OpenAI

RAG
Technique

February 2024

Overview

Retrieval-Augmented Generation enhances the capabilities of language models by combining them with a retrieval system. This allows the model to leverage external knowledge sources to generate more accurate and contextually relevant responses.

Example use cases

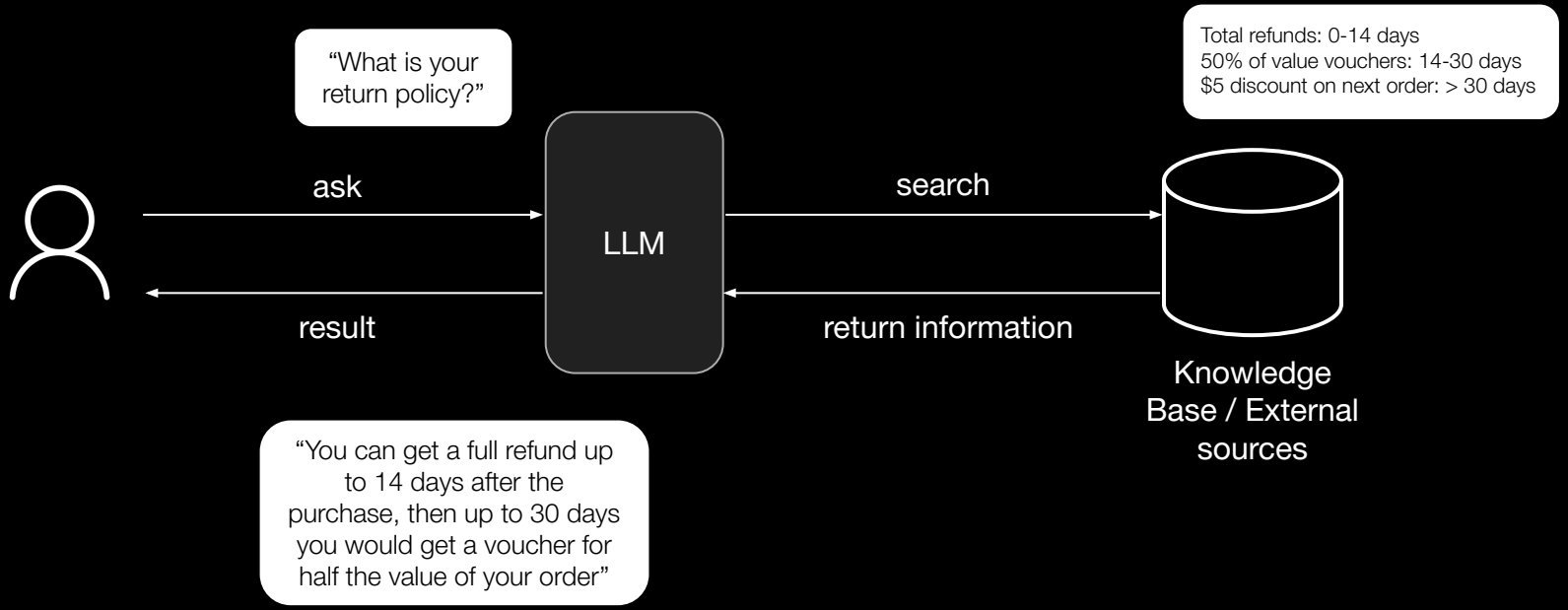
- Provide answers with up-to-date information
- Generate contextual responses

What we'll cover

- Technical patterns
- Best practices
- Common pitfalls
- Resources

What is RAG

Retrieve information to **A**ugment the model's knowledge and **G**enerate the output



When to use RAG

Good for ✅

- Introducing new information to the model to update its knowledge
- Reducing hallucinations by controlling content
  /!\ Hallucinations can still happen with RAG

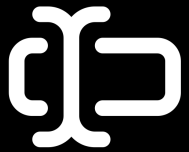
Not good for ❌

- Teaching the model a specific format, style, or language
  ➔ Use fine-tuning or custom models instead
- Reducing token usage
  ➔ Consider fine-tuning depending on the use case

