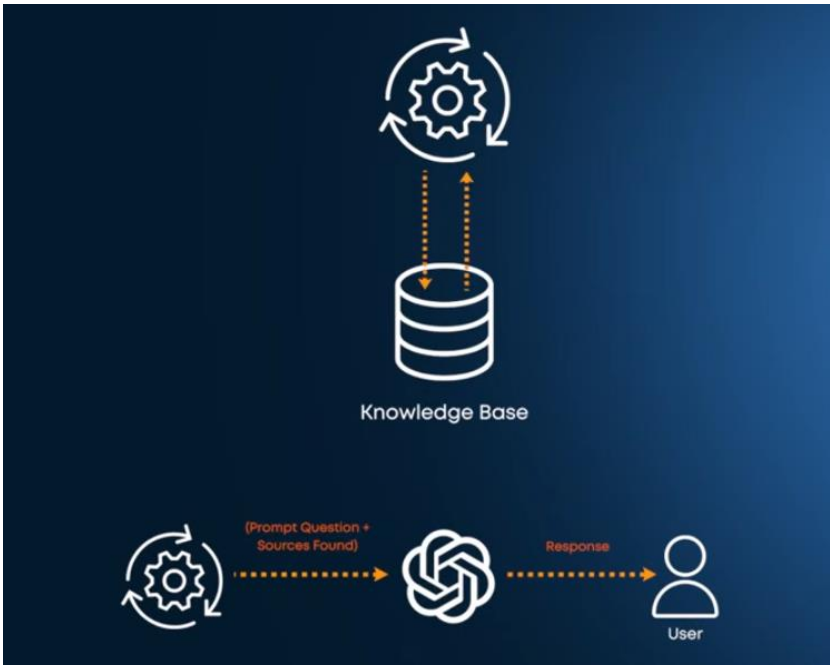


LLMs don’t think like humans, you need to ask it to explain its rational for making the conclusions





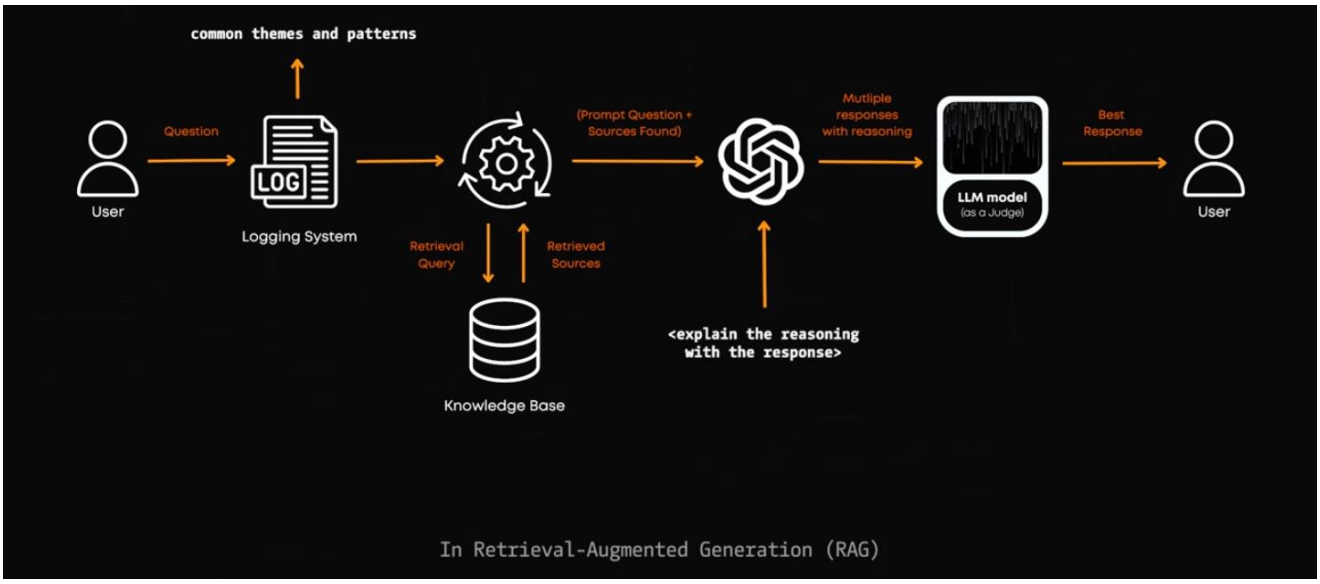
Try to make the responses similar in length when passing them to the LLM judge for comparison


