aivancity
SCHOOL FOR
TECHNOLOGY, BUSINESS & SOCIETY
PARIS-CACHAN

PGE 5

Generative AI

2023-2024
Y. Almehio
Part 3

Generative AI

- Gen AI definition
- Primary Gen AI models
- GenAI Model Types
- How does Gen AI works (fast shot)
    - Language models
    - Evolution from simple to complex models
- Transformers
    - From RNN to Transformers
    - Architecture
    - Attention mechanisms
    - Project
    - Using trained transformers

aivancity
PARIS-CACHAN

Generative AI

- Transformers
  - Representation
  - Modeling
- Transformer trained for GPT
- LLM models
  - Capacity toward human level performance
  - AI evolution
  - What is LLM model
  - How is LLM trained
- Ways of using LLM
  - Finetuning: LoRa
  - Prompting (prompt engineering)
  - RAG
  - Tools
- Final project

aivancity
PARIS–CACHAN

LLM usage techniques

Retrieval-Augmented Generation RAG

aivancity
PARIS-CACHAN

Part 3

RAG
Definition & challenges

- Improve the efficacy of LLM, specially toward generative applications
- Leveraging custom data as input context
- For a given prompt, knowledge used form custom data will be analyzed and indexed ⇒ document/context retrieval

What challenges does the retrieval augmented generation approach solve

LLM does not know your data
Business, internal knowledge

AI apps must leverage custom data to be helpful

Fine-tuning is NOT accessible to all actors
Resources, curated data

Part 3

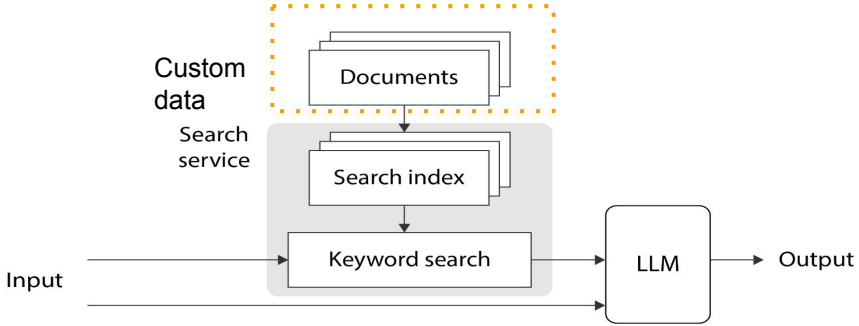
RAG *Benefits*

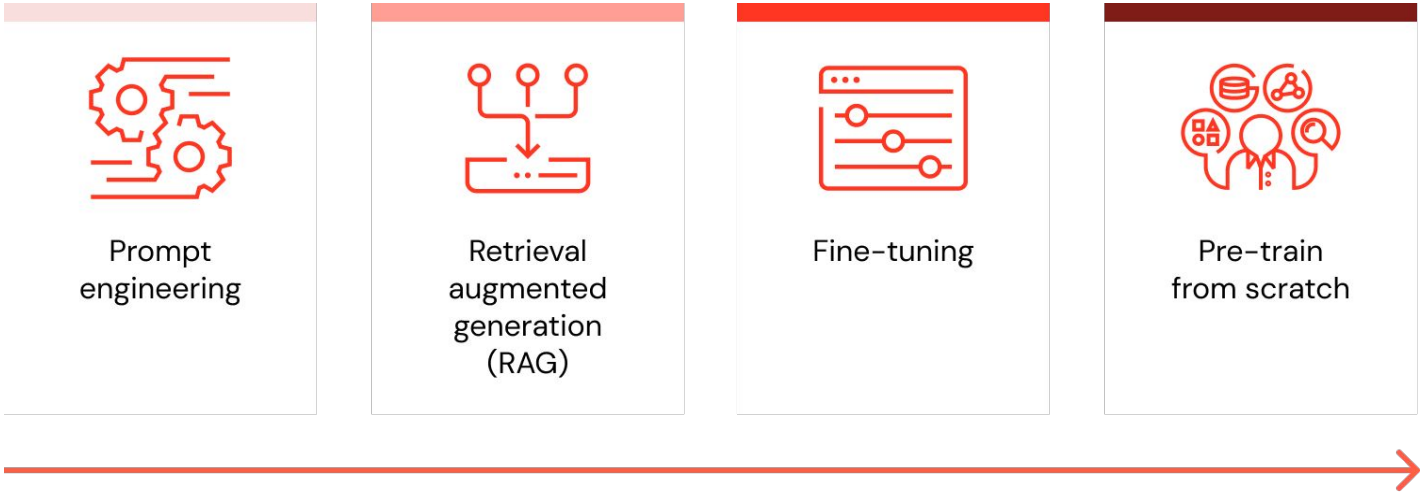
Use Cases?

