
Lizzy AI
RAG
The next generation
of
Contract AI

Introduction

What is RAG ?

- Retrieval-Augmented Generation (RAG) is a technique that enhances the capabilities of Large Language Models (LLMs) by incorporating external knowledge sources or custom.
- This is done by retrieving data/documents relevant to a question or task and providing them as context for the LLM.
- RAG’s are domain specific QA engines

Why RAG ?

- **LLM models do not know *your data***
  - LLMs, powered by vast datasets and billions of parameters, and excel at tasks like question answering. However, their knowledge is confined to publicly available data.
  - Data Can be outdated
- **Providing domain-specific, relevant responses**
  - For LLMs to give relevant and specific responses, organizations need the model to understand their domain and provide answers from their data vs. giving broad and generalized responses.

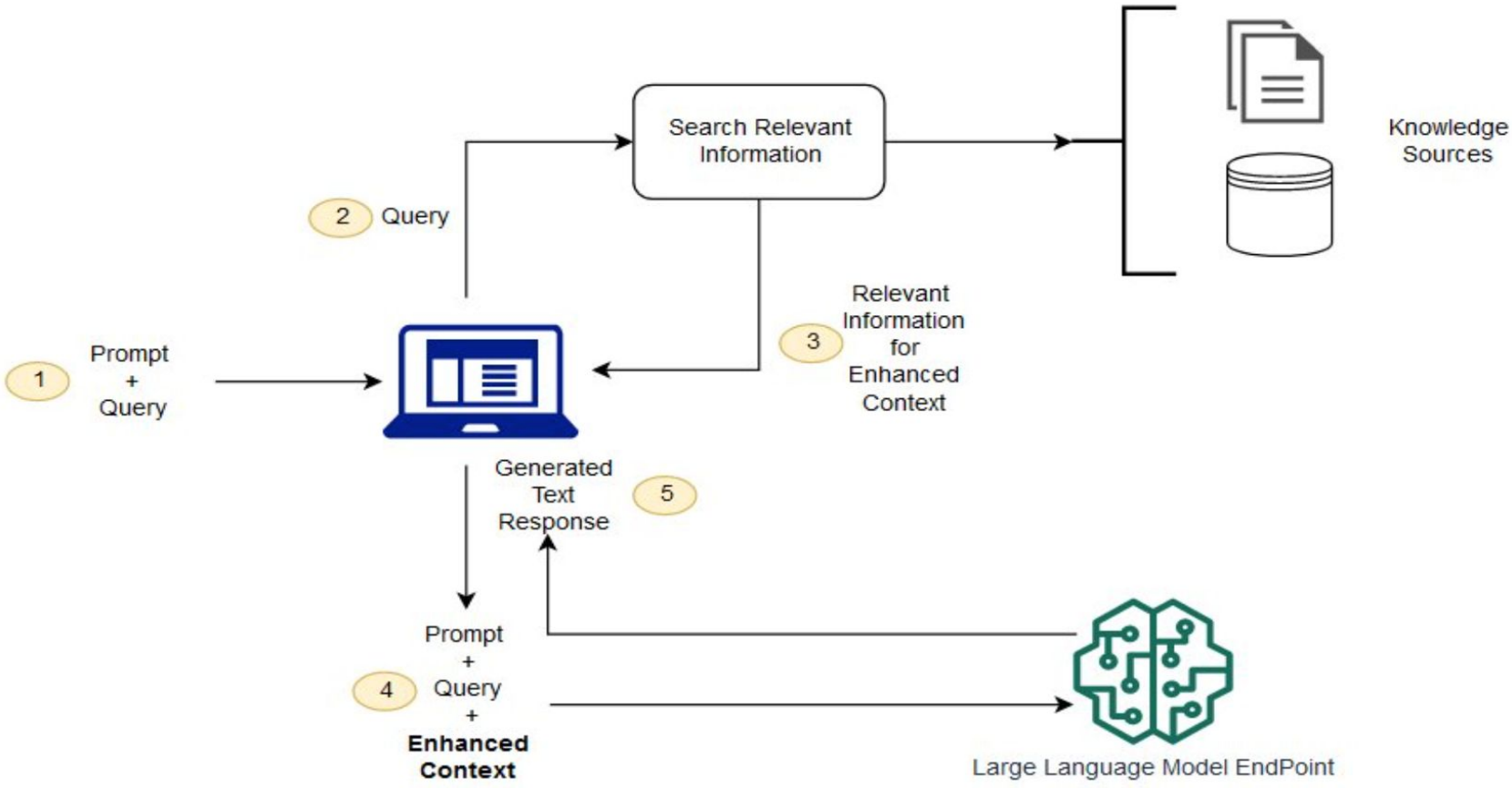
Why RAG ?