Knowledge Base
(Prompt Question + Sources Found)
Response
User

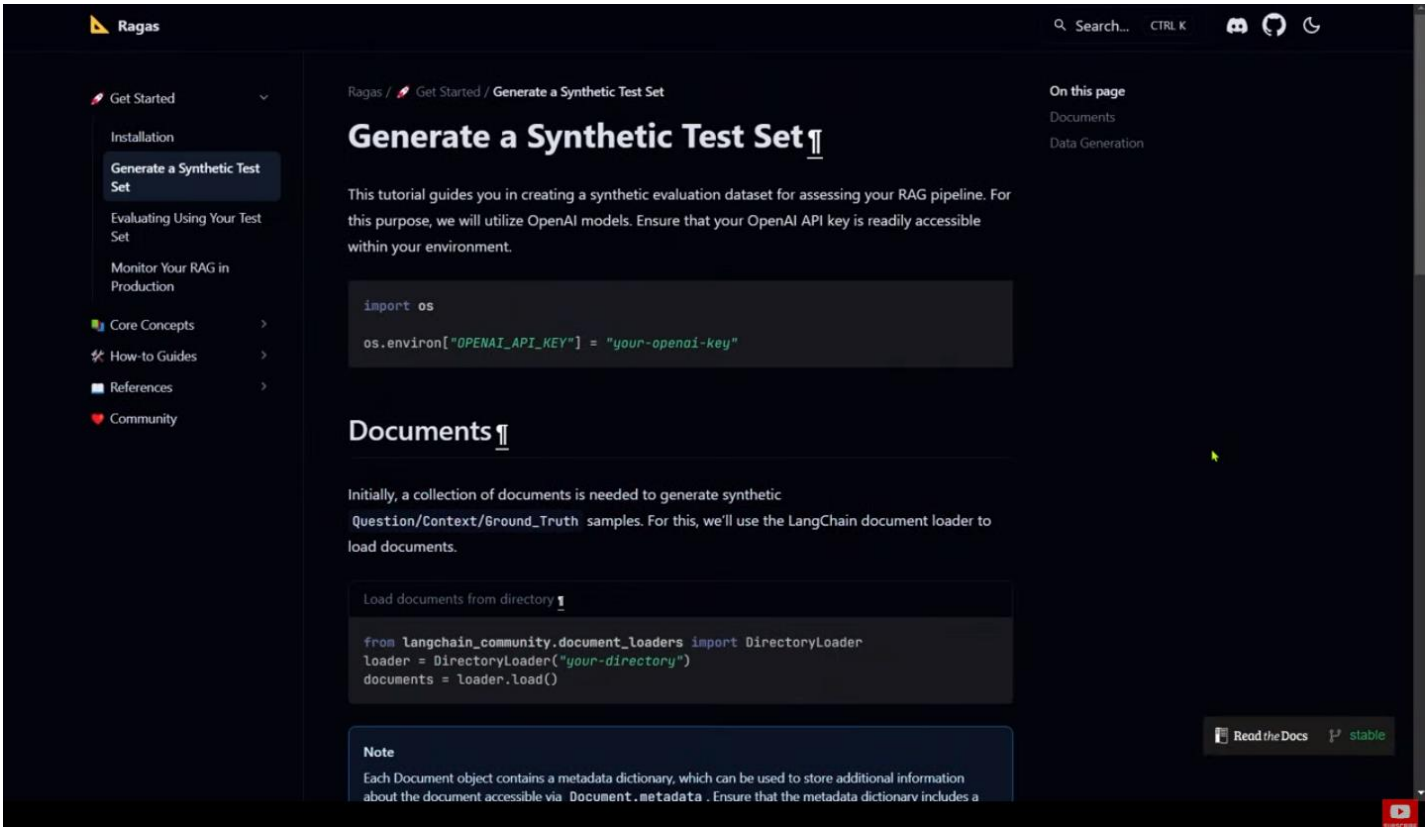

COMPREHENSIVE VIEW OF PERFORMANCE


UNDERSTANDING YOUR USERS' ACTUAL NEEDS


EFFECTIVE WAY TO UNDERSTAND USERS NEEDS
- Logging user queries
- Analyzing user queries


common themes and patterns
Question
User
LOG
Logging System
Retrieval Query
Retrieved Sources
Knowledge Base
(Prompt Question + Sources Found)
<explain the reasoning with the response>
Mutliple responses with reasoning
LLM model
(as a Judge)
Best Response
User
In Retrieval-Augmented Generation (RAG)


IMPLEMENT THE EVALUATION PROCESS


Ragas
Search... CTRL K
Get Started
Installation
Generate a Synthetic Test Set
Evaluating Using Your Test Set
Monitor Your RAG in Production
Core Concepts
How-to Guides
References
Community
Ragas / Get Started / Generate a Synthetic Test Set
Generate a Synthetic Test Set
This tutorial guides you in creating a synthetic evaluation dataset for assessing your RAG pipeline. For this purpose, we will utilize OpenAI models. Ensure that your OpenAI API key is readily accessible within your environment.
import os
os.environ["OPENAI_API_KEY"] = "your-openai-key"
Documents
Initially, a collection of documents is needed to generate synthetic Question/Context/Ground_Truth samples. For this, we'll use the LangChain document loader to load documents.
Load documents from directory
from langchain_community.document_loaders import DirectoryLoader
loader = DirectoryLoader("your-directory")
documents = loader.load()
Note
Each Document object contains a metadata dictionary, which can be used to store additional information about the document accessible via Document.metadata . Ensure that the metadata dictionary includes a
On this page
Documents
Data Generation
Read the Docs
stable

Note

Each Document object contains a metadata dictionary, which can be used to store additional information about the document accessible via `Document.metadata`. Ensure that the metadata dictionary includes a key called `filename`, as it will be utilized in the generation process. The `filename` attribute in metadata is used to identify chunks belonging to the same document. For instance, pages belonging to the same research publication can be identified using the filename.