- Up-to-date external data for accurate and current responses
- minimize errors and "hallucinations," ensuring responses are grounded in verifiable data
- enables the LLM to offer responses specific to particular domains or organizational needs
- Offers a simple and cost-effective solution for customizing LLMs to specific domains without the need for frequent manual updates, enhancing efficiency.

Part 3

RAG *diagram*

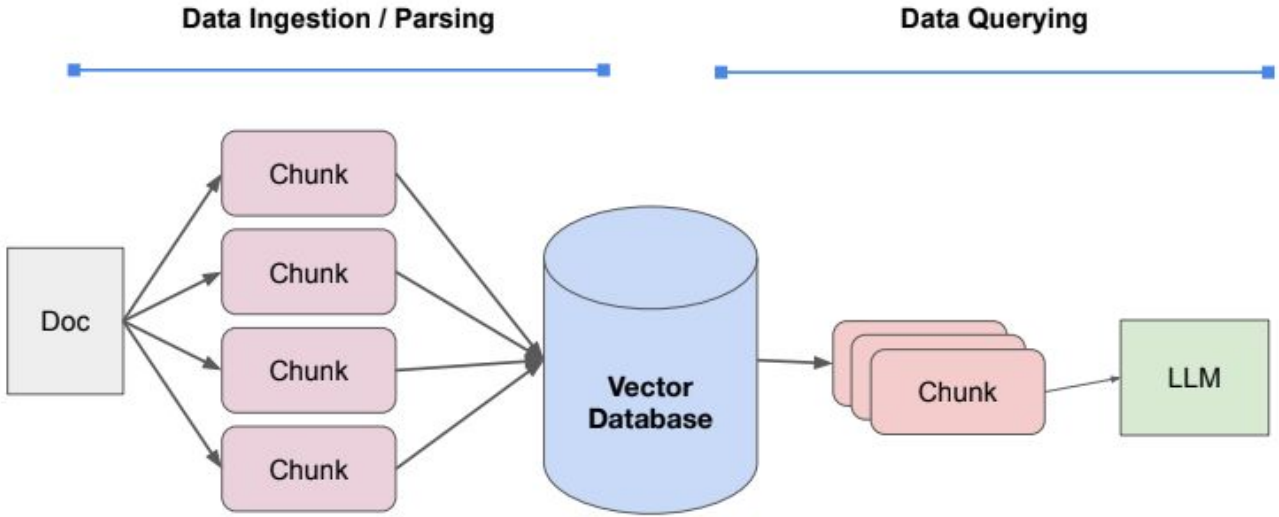




Part 3

RAG

Easy Prototype

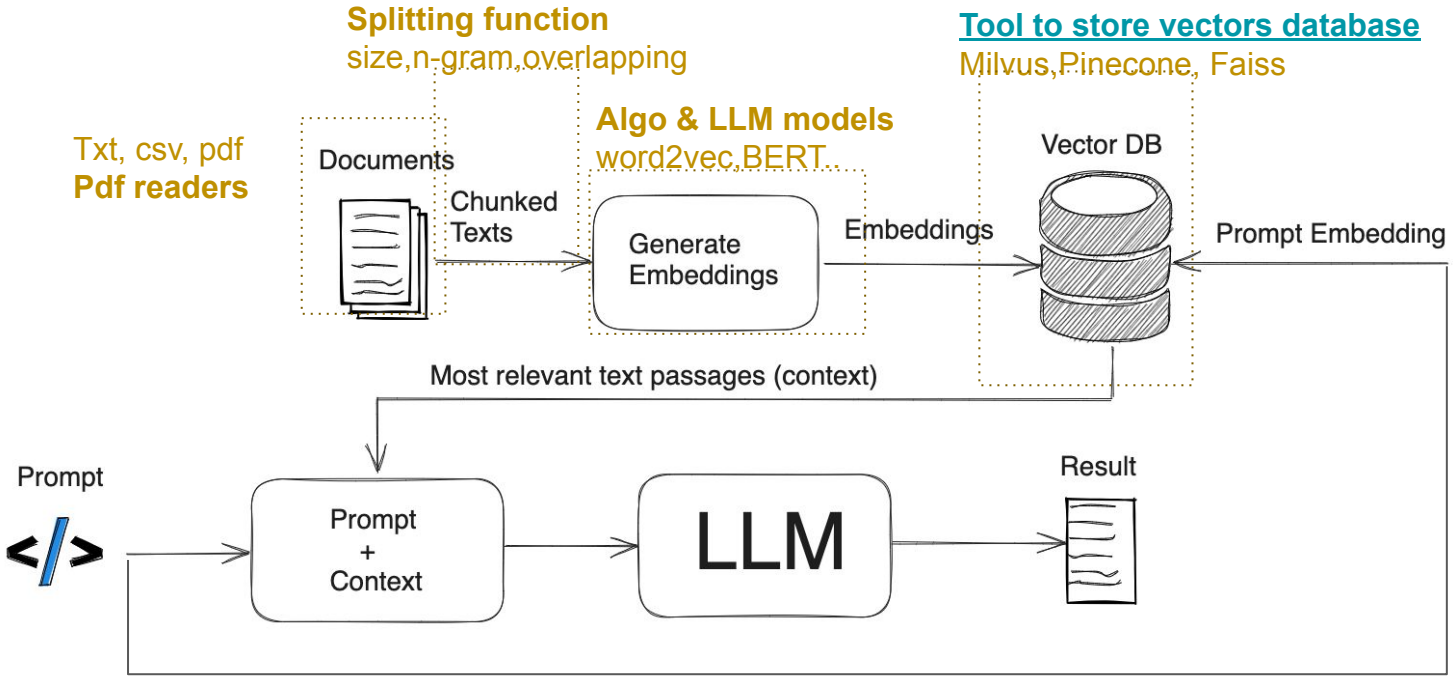


Few code lines

Part 3

RAG

Prototype components



Few code lines

Part 3

RAG

Chunking: what is the right strategy

Level 1: **Fixed sized chunking** :chunks of specified number of characters
Langchain & Llamaindex: offer CharacterTextSplitter and SentenceSplitter

