



Simple Evaluation of RAG with Ragas



Namrata Tanwani · [Follow](#)

5 min read · Mar 16, 2024



75



Like a toddler an LLM is taught to learn and “understand” a language during its training. But, this training is not perfect as the LLMs are trained up to a certain period of time, hence, their knowledge can prove to be outdated in many situations and they tend to hallucinate or generate gibberish if unfamiliar with a query or concept.



[Tenor GIF Keyboard](#)

To overcome these challenges, we implement RAG. RAG or Retrieval Augmented Generation is like an open book examination but for large language model. The LLM now has a book (knowledge base) to take reference from and generate answers.

The pipeline consists of mainly following steps:

1. Document Loading: Collecting all relevant documents from different sources to create knowledge base.
2. Document Chunking: Splitting large documents into smaller chunks to be able to be passed as context to an LLM.
3. Embedding Generation: These chunked documents are now transformed into a dense array of numbers or embeddings.

4. Vector Storage: These embeddings are now stored in vector storage for the ease of retrieval.
5. Context Retrieval: The embeddings that are relevant to the user query are now passed as context to the LLM.
6. Answer Generation: The LLM generates answers based on the context supplied.

One might need to perform simple prompt engineering to get the best results.

But, the real question is how does one evaluate these RAG pipelines?



A RAG pipeline is evaluated based on ground truths. So, the evaluation dataset must contain user queries, ground truths, results generated, and context supplied. Before diving into code, we shall explore all the required metrics.

We can divide this process into two parts:

[Open in app](#) ↗



 Search

 Write



- **Context Recall:** The metric checks if all the relevant answers to the question are present in the context. For the user query: *“Who discovered the Galapagos Islands and how?”* A context with high recall will answer both the parts of question — *Who* and *How*. The answer to both of these questions is present in the ground truth. Hence, this metric utilizes context and ground truth to determine a score between 0 and 1. 1 being the highest recall.
- **Context Precision:** This metric determines whether context that is closest to the ground truth is given high score. The more relevant is the chunk to the ground truth the higher will be the score. Context Precision is determined by ground truth, context, and user query. Higher the score, ranging from 0 to 1, higher the context precision.
- **Context Entities Recall:** This recall determines whether all entities present in the ground truth are also present in the supplied context. For query: *“In which countries are snow leopards found?”* the ground truth mentions 12 countries. If the context contains all the names of these countries, then it would result in high context entity recall.

2. Generation Evaluation: This part evaluates the answer generated by the LLM.

- **Faithfulness:** The metric outputs a score between 0 and 1, determining the extent to which the generated response relies solely on the provided context. The lower the score, the less trustworthy the generated answer is, the less reliance on the supplied context.
- **Answer Relevance:** It measures how relevant and pertinent is the generated answer to the user query. For query: “*What are the threats to penguin populations?*” an irrelevant answer might focus on location of penguins while a relevant answer would mention the threats to penguins populations.
- **Answer Similarity:** It calculates how semantically similar are the generated answer and the ground truth. In simple terms, how these two are similar conceptually. For query: “*In which countries are snow leopards found?*” the generated answer might mention only a few countries with snow leopards, but it would have a high answer similarity because the answer is conceptually similar to the ground truth. Again, 1 being the highest similarity score and 0 being the lowest.
- **Answer Correctness:** This metric determines how factually correct is the generated output. It utilizes the ground truth and generated answer to determine this score. The higher the better, from 0 to 1. It is not to be confused with faithfulness as an answer can be (factually) correct but can not be faithful if it is not generated using the context.
- **Answer Harmfulness:** This metric simply determines if the output is potentially offensive to an individual, group, or a society. The output is binary, 0 or 1.

We can now dive into the code to perform RAG and determine these metrics using Ragas (RAG Assessment) framework. Ragas is a simple framework that helps evaluate RAG pipelines.

We start with importing libraries.

Then, we read the OpenAI's API key as we'll be utilizing GPT-3.5. We also create our knowledge base using the data scraped from four URLs.

These documents need to be transformed to string for preprocessing and ultimately, arranged as a list of string rather than list of documents.

We now perform semantic chunking which simply creates chunks of documents based on semantic similarity between sentences. Two sentences will fall in the same chunk if they are semantically similar. To find semantically similar data, we require dense embeddings of sentences, therefore, we use sentence transformers *all-MiniLM-L6-v2* model to generate these embeddings.

Further, these chunked documents are stored as in Chroma's vector store in a folder called *chroma_db*. We utilize the same sentence transformers model to store these documents as embeddings.

Moving on, we define the prompt template for the LLM. We use GPT's model 3.5 with 16k context length. We make sure that the result also returns all the contexts/ source documents utilized.

We define our queries and ground truths for evaluation.

Finally, we generate results and store context supplied to the LLM.

The evaluation dataset is prepared as a dictionary and the evaluation metrics are calculated for each query using Ragas. These scores are further stored as a csv file.

