deeplearning.ai Short Course

Building AI Applications Using Haystack

Date: 25 08 2024

(1) Haystack Building Blocks

- Components take in inputs & Outputs
- Pipelines are composed by connecting components

- OpenAI Document Embedder : Component
  text-embedding-3-small : OpenAI emb model
  takes in list[Document], returns Some + meta

- Converter, splitter, embedder, writer
- These Components added to pipeline & Connected
- show() Creates pipeline flowchart

- Text file → list[doc] → Split to → Create → Write
  Smaller docs emb in mem

Doc search pipeline

- OpenAI text Emb, In mem Emb retriever
- query_embedder.embedding → retriever.query_emb
- Query → embedded → top k results from query
  In mem doc store

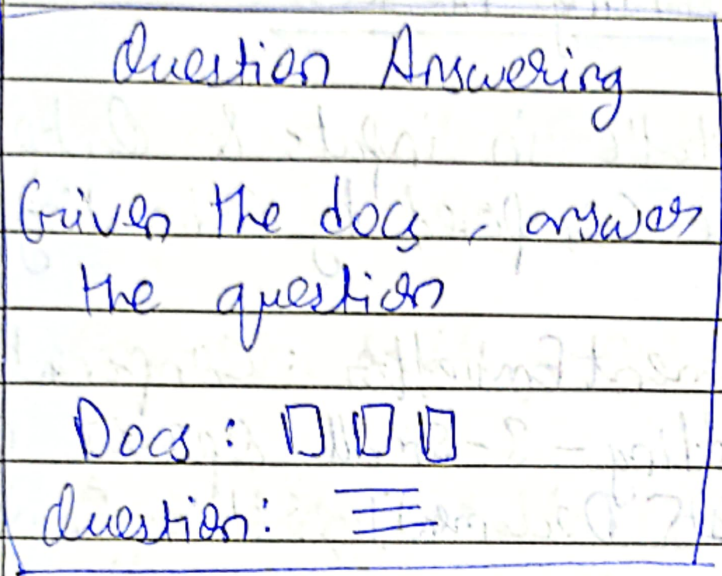
top_k Changed, Questions changed

#learnthesmarterway

② Build Customized RAG

Date: __/__/____

RAG: Retrieval-Augmented Generation



Retrieval: finding relevant docs for query

1. Embedding Retrieval (or Semantic Search)
2. Keyword based Retrieval (ex: BM25)
3. Retrieval + Reranking
4. Retrieval from a different API (web, email, slack etc)

- Indexing & RAG pipeline

Date: __/__/____

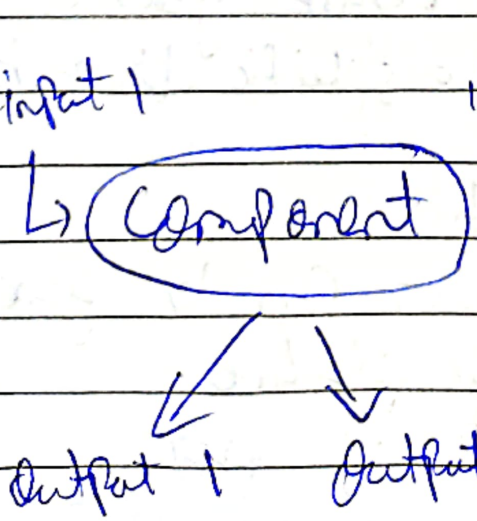
Indexing pipeline: Urls → LinkContent Fetcher → HTML to Document converter → Cohere doc Embedder → Writer

- Jinja is used as prompt template tool

RAG pipeline: text → CohereText Embedder → InMem EmbRet → Prompt Builder → OpenAI gen → o/p

- Behavior Customized by modifying the prompt to include Source URL & o/p lang

(3) Custom Components : News Summarizer



Ready made components:
- Embedders
- Retrievers
- Generators
- Preprocessors
- Rankers
- Routers

Custom Components

Date: __/__/____

- class with @component
- run() w/ @component.output_types
- should return a dict

- Greeter
- Greeter + Dialogue Builder
- HN Summarizer

top-k → HN → Prompt → OpenAI → Output
int Fetcher Builder Gen (Summary)