Level 2: **Recursive chunking**: breaks down text into smaller pieces iteratively and hierarchically using various separators. When the initial split doesn't meet the desired chunk size or structure, the process repeats on the resulting pieces with a different separator or criteria
Langchain : RecursiveCharacterTextSplitter

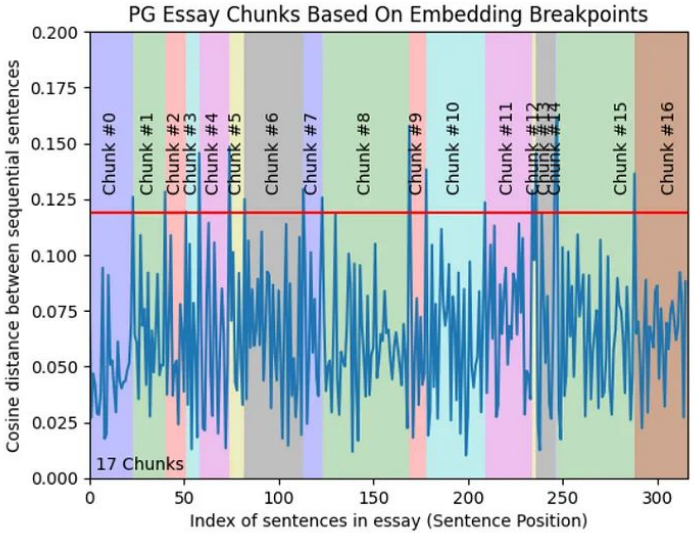
Level 3: **Content-aware chunking**: split w.r.t markdown as separator: Split with code:, split as tabular relationship.
Langchain: PythonCodeTextSplitter; MarkdownTextSplitter

Part 3

RAG

Chunking: what is the right strategy

Level 4: **semantic chunking** :previous deals with content and structures
Need a constant chunk size.
Extract the semantic meaning from embedding and assess the semantic relationship between chunks



Llamindex has [SemanticSplitterNodeParse class]