- **Reducing inaccurate responses, or hallucinations**:
  - By providing only relevant data, and focusing on domain knowledge, RAG attempts to mitigate the risk of responding with incorrect or fabricated information (also known as hallucinations).
- **Cost Effective**
  - If you need to customize LLM model to know your domain specific data, using techniques like SFT (supervised fine tuning), it is going to be very expensive (Needs thousands of examples, Infrastructure, time etc..). Where as Rag is a simple architectural approach that can solve the issue of domain knowledge

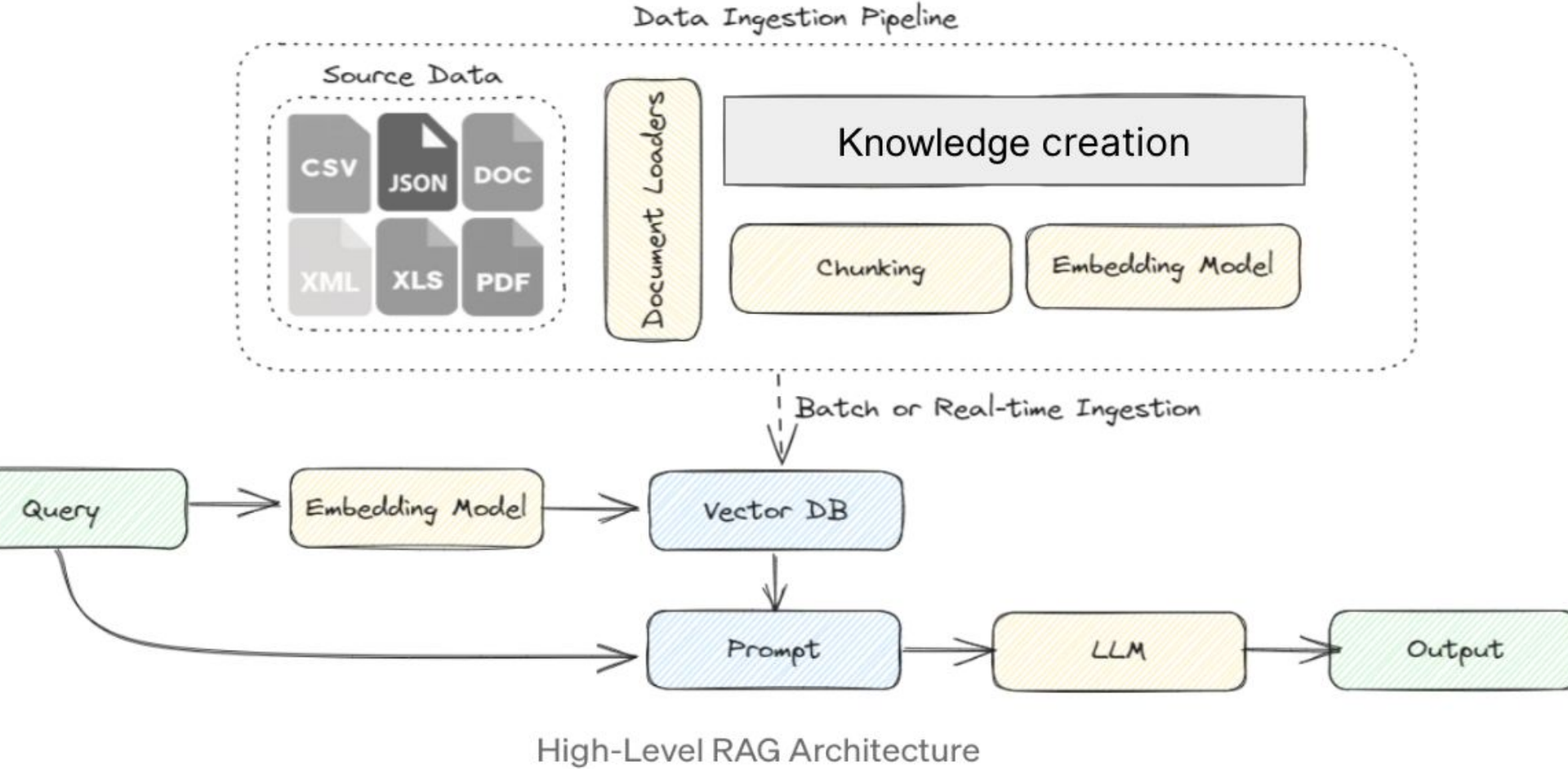
RAG Architecture

Lizzy AI

RAG process (Retrieval + Generation)