We can also see the mean of all metrics that gives us the score for entire dataset.

faithfulness	0.955000
answer_relevancy	0.923192
context_precision	0.733333
context_recall	0.916667
context_entity_recall	0.322197
answer_similarity	0.941792
answer_correctness	0.665889
harmfulness	0.000000
dtype:	float64

The RAG pipeline exhibits strong performance in terms of faithfulness, answer relevancy, context recall, and answer similarity. However, there are areas for improvement, particularly in context entity recall, context precision, answer correctness, to enhance the overall quality and accuracy of the responses generated by the model.

You can also find this code, chrom_db folder, and evaluation metrics scores in my github repository: <https://github.com/namratanwani/Evaluate-RAG/blob/master/RAG-evaluation.ipynb>

References

[Introduction | Ragas](#)

[Introduction | !\[\]\(9dfdaff1d86ba3c1f8353b4d1b61b8c5_img.jpg\) Langchain](#)

Hope you found this helpful! Please feel free to share your feedback :)

Tenor GIF Keyboard

Rag

Llm

Gpt

Evaluation

AI



Written by Namrata Tanwani

Follow



1.6K Followers

ML. Living to learn, learning to live. Let's connect:
<https://www.linkedin.com/in/namratanwani>

More from Namrata Tanwani



 Namrata Tanwani in Analytics Vidhya

ANACONDA 101

So anyone here who is beginning with Anaconda that too on Windows, You have go...

3 min read · Jan 7, 2021



173



 Harikrishnan N B in Analytics Vidhya

Confusion Matrix, Accuracy, Precision, Recall, F1 Score

Binary Classification Metric

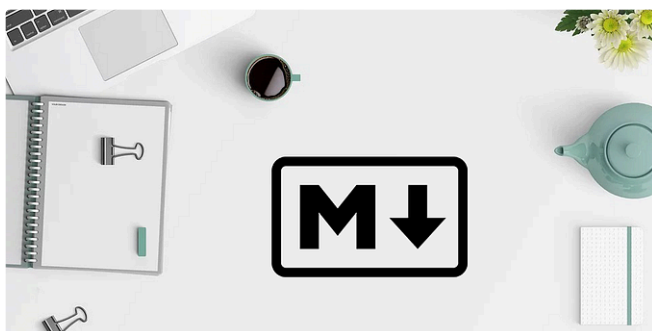
6 min read · Dec 10, 2019




925



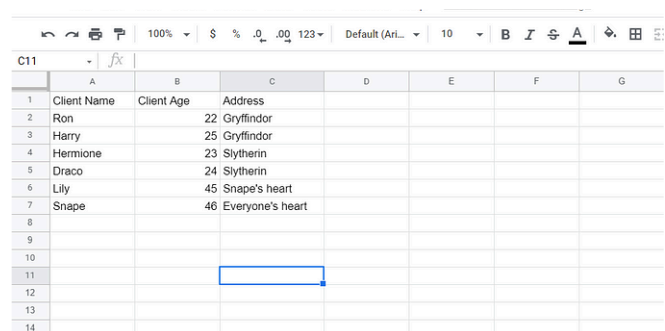
6




 Hannan Satopay in Analytics Vidhya

The Ultimate Markdown Guide (for Jupyter Notebook)

An in-depth guide for Markdown syntax usage for Jupyter Notebook



	A	B	C	D	E	F	G
1	Client Name	Client Age	Address				
2	Ron	22	Gryffindor				
3	Harry	25	Gryffindor				
4	Hermione	23	Slytherin				
5	Draco	24	Slytherin				
6	Lily	45	Snape's heart				
7	Snape	46	Everyone's heart				
8							
9							
10							
11							
12							
13							
14							

 Namrata Tanwani in Analytics Vidhya

Changing Ownership of your Google Data Studio Visualization

Imagine you're working as an intern in a company, and you use your personal account...

10 min read · Nov 18, 2019

🌟 · 3 min read · Feb 13, 2021

👏 2.2K 💬 13

🔖+ ⋮

👏 481 💬 1

🔖+ ⋮

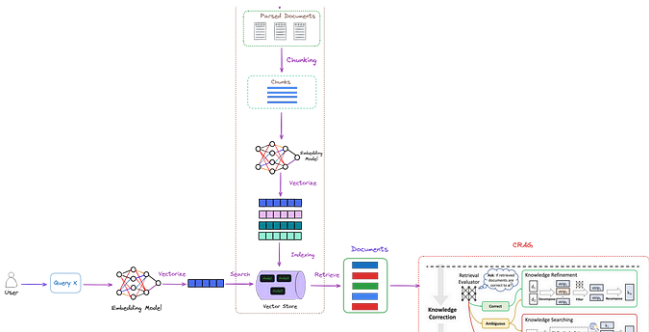
See all from Namrata Tanwani

Recommended from Medium

$$\text{Precision@k} = \frac{\sum \text{precision@k}}{\text{total number of relevant items in the dataset}}$$

$$\text{Precision@k} = \frac{\text{true positives@k}}{(\text{true positives@k} + \text{false positives@k})}$$

number of chunks in contexts



 Rupesh Yadav


Evaluation of RAG (Retrieval-Augmented Generation) using...