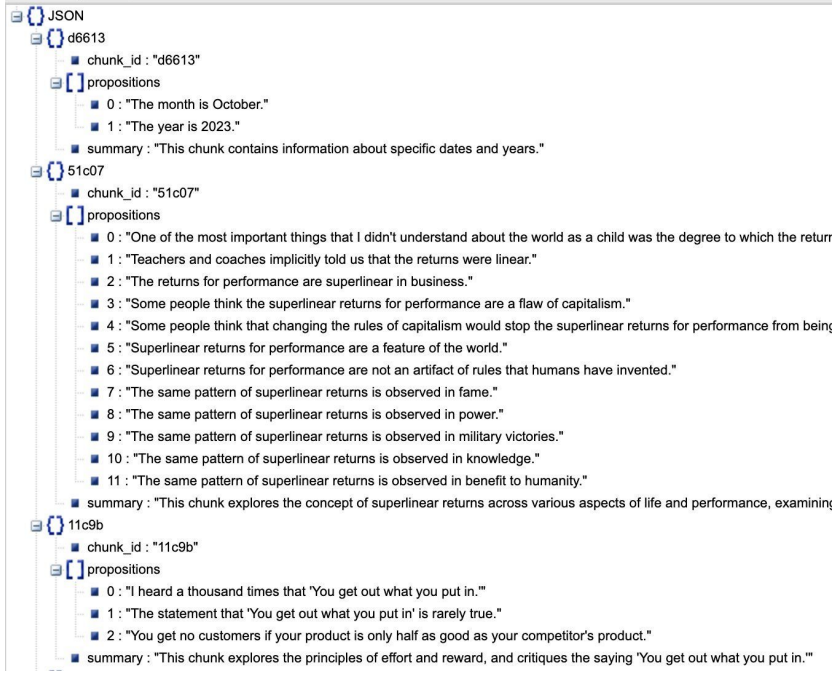
Your goal is not to chunk for chunking sake, it is to get our data in a format where it can be retrieved for value later.

Part 3

RAG

Chunking: what is the right strategy

Level 5 agentic chunking :explore the possibility to use LLM to determine how much and what text should be included in a chunk based on the context. [concept of Propositions based on paper]



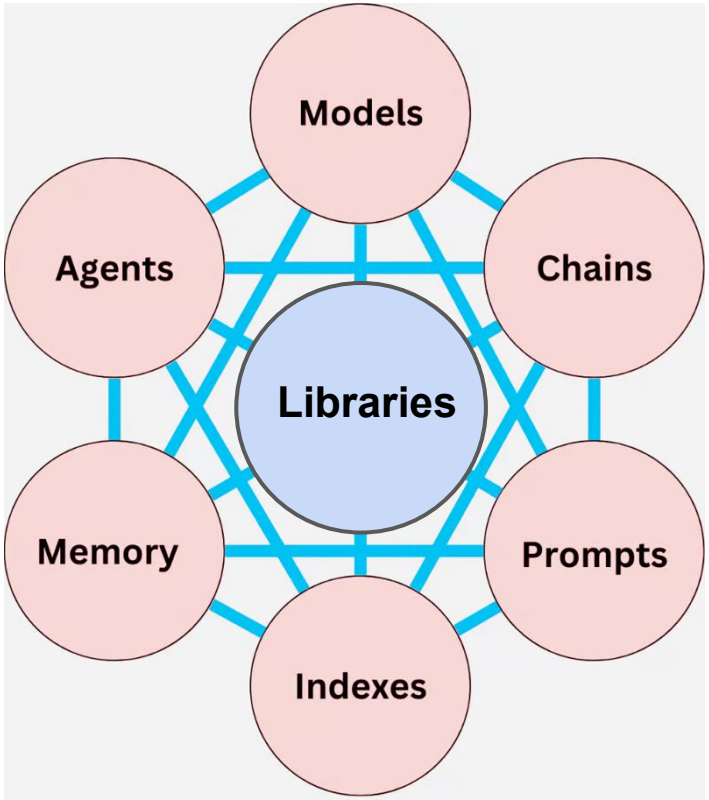
[Quick tutorial]

Tools to work with LLM

Part 1

Libraries for LLM-based applications

- **Models**: Standard interface that allows for interactions with LLM
- **Prompt**: Classes and functions to simplify the process of creating and handling prompts
- **Intelligent agents**: agents with a comprehensive toolkit, agent choose which tool to use based on user input
- **Indexes**: methods to organize document in a mapping that facilitates retrieval process with LLM
- **Chains**: For more complex task, chining few LLMs could be efficient, so standar interface and functions are provided for this purpose