- Edited prompt to create md table

4 Fallbacks with branching pipelines

- Conditional Router: Define your own route
  Trigger a specific branch

- Fallback to websearch:
  ① Evaluate if answer can be generated using the RAG pipeline
  ② Use web search if not

Date: __/__/____

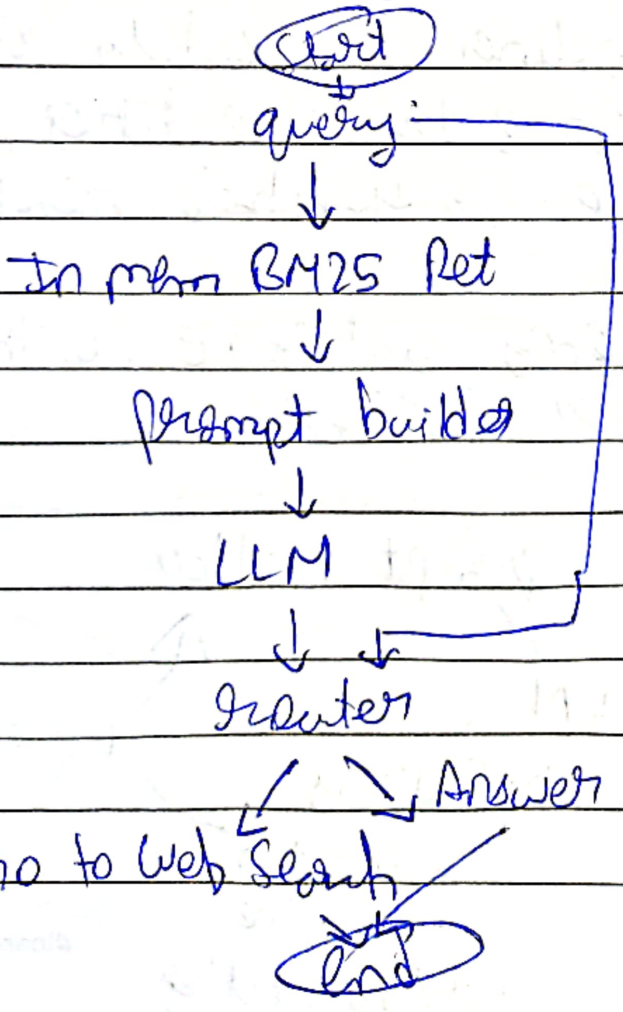
- Prompt template has: "If the answer is not contained within the docs, reply with no_answer"

- Keyword based (BM25) Retriever

query → In mem BM25 Retriever → Prompt builder → llm → o/p

- Conditional Router routes is a list[dict]

- no_answer is lowercased in Jinja



#learnthesmarterway

(5) Self-Reflecting Agents with loops

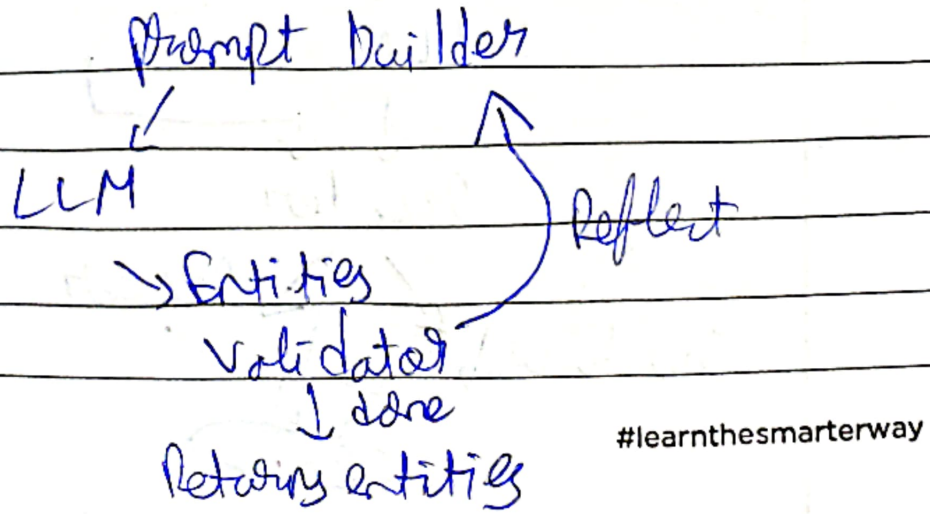
Date: __/__/____

Task
↓
Generate (LLM) ←
↓
Response
↓
Reflect
↓
Done

Critique/ Reflect

- Self Reflection is a mechanism for self assessment and correction
- Refining outputs, providing more reliable and accurate info