You've built a Retrieval-Augmented Generation (RAG) application, showcasing it...

14 min read · Feb 8, 2024

👏 57 💬

🔖+ ⋮

 Florian June in AI Advances

Advanced RAG 10: Corrective Retrieval Augmented Generation...

Intuitive Example, Priciples, Code Explanation and Insights about CRAG

🌟 · 11 min read · 5 days ago

👏 358 💬 1

🔖+ ⋮

Lists



The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 360 saves



Natural Language Processing

1390 stories · 886 saves



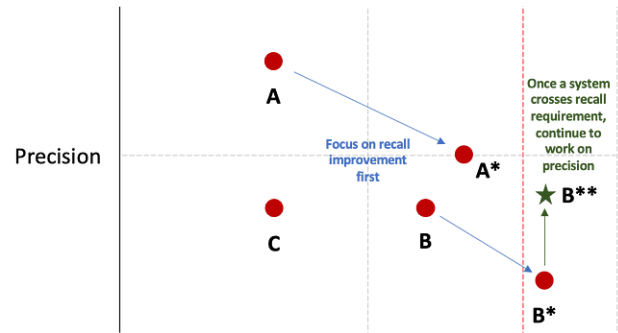
Generative AI Recommended Reading

52 stories · 948 saves



What is ChatGPT?

9 stories · 338 saves



Yi Zhang in Relari Blog

A Practical Guide to RAG Pipeline Evaluation (part 1)

Series of blogs using on how to evaluate and improve your LLM and RAG Pipelines. Deep...

11 min read · Dec 11, 2023

134 2

Repository Name	About	Stars	Forks	Contributors	Issues	Releases	Watchers	Time Since Last Commit	License	Languages
sglance	Transformers: State-of-the-art Machine Learning for Pytorch, TensorFlow, and JAX.	121,891	24,181	434	1,032	141	1,086	0 days, 8 hrs, 4 mins	Apache License 2.0	Python, Cuda, Shell, C++, Dockerfile, C, Makefile, Cython, Jupyter
PTNextW	ChatGPT-Next-Web: A cross-platform ChatGPT/Gemini UI (Web / PWA / Linux / Win / MacOS) - 一键部署自己的知识库 ChatGPT/Gemini UI.	64,002	52,899	167	223	57	362	0 days, 11 hrs, 13 mins	MIT License	TypeScript, SCSS, JavaScript, Dockerfile, Rust
all	gpt4all: run open-source LLMs anywhere	62,317	6,832	88	364	19	616	0 days, 9 hrs, 34 mins	MIT License	C++, GML, Python, CMake, J, C#, JavaScript, C, Go, Makefile, Shell, C, Swift, PowerShell, Bashfile, CSS
anov	Llama.cpp	52,671	7,411	479	1,229	1,544	493	0 days, 9 hrs, 47 mins	MIT License	C++, C, Cuda, Python, Metal, Objective-C, Shell, CMake, Makefile, Nix, Dockerfile, Zig, Bashfile
tes	privateGPT	48,358	6,351	62	142	6	432	0 days, 9 hrs, 7 mins	Apache License 2.0	Python, MDX, Makefile
i	ollama	46,681	3,114	146	656	51	295	0 days, 9 hrs, 51 mins	MIT License	Go, Shell, C, TypeScript, PowerShell, C++, Dockerfile, I Setup, Python, CMake, CSS, Objective-C, JavaScript, HTML
ooga	next-generation-webui	34,130	4,564	299	276	34	297	0 days, 14 hrs, 8 mins	GNU Affero General Public License v3.0	Python, CSS, JavaScript, Shell, Bashfile, Jupyter Notebook, Dockerfile
nwingley	chatbot-ui	25,201	6,829	31	62	0	231	0 days, 22 hrs, 47 mins	MIT License	TypeScript, PostgreSQL, JavaSC, CSS, Shell
ib	lobe-chat	21,931	4,502	69	208	480	118	0 days, 8 hrs, 23 mins	MIT License	TypeScript, JavaScript, Dockerfile
	llm-chat									

Vince Lam in The Deep Hub

Cobus Greyling

LLamaIndex Agentic RAG Demo

Agentic RAG is an agent based approach to perform question answering over multiple...

6 min read · Feb 1, 2024

248 2



Tejaswi kashyap

50+ Open-Source Options for Running LLMs Locally

In my previous post I discussed the benefits of using locally hosted open weights LLMs,...

8 min read · Mar 12, 2024

 459  6  

RAG processing using Llamaindex

Query over your PDF's using llama index

6 min read · Mar 15, 2024

 10  1  

See more recommendations