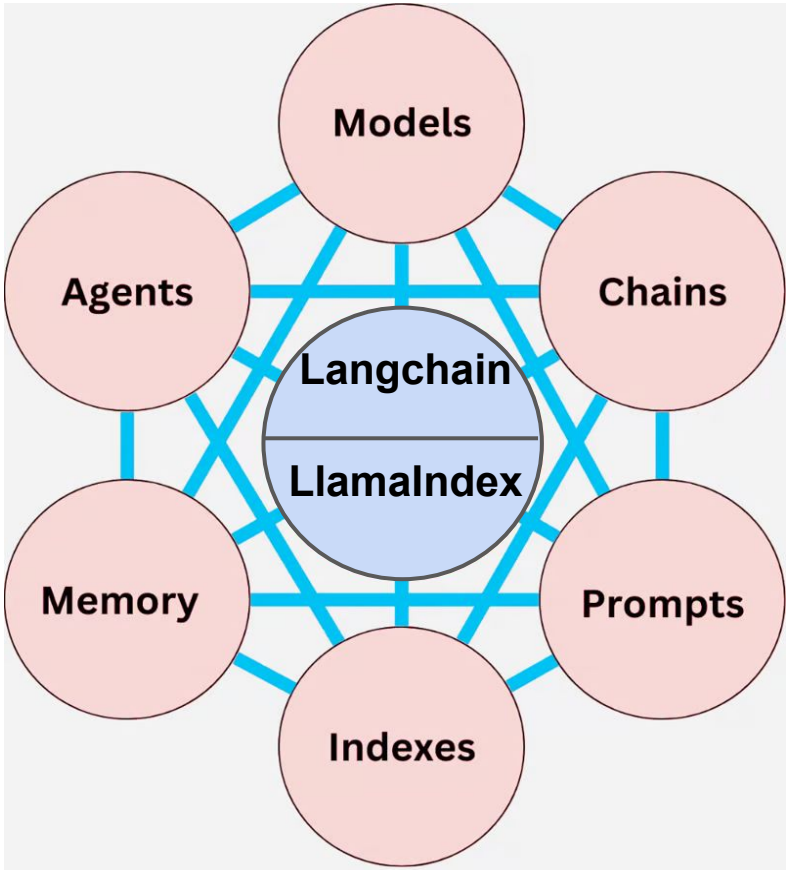


Part 1

Libraries for LLM-based applications

Langchain

a generic framework for developing stuff with LLM.



Llamaindex

A framework dedicated for building RAG Vector store for LL-based search

Part 3

Applications

Video summary
Chat with docs using local LLM (Llama 2)
Gemini

Let’s develop your own chatbot: starter blog

Part 3

General platform to RAG your docs

The target application:

- An interface lets the user to design his own RAG without code. Where he can define his external data (documents), with the help of interface,
- And to decide the LLM model (local (selectfile) or from huggingface (set a url)
- And to choose the parameters of chunking (all variant possible)
- And to set crafter prompt (zero of few shot), to communicate with the docs
- And results in: answer, retrieved doc, the found context, and the number of pages
- In case of few shot prompt, it has to provide several retrieved doncs
- Documents could be pdf, docs, csv, txt.....
    - Report, slides, papers

Part 3

General platform to RAG your docs

The application architecture:

- Gradio-based interface
- User interface window 1:
  - define which use case dataset to search with
  - to choose the model
  - To write an input prompt
- User interface window 2: to
  - Choose data docs
  - Split, embed and store them (pinecone, milvus..)
  - To tune all chunking variables

General platform to RAG your docs

Usecases dataset

- usecase1 (i.e: scientific paper)
  - Paper 1
  - Paper 2
  - ...
- Usecase 2: (i.e slides and technical reports)
  - Presentation1 (pdf)
  - Presentation 2 (pdf)
  - Report1 .
  - ....
- Usecase 3: (i.e personal docs)
  - Notes,
  - Financement
  - ....