Technical patterns

Data preparation

- Chunking
- Embeddings
- Augmenting content



Input processing

- Input augmentation
- NER
- Embeddings

Retrieval

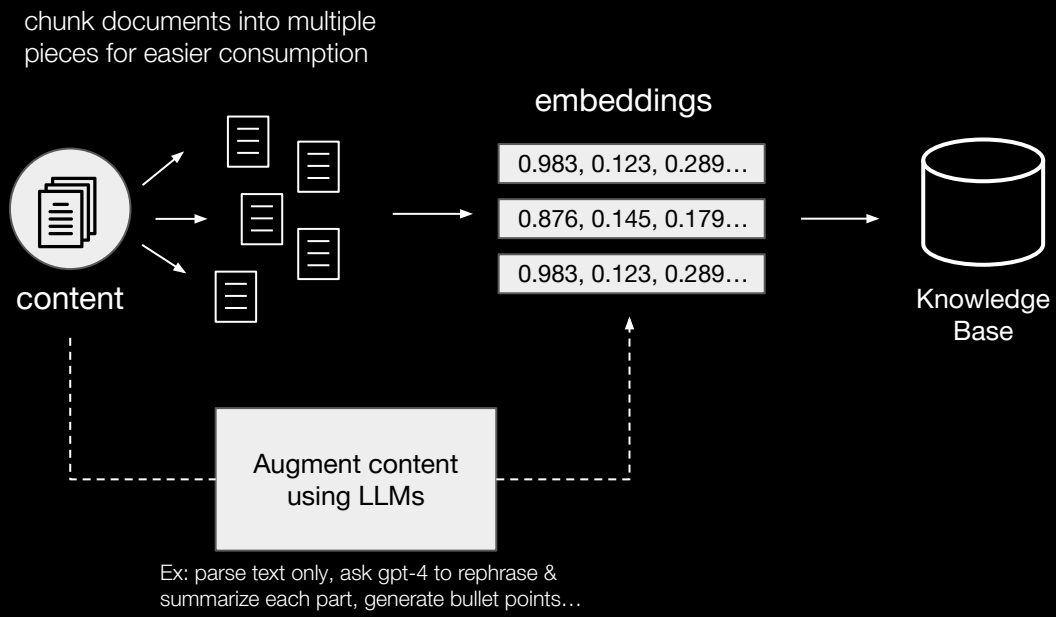
- Search
- Multi-step retrieval
- Re-ranking

Answer Generation

- Context window
- Optimisation
- Safety checks

Technical patterns

Data preparation



BEST PRACTICES

Pre-process content for LLM consumption:
Add summary, headers for each part, etc.
+ curate relevant data sources

COMMON PITFALLS

- ➔ Having too much low-quality content
- ➔ Having too large documents

Technical patterns

Data preparation: chunking

Why chunking?

If your system doesn't require entire documents to provide relevant answers, you can chunk them into multiple pieces for easier consumption (reduced cost & latency).

Other approaches: graphs or map-reduce

Things to consider

- Overlap:
  - Should chunks be independent or overlap one another?
  - If they overlap, by how much?
- Size of chunks:
  - What is the optimal chunk size for my use case?
  - Do I want to include a lot in the context window or just the minimum?
- Where to chunk:
  - Should I chunk every N tokens or use specific separators?
  - Is there a logical way to split the context that would help the retrieval process?
- What to return:
  - Should I return chunks across multiple documents or top chunks within the same doc?
  - Should chunks be linked together with metadata to indicate common properties?

Technical patterns

Data preparation: embeddings

What to embed?

Depending on your use case you might not want just to embed the text in the documents but metadata as well - anything that will make it easier to surface this specific chunk or document when performing a search

Examples

Embedding Q&A posts in a forum
You might want to embed the title of the posts, the text of the original question and the content of the top answers.
Additionally, if the posts are tagged by topic or with keywords, you can embed those too.

Embedding product specs
In additional to embedding the text contained in documents describing the products, you might want to add metadata that you have on the product such as the color, size, etc. in your embeddings.

Technical patterns

Data preparation: augmenting content

What does “Augmenting content” mean?

Augmenting content refers to modifications of the original content to make it more digestible for a system relying on RAG. The modifications could be a change in format, wording, or adding descriptive content such as summaries or keywords.

Example approaches

Make it a guide*
Reformat the content to look more like a step-by-step guide with clear headings and bullet-points, as this format is more easily understandable by an LLM.

