High-Performance RAG

Llamaindex edition!!

* taken from Ai Makerspace presentation by Dr Greg Loughnane & Chris Alexiuk

WHY RAG?

Hallucinations lead to confident BUT FALSE responses.
Better to have LLMs fact check against your documents.
(especially in business context)

Retrieval - find reference in docs
Augmented - add reference to prompts
Generation - improves answer to questions

Helpful in specialized domains → legal, health, finance,
(with lots of jargons) gov, research, insurance

RAG ≡ Question Answering over docs

broken into 2 pieces

Retrieval (using vectors)

Augmenting prompt
(collected info into prompt)

within LLM context window

RETRIEVAL (major emphasis in Llamaindex)

3 pieces →
- Ask question
- Search database for stuff similar to question
- Return stuff

Database used is usually VECTOR database — can also be called as INDEX

BUILDING INDEX

1. Split the docs into chunks
2. Create embeddings for each chunk — using some embedding model
3. Store these embeddings in vector store index

RETRIEVERS

1. Ask a question
2. Convert it into a vector
3. Look for similar vectors in vector store index — also called as retrieved context
4. Return retrieved context to put into prompt

DENSE VECTOR RETRIEVAL

Question Database Similarity → Vector → Natural Language

But stuff returned

RAG OVERVIEW

Query → Embedding Model → Similar vector in vector store

dense vector retrieval

↓

Pass to LLM ← Set up prompt ← Rank order results

(in context learning)

LLAMAINDEX OVERVIEW

Tool to build VERY POWERFUL index and doing retrieval

"A data framework for LLM apps to ingest, structure & access private or domain specific data"

(Big companies → sizeable data → diff types of docs)

LLAMAINDEX TERMINOLOGY

Sentence-level information → Nodes

PDF-level information → Documents

NODE

(first class citizen in llamaindex)

- → chunk of (source) doc
- → inherit metadata
- → can track where they came from
- → nodes are what get stored in vector store
- → Asking query returns "k" most similar nodes
- → these nodes are passed through Response synthesis

↓ to create nodes

NODE PARSERS

list of docs ⟶ chunks into node objects
(different chunking strategies)

QUERY ENGINE

(magic of llamaindex)

- → Generic interface allowing Question Answering over your own data
- → as IMPORTANT as "chain" in LangChain

DOMAIN SPECIFIC EXAMPLE - Camelids family (vetrinary domain)

1. Data : research paper
2. Index building : load data, chunk nodes, create embeddings, store nodes in index

IMPROVING RETRIEVAL

Simple to Advanced RAG → Check previous post

Table stakes ⟷ Adv Retrieval ⟷ Fine-tuning

1. Fine-tuning Embeddings

Reason → Specialized vocabulary in vetrinary research context
→ Handle mumbo-jumbo in docs

Process

1. Need training-validation-testing sets of (Question, Retrieved context) pairs

2. Loss function → HF Sentence Transformers loss (built-in)
Takes positive pairs and automatically augments dataset with negative pairs (Question, wrong context)

3. Embedding Model → bge-small-en
Cost efficient OpenSource model from BAAI

* CODE EXPLANATIONS WILL BE COVERED IN NEXT POST *

SIMPLE-TO-BIG RETRIEVAL

We also look around the area of context returned

Sentence Window Node Parser

Splits docs into nodes → each node = sentence
Window around each node → sentences either side of node
(look for relevant context)

EVALUATING RETRIEVAL

Relevant context is MURKY TERM. Thus we utilize some libraries to evaluate returned context
(built-in Llamaindex)

RAG Evaluation →
- Correctness
- Semantic Similarity

} Generation

- Faithfulness
- Context Relevancy

} Retrieval

Generation Side

created using GPT-4

① Correctness

Comparing generated response to reference answer

② Semantic similarity

Generated response is SIMILAR to reference

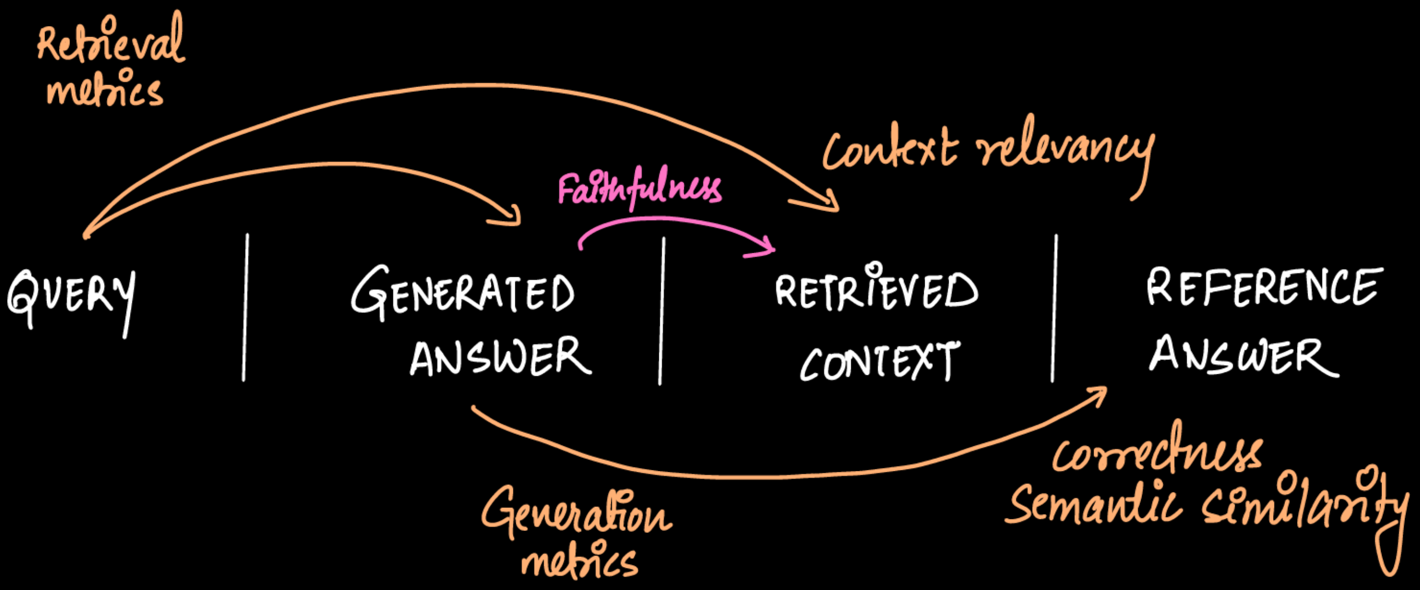
Retrieval Side

③ Faithfulness

Are we seeing lots of hallucinations? Are our answers relevant to our context or not?

④ Context Relevancy

Are retrieved context AND answer relevant to query?



NOTE

→ When you analyze retrieval, then downstream during generation these generations also improve.

→ No matter what metric you use, the results should be better compared to base RAG

OBSERVATIONS

→ Fine-tuning embedd + small-to-big retrieval improves RAG performance across all evaluation metrics

→ Biggest gain are in $\underbrace{\text{Faithfulness}}_{}$ & Relevancy

massive decrease in hallucinations

→ With more docs, RAG system seems to get mastery over domain and also its mumbo-jumbo

→ Complex docs are next steps in more advanced retrieval methods

CONCLUSION

→ Many ways to enhance retrieval. Fine-tuning embeddings is really recommended for specialized vocabulary.

→ Generation metrics also improve with Retrieval

→ Evaluation of RAG systems is easier than ever GPT-4 still used as Ground Truth

(it is really good at self assessment)