RAG architecture



High-Level RAG Architecture

RAG Components

Lizzy AI

1. Data - Chunking

- **Chunking**
  - The first step to do to create you knowledge source is to break down your data into list of chunks
  - LLMS has a limited context window
    - Limit in token
    - We can only provide limited number of tokens data to LLM → can't provide whole data
  - Noise Reduction
    - We want to send only relevant documents to LLM as a context

1. Data - Chunk Size

- **small chunk sizes**
  - Small chunk size leads to an accurate retrieval - Retrieving most similar chunks to your query
  - But it might not provide the full context to the LLM, as they are very short
  - Ex, sentences
- **long chunk chunk sizes**
  - They provide much better context to the LLM
  - But, the retrieval step that is based on similarity search might not be accurate
  - They may contain noises, unwanted information for the LLM

1. Data - Chunk strategies

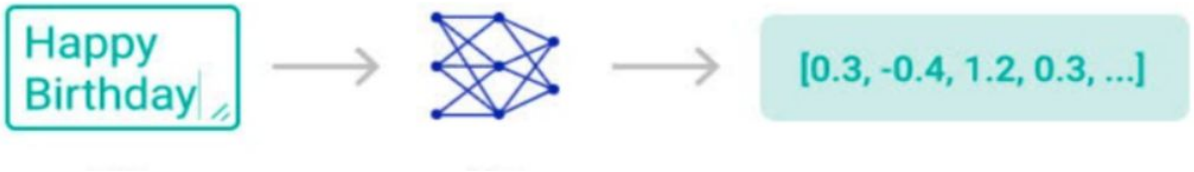
- **Navie chunking**
  - Given a text document, it just simply tries to break down the document into Into chunks of a specified number of characters,
  - It does not consider the underlying structure of the document
  - Simple
- **Recursive**
  - In this method, we divide the text into smaller chunk in a hierarchical and iterative manner using a set of separators. If he initial attempt at splitting the text doesn't produce chunks of the desired size, the method recursively calls itself on the resulting chunks with a different separator
  - Paragraph → Sentence → Word

1. Data - Chunk strategies

- **Semantic**
  - This chunking method aims to extract semantic meaning from embeddings and then assess the semantic relationship between these chunks. The idea is to keep together chunks that are semantic similar.
  - It is content aware, chunks that are talking about similar topic will be in a single group
  - So given a query, the retrieved document will contain most relevant chunk
- **Structural**
  - HTMLBasedChunking, MarkDownChunking etc...

2. Vector Stores

- The power of RAG comes in from its ability to retertive the most relevant documents to answer a given question
- Vector store store and manage high-dimensional vector data, typically derived from complex data structures like text into mathematical embedding vector
- Embedding vector are numerical representation of a the text. But they are not jus random numbers, they hold the semantic information of the text



2. Vector Store Things to consider

- **Query Speed (Indexing)**
    - In Terms of Query speed, indexing play a major role. Indexing is a way to organize or startcuture or vector data in the vector store, so that searching similar documents can be efficient and fast
    - There are different methods of indexing with their own pros and cons
    - What type of indexing does a vector store supports ?
- **Accurate Retrieval**
    - The type of similarity method the vector store supports
- **Embedding Model** - You need to consider the stability of the embedding model, and it's performance
- **Flexibility -** Do your application requires frequent update of the knowledge base or is it static ?
- **Cloud integration:** AWS opeansearch, Google Vertext