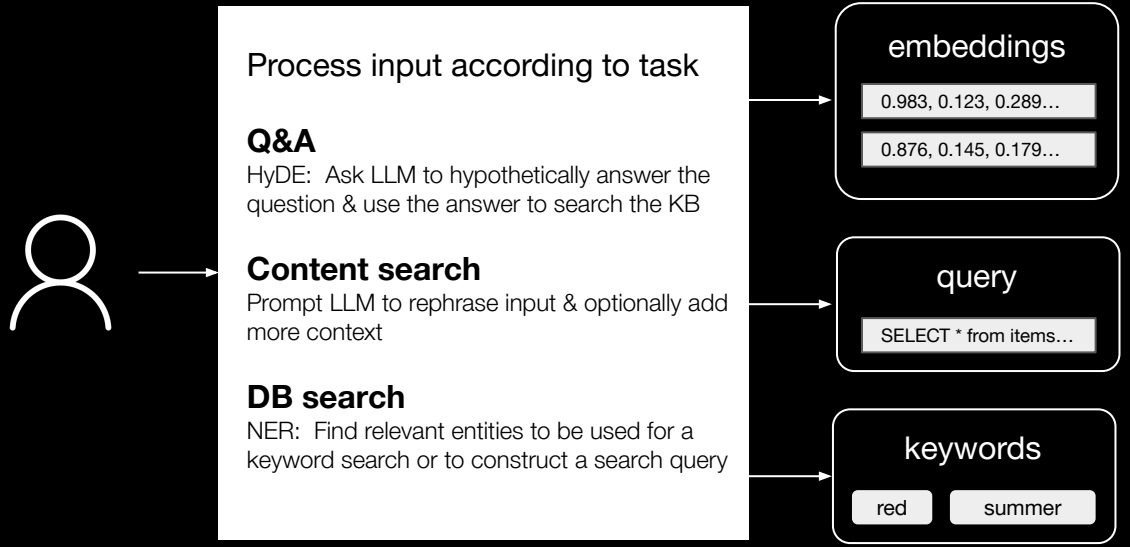
Add descriptive metadata*
Consider adding keywords or text that users might search for when thinking of a specific product or service.

Multimodality
Leverage models such as Whisper or GPT-4V to transform audio or visual content into text.
For example, you can use GPT-4V to generate tags for images or to describe slides.

* GPT-4 can do this for you with the right prompt

Technical patterns

Input processing



BEST PRACTICES

Consider how to transform the input to match content in the database
Consider using metadata to augment the user input

COMMON PITFALLS

➔ Comparing directly the input to the database without considering the task specificities

Technical patterns

Input processing: input augmentation

What is input augmentation?

Augmenting the input means turning it into something different, either rephrasing it, splitting it in several inputs or expanding it.
This helps boost performance as the LLM might understand better the user intent.

Example approaches

Query expansion*
Rephrase the query to be more descriptive

HyDE*
Hypothetically answer the question & use the answer to search the KB

Fallback
Consider implementing a flow where the LLM can ask for clarification when there is not enough information in the original user query to get a result (Especially relevant with tool usage)

Splitting a query in N*
When there is more than 1 question or intent in a user query, consider splitting it in several queries

* GPT-4 can do this for you with the right prompt

Technical patterns

Input processing: NER

Why use NER?

Using NER (Named Entity Recognition) allows to extract relevant entities from the input, that can then be used for more deterministic search queries.
This can be useful when the scope is very constrained.

