Retrieval Augmented Generation

"RAG takes input and retrieves a set of relevant/supporting documents given a source (e.g., Wikipedia). The documents are concatenated as context with the original input prompt and fed to the text generator which produces the final output. This makes RAG adaptive for situations where facts could evolve over time. This is very useful as Large Language Models's parametric knowledge is static. RAG allows language models to bypass retraining, enabling access to the latest information for generating reliable outputs via retrieval-based generation."

Introduction

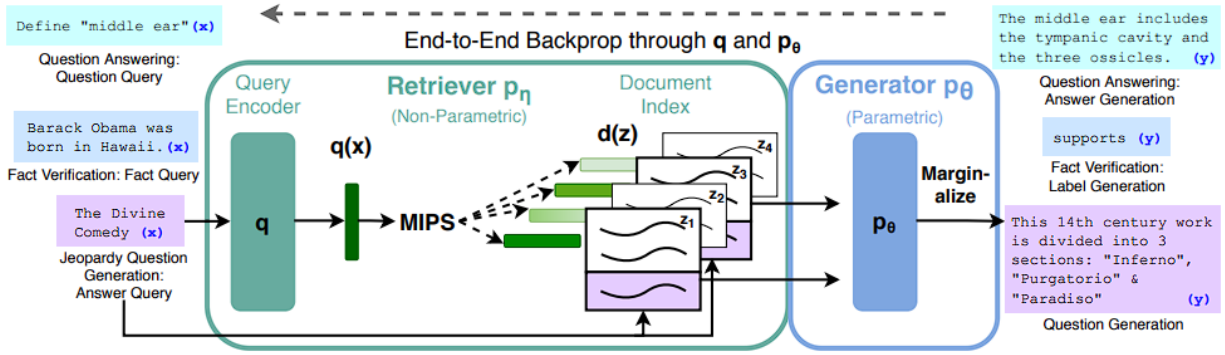
- LLM's ability to access and precisely manipulate knowledge is still limited
- Performance lags on knowledge extensive tasks.
- Continuous need to retrain and finetune models
- Pretrained models with a differentiable access mechanism to explicit non-parametric memory have so far been only investigated for extractive downstream tasks.
- **Parametric Data:** Data which the model stores while training the model. Basically the training data of the LLM
    - Parametric memory is a feature of pre-trained neural language models like BERT or GPT, refers to the model's ability to store knowledge in the weights of its neural network.
    - Models with parametric memory are notable for their ability to grasp linguistic nuances and complex contexts but have limitations in terms of knowledge updating and the explainability of their decisions.
- **Non Parametric Data:** Data which is outside the training scope of the model.
    - It stores information in an external database (such as a collection of documents or Wikipedia itself), allowing the system to consult and retrieve specific information when needed.