Query optimization

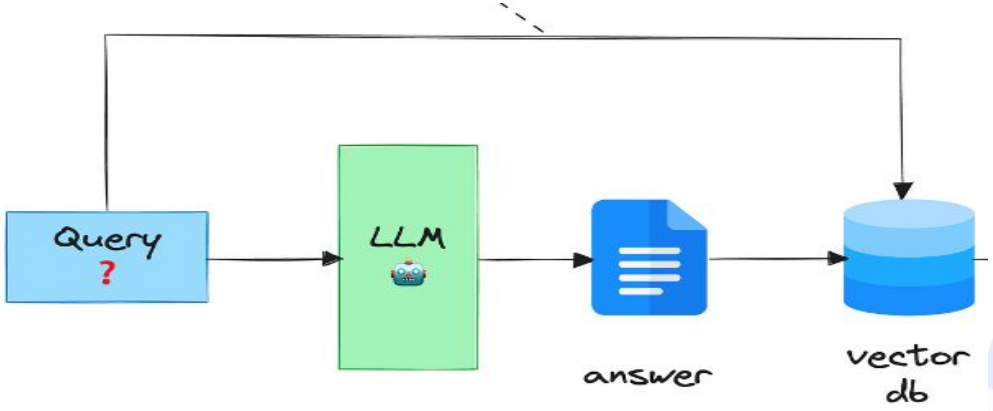
Lizzy AI

3. LLM (Query Prompt)

- **Generating Answer**
  - The relevancy and accuracy of the generated answer depends not only on the retrieved informations given but also by the LLM and the query prompt
  - We don’t have control over the LLM, but we can optimize our query so that we get a more accurate answers
  - In order to optimize our query, there are different approaches

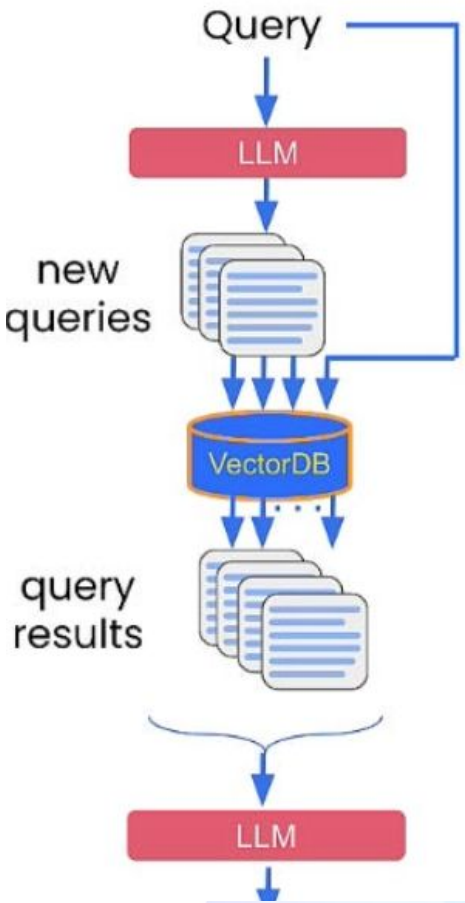
3. LLM (Query Prompt)

- **1. Query Expansion with Hypothetical Answer**
  - One of the reasons our retrieval is not retrieving relevant documents could be due to the fact that it's only considering the question but not the expected answer.
  - Given an input query, this method first instructs an LLM to provide a hypothetical answer, whatever its correctness. Then, the query and the generated answer are combined in a prompt and sent to the retrieval system.