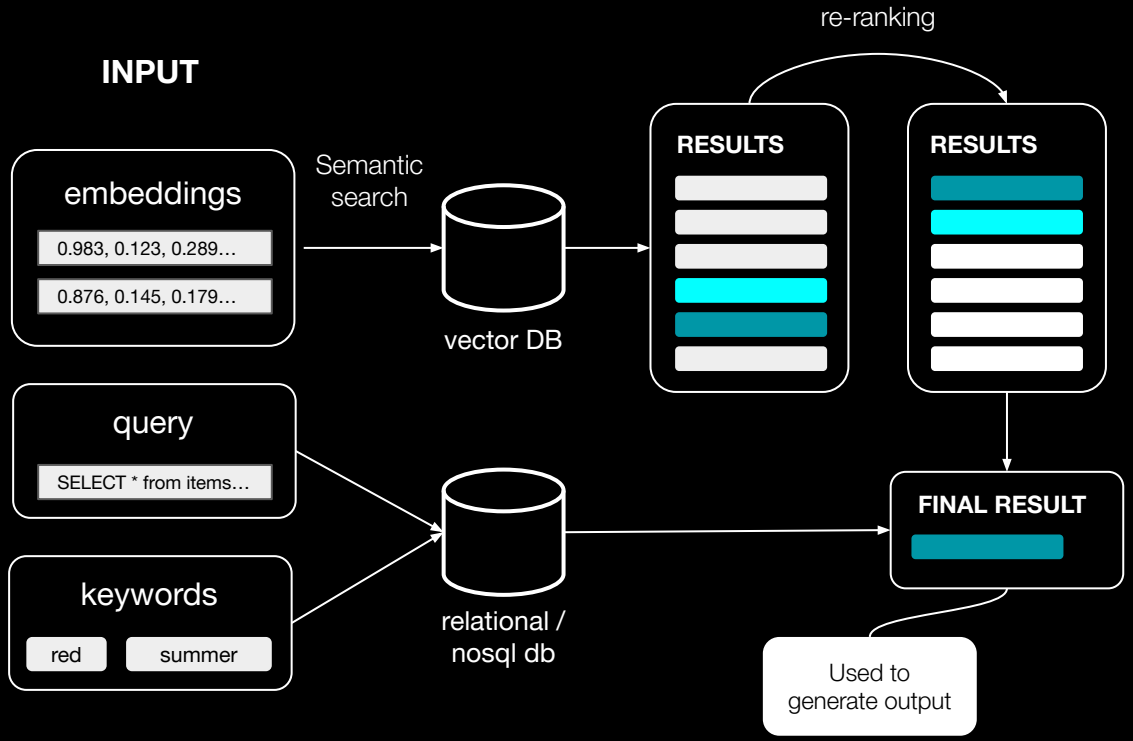
Example

Searching for movies
If you have a structured database containing metadata on movies, you can extract genre, actors or directors names, etc. from the user query and use this to search the database

Note: You can use exact values or embeddings after having extracted the relevant entities

Technical patterns

Retrieval



BEST PRACTICES

Use a combination of semantic search and deterministic queries where possible

+ Cache output where possible

COMMON PITFALLS

➔ The wrong elements could be compared when looking at text similarity, that is why re-ranking is important

Technical patterns

Retrieval: search

How to search?

There are many different approaches to search depending on the use case and the existing system.

Semantic search

Using embeddings, you can perform semantic searches. You can compare embeddings with what is in your database and find the most similar.

Keyword search

If you have extracted specific entities or keywords to search for, you can search for these in your database.

Search query

Based on the extracted entities you have or the user input as is, you can construct search queries (SQL, cypher…) and use these queries to search your database.

You can use a hybrid approach and combine several of these. You can perform multiple searches in parallel or in sequence, or search for keywords with their embeddings for example.

Technical patterns

Retrieval: multi-step retrieval

What is multi-step retrieval?

In some cases, there might be several actions to be performed to get the required information to generate an answer.

Things to consider

- Framework to be used:
  - When there are multiple steps to perform, consider whether you want to handle this yourself or use a framework to make it easier
- Cost & Latency:
  - Performing multiple steps at the retrieval stage can increase latency and cost significantly
  - Consider performing actions in parallel to reduce latency
- Chain of Thought:
  - Guide the assistant with the chain of thought approach: break down instructions into several steps, with clear guidelines on whether to continue, stop or do something else.
  - This is more appropriate when tasks need to be performed sequentially - for example: “if this didn’t work, then do this”

Technical patterns

Retrieval: re-ranking

What is re-ranking?

Re-ranking means re-ordering the results of the retrieval process to surface more relevant results.
This is particularly important when doing semantic searches.