Here's an example of how to do this:

```
for document in documents:
    document.metadata['filename'] = document.metadata['source']
```

At this point, we have a set of documents ready to be used as a foundation for generating synthetic `Question/Context/Ground_Truth` samples.

Data Generation ¶

Now, we'll import and use Ragas' `TestsetGenerator` to quickly generate a synthetic test set from the loaded documents.

Create 10 samples using default configuration ¶

```
from ragas.testset.generator import TestsetGenerator
from ragas.testset.evolutions import simple, reasoning, multi_context
from langchain_openai import ChatOpenAI, OpenAIEmbeddings

# generator with openai models
generator_llm = ChatOpenAI(model="gpt-3.5-turbo-16k")
critic_llm = ChatOpenAI(model="gpt-4")
```


Evaluating Using Your Test Set ¶

Once your test set is ready (whether you've created your own or used the **synthetic test set generation module**), it's time to evaluate your RAG pipeline. This guide assists you in setting up Ragas as quickly as possible, enabling you to focus on enhancing your Retrieval Augmented Generation pipelines while this library ensures that your modifications are improving the entire pipeline.

This guide utilizes OpenAI for running some metrics, so ensure you have your OpenAI key ready and available in your environment.

```
import os
os.environ["OPENAI_API_KEY"] = "your-openai-key"
```

Note

By default, these metrics use OpenAI's API to compute the score. If you're using this metric, ensure that you've set the environment key `OPENAI_API_KEY` with your API key. You can also try other LLMs for evaluation, check the **Bring your own LLM guide** to learn more.

Let's begin with the data.

The Data ¶

For this tutorial, we'll use an example dataset from one of the baselines we created for the **Amnesty**

Ragas

Search... CTRL K

- Get Started
  - Installation
  - Generate a Synthetic Test Set
  - Evaluating Using Your Test Set
  - Monitor Your RAG in Production
- Core Concepts
- How-to Guides
- References
- Community

See also

See **test set generation** to learn how to generate your own `Question/Context/Ground_Truth` triplets for evaluation. See **preparing your own dataset** to learn how to prepare your own dataset for evaluation.

Metrics ¶

Ragas provides several metrics to evaluate various aspects of your RAG systems:

1. Retriever: Offers `context_precision` and `context_recall` that measure the performance of your retrieval system.
2. Generator (LLM): Provides `faithfulness` that measures hallucinations and `answer_relevancy` that measures how relevant the answers are to the question.

There are numerous other metrics available in Ragas, check the **metrics guide** to learn more.

Now, let's import these metrics and understand more about what they denote.

import metrics ¶

```
from ragas.metrics import (
    answer_relevancy,
    faithfulness,
    context_recall,
    context_precision,
)
```

Here we're using four metrics, but what do they represent?

On this page
- The Data
- Metrics
- Evaluation

Read the Docs stable

Ragas

Search... CTRL K

- Get Started
  - Installation
  - Generate a Synthetic Test Set
  - Evaluating Using Your Test Set
  - Monitor Your RAG in Production
- Core Concepts
- How-to Guides
- References
- Community

Evaluation ¶

Running the evaluation is as simple as calling `evaluate` on the `Dataset` with your chosen metrics.

evaluate using sample dataset ¶

```
from ragas import evaluate

result = evaluate(
    amnesty_qa["eval"],
    metrics=[
        context_precision,
        faithfulness,
        answer_relevancy,
        context_recall,
    ],
)

result
```

There you have it, all the scores you need.

Note

Depending on which LLM provider you're using, you might have to configure the `llm` and `embeddings` parameter in the `evaluate` function. Check the **Bring your own LLM guide** to learn more.

And depending on the provider's, rate_limits, you might want to configure parameters like max_workers, rate_limits, timeouts, etc. Check the **Ragas Configuration** guide to learn more.

If you want to delve deeper into the results and identify examples where your pipeline performed poorly or exceptionally well, you can convert it into a pandas DataFrame and use your standard

On this page
- The Data
- Metrics
- Evaluation

Read the Docs stable

Importance of Human Evaluation

- Human judgment is best for assessing factors like,
  - Fluency
  - Naturalness of responses
  - Usefulness in real-world scenarios
- Users are the best for evaluation, via A/B testing or paying them