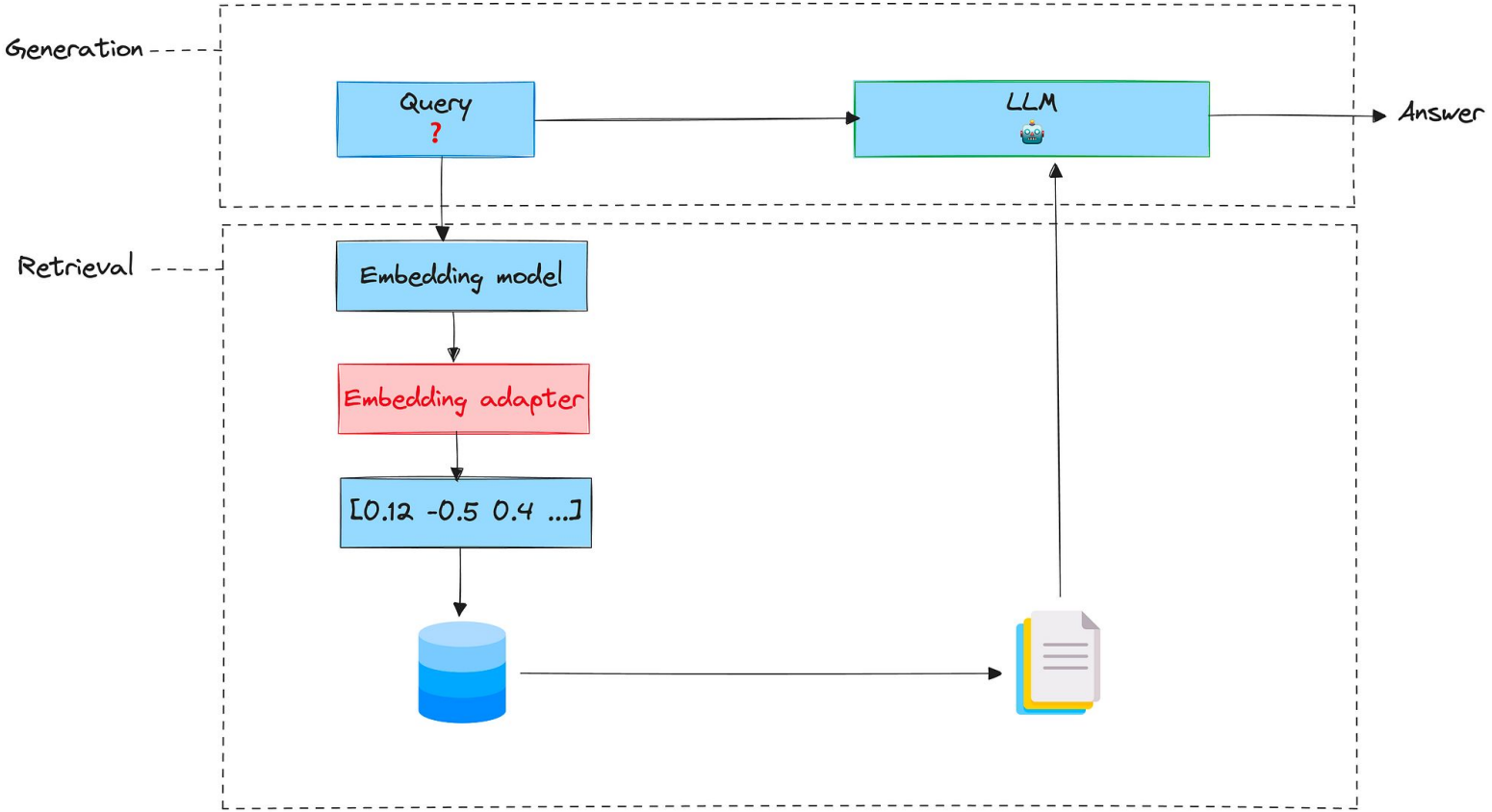
3. LLM (Query Prompt)

- **2. Query Expansion with Multiple queries**
  - In this approach, Given a query , we ask LLM to generate list of N queries that are related to the original query
  - The goal is here is to retertive different topics that are in important for answering our question
  - Note that, we have to make sure the additional queries generated are not a rephrasing of the original query. Rather they must be asking different question that is related to the original question