Part 3

RAG Evaluation

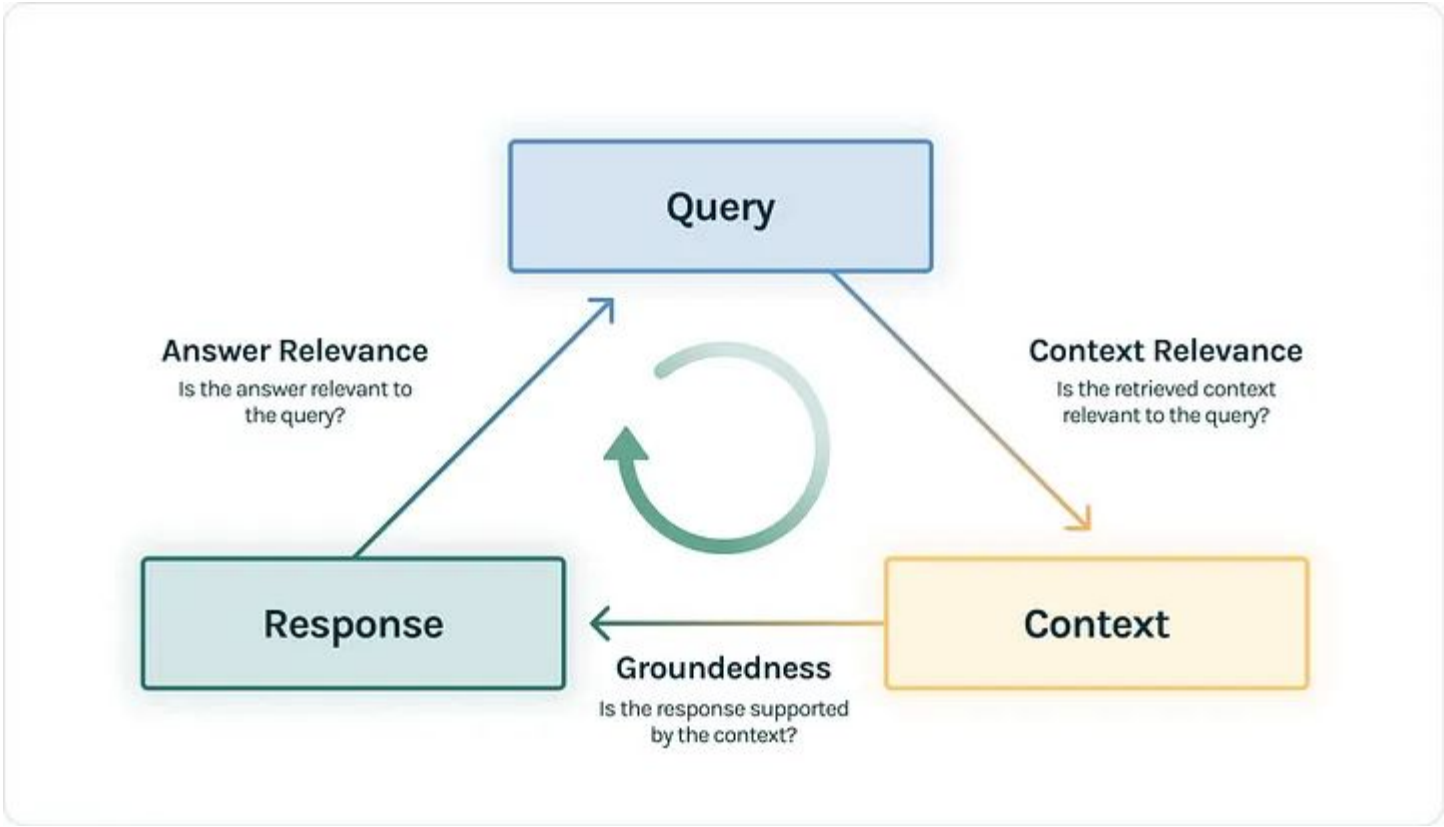
RAG merits to verify whether this data is used correctly and efficiently :

1. The response provided by the model is relevant to the query submitted by the user.
2. The parts of the external data retrieved by RAG are relevant to the user query.
3. The response the model produces is based on the retrieved data rather than its own hallucinations or general knowledge gained during pre-training.

TrueLens is an open-source tool for RAG evaluation. It formalizes the three concepts listed above as pair-wise comparisons between the response, the query, and the context (which is the retrieved external data).

Part 3

RAG Evaluation



Hypothetical Document Embeddings(HyDE)

Do a small research on this subject and answer the following questions:

1. Hyde definition
2. WHy it’s needed
3. Is there any disadvantages


WHEN YOU FINALLY UNDERSTAND
HOW
GENERATIVE AI WORKS