- Looping Pipelines for entity extraction, Reflect on extracted entities, loop until done (or threshold reached)
- JSON schema adherance, with loop back for correction
-



#learnthesmarterway

⑥ Chat Agent w/ Function Calling

Date: __/__/____

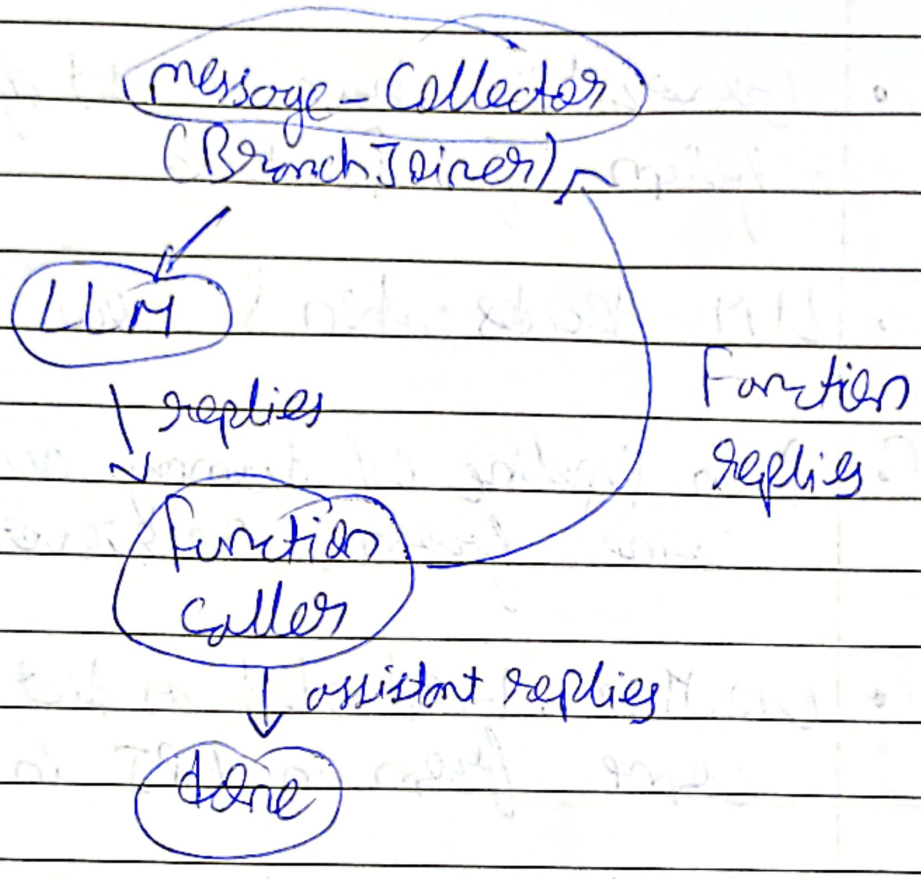
- Generators have a set of tools in the form of functions
- LLM decides when & how to call a fn

Tools to LLM:

- RAG pipeline w/ dummy docs (These would come from a retriever in real usecase)
- Weather: Hardcoded as dict (These would come from an API in real use case)

- tools defined as list[dict] w/ each dict having the functions name, desc & params
- OpenAIChatGenerator: takes in the user query and returns a resp that indicates which tool to use
- OpenAIFunctionCaller: processes a list of chat msgs and calls Python fns when needed
- BranchJoiner: Joins multiple branches in a pipeline, allowing their outputs to be reconciled into a single Branch

Date: __/__/____



- Msgs either come from User or Function caller replies
- With a initial Sys prompt, a loop is started for back and forth b/w user and bot, with the bot using tools when required and exits manually
- Some is served as a Gradio app