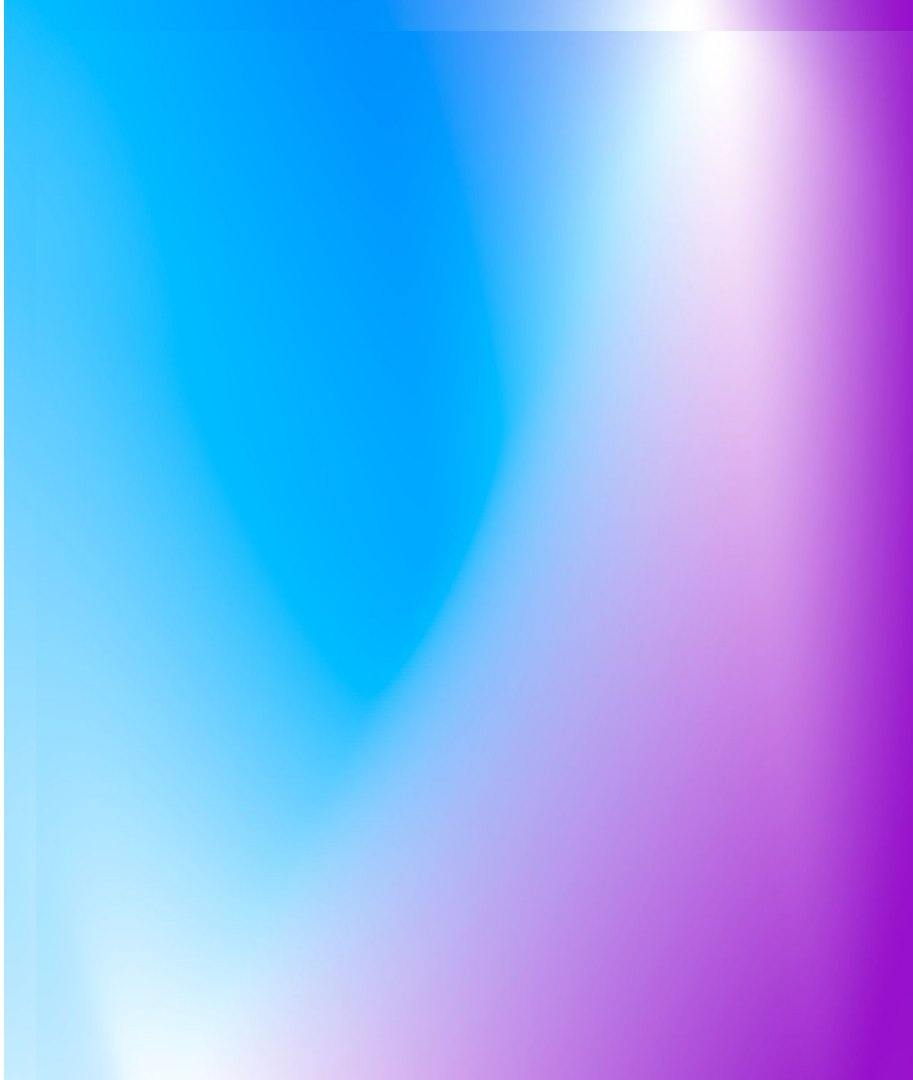


3. LLM (Query Prompt)

- **3. Embedding Adaptors**
    - Another approach to optimize our query is to add Embedding Adaptors after the embedding
    - An embedding adapter is used to modify the embeddings of queries in order to improve the relevance of retrieved results. It acts as an additional stage in the retrieval process, inserted after the initial embedding model and before the retrieval of relevant results.
    - The adaptor is built by training a model that can score the relevancy of a document to a given query. Then weight matrix of that model is used as an adaptor


Generation
Query
?
LLM
Answer
Retrieval
Embedding model
Embedding adapter
[0.12 -0.5 0.4 ...]

RAG evaluation



Lizzy AI

3. RAG evaluation

- RAG evaluation is to measure how well our RAG is doing
- There can be two sources of error
    - Retrieval Evaluation: To assess the accuracy and relevance of the documents that were retrieved
    - Response Evaluation: Measure the correctness of the response generated by the system when the context was provided
- It is recommended to have a mechanism to separately measure our retrieval and generation components

Retrieval Evaluation: To assess the accuracy and relevance of the documents that were retrieved
Response Evaluation: Measure the appropriateness of the response generated by the system when the context was provided

3. RAG evaluation challenges

- Using Deterministic metrics
  - Deterministic metrics like cosine similarity, or metrics that depends on word overlap or similarity does not necessarily evaluate the correctness of an answer
  - A very good answer might have different phrasing from that of the ground truth answer
  - QA evaluation is something like human judgment
- Using LLM to directly give score
  - Recent studies shows LLM are biased for scoring a QA answer
  - They prefer long responses rather than short ones
  - If you instruct an LLM to select a score from a range of score, LLMs tend to be biased to certain value

3. RAG evaluation challenges

- **Using RAGAS**
  - Ragas is a framework that helps you evaluate your QA pipelines across these different aspects. It provides you with a few metrics to evaluate the different aspects of your QA systems namely
  - Ragas uses LLMs under the hood to do the evaluations but uses them in different way to mitigate the biases
  - There are fundamentally four metrics provided for evaluating the retrieval as well as the generating components

3. RAG evaluation challenges

- **Faithfulness**
  - Evaluates if the generated answer is backed by the context (i.e retertived document).
  - First it breaks the output into list of statements then asks sends the statement s to LLM.

    Then the ratio of correct statements will be score for faithfulness
- **Answer Relevance**:
  - Evaluates if the answer generated is relevant to the question answered.
  - It does not necessarily need to be correct
  - Is the answer to the point
  - Uses LLM to generate probable questions to the given answer, and compares them the original question

3. RAG evaluation challenges

- **Context Relevance**
  - Given a question, RAGS tries to measure if the statements in the context are relevant to the question.
  - Asks LLM if the which statements are relevant to the question
  - Another way to measure noise
- **Context Recall**:
  - Measures the ability of the retriever to retrieve all the necessary information needed to answer the question.
  - Ragas calculates this by using the provided ground_truth answer and using an LLM to check if each statement from it can be inferred in the retrieved context.

THANK YOU

www.linkedin.com/in/daniel2371