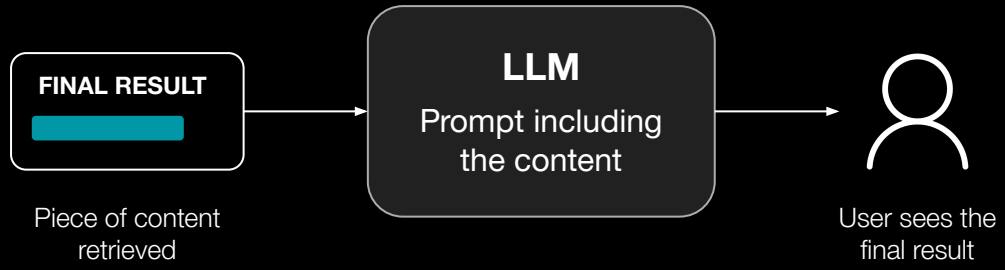
Example approaches

Rule-based re-ranking
You can use metadata to rank results by relevance. For example, you can look at the recency of the documents, at tags, specific keywords in the title, etc.

Re-ranking algorithms
There are several existing algorithms/approaches you can use based on your use case: BERT-based re-rankers, cross-encoder re-ranking, TF-IDF algorithms…

Technical patterns

Answer Generation



BEST PRACTICES

Evaluate performance after each experimentation to assess if it's worth exploring other paths
+ Implement guardrails if applicable

COMMON PITFALLS

- ➔ Going for fine-tuning without trying other approaches
- ➔ Not paying attention to the way the model is prompted

Technical patterns

Answer Generation: context window

How to manage context?

Depending on your use case, there are several things to consider when including retrieved content into the context window to generate an answer.

Things to consider

- Context window max size:
  - There is a maximum size, so putting too much content is not ideal
  - In conversation use cases, the conversation will be part of the context as well and will add to that size
- Cost & Latency vs Accuracy:
  - More context results in increased latency and additional costs since there will be more input tokens
  - Less context might also result in decreased accuracy
- “Lost in the middle” problem:
  - When there is too much context, LLMs tend to forget the text “in the middle” of the content and might look over some important information.

Technical patterns

Answer Generation: optimisation

How to optimise?

There are a few different methods to consider when optimising a RAG application. Try them from left to right, and iterate with several of these approaches if needed.

Prompt Engineering

At each point of the process, experiment with different prompts to get the expected input format or generate a relevant output.
Try guiding the model if the process to get to the final outcome contains several steps.

Few-shot examples

If the model doesn't behave as expected, provide examples of what you want e.g. provide example user inputs and the expected processing format.

Fine-tuning

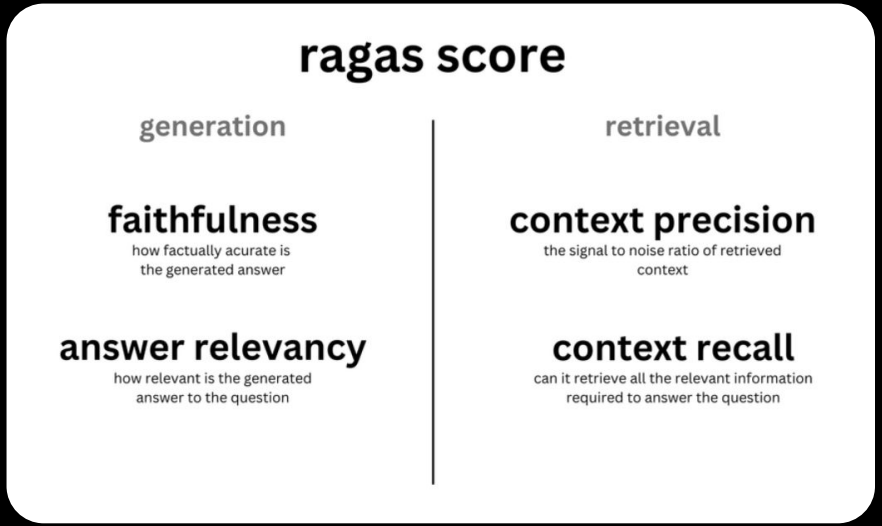
If giving a few examples isn't enough, consider fine-tuning a model with more examples for each step of the process: you can fine-tune to get a specific input processing or output format.

Technical patterns

Answer Generation: safety checks

Why include safety checks?

Just because you provide the model with (supposedly) relevant context doesn't mean the answer will systematically be truthful or on-point. Depending on the use case, you might want to double-check.



Example evaluation framework: RAGAS