RAG

Introduced by Meta in 2021

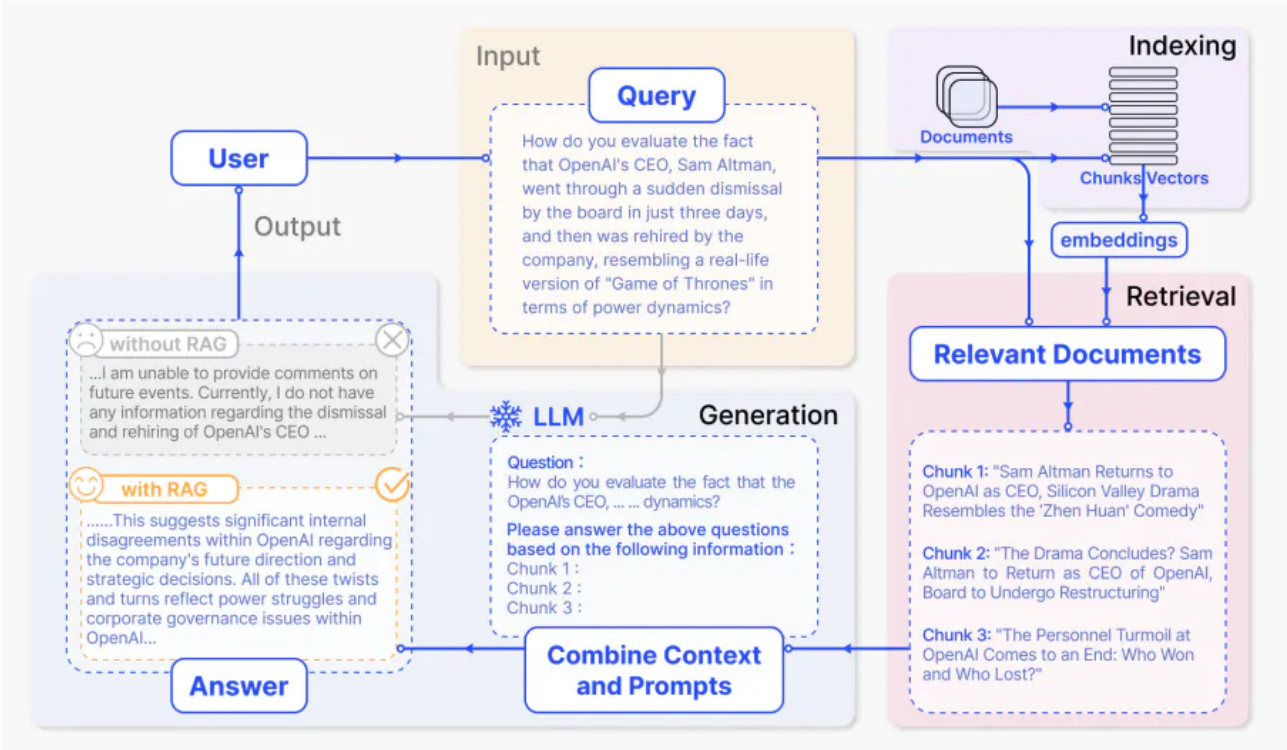
- **Parametric Memory**: Pre-trained Seq-2-Seq Model.
- **Non Parametric Memory:** Vector Database of Wikipedia with a pre-trained neural retriever

- **REALM -** Retrieval-Augmented Language Model Pre-Training
- **ORQA -** Open Retrieval Question Answering
  `Retrieval Augmented Generation Architecture



RAG model consists of:

- pre-trained retriever (Query Encoder + Document Index)
- Seq-2-Seq model (Generator)
- Query Encoder - MIPS (Maximum Inner Product Search) - finds top-K documents
- DPR (BERT was used to produce the latent document) - Dense Passage Retriever, provides latent documents conditioned on the input,
- Generator(BART was used in paper) - conditions on the inputs all together to provide the output.
- Uses top-k approximation for choosing the most relevant documents



DPR - Dense Passage Retriever

- Follows a bi-encoder architecture.

$$p_\eta(z|x) \propto (\text{BERT}_d(z)) > \text{BERT}_q(x)$$

$$p_\eta(z|x) \propto \exp(d(z)) > q(x)$$

- d(z) - dense representation of a document

RAG Encoding

- The RAG Encoder consists of two major parts namely:
  - **Encoder** - Embedding Models (**Example**: OpenAI, Jina, Ollama, Sentence Transformers)
  - **Re-ranker** - BGE, Cohere, etc.
- The process is as follows:
  - The encoder encodes the documents to vectors
  - The retriever retrieves the top-k documents.
  - The re-ranker re-ranks the retrieved documents to bring highly relevant documents on top.
- **Note:** Re-Rankers were introduced later after the initial paper by Facebook. There won't be any mention of re-rankers in the RAG paper. Read this paper to view the use of re-rankers in RAG

RAG Decoding

Decoding in RAG models follows two main approaches, depending on the variant used:

- **RAG-Sequence**: In this approach, the model uses the same retrieved document to predict each token in the response. This is useful for maintaining consistency and cohesion when the response requires a unified information base.
- **RAG-Token**: Unlike RAG-Sequence, RAG-Token allows different documents to be used for predicting each token. This offers flexibility to the model to compile responses from multiple information sources, making it ideal for questions that span a broader range of topics.

Evaluating RAG Models

- Hit Rate
- Mean Reciprocal Rank (MRR)

Hit Rate

- **Hit Rate** - fraction of queries for which the answer is present in top-k retrievals
- For a given query, if we have a top 'k' value of 5, and if the correct answer is present in the first 5 retrieved documents, then the hit rate will be 1; otherwise, it will be 0.

$$\text{Hit Rate} = \frac{\text{Number of Relevant Items Retrieved}}{\text{Total Number of Relevant Items}}$$

Mean Reciprocal Rank

- Based on the rank of highest relevant document.
- Takes the inverse of the rank of relevant document

$$\text{MRR} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\text{rank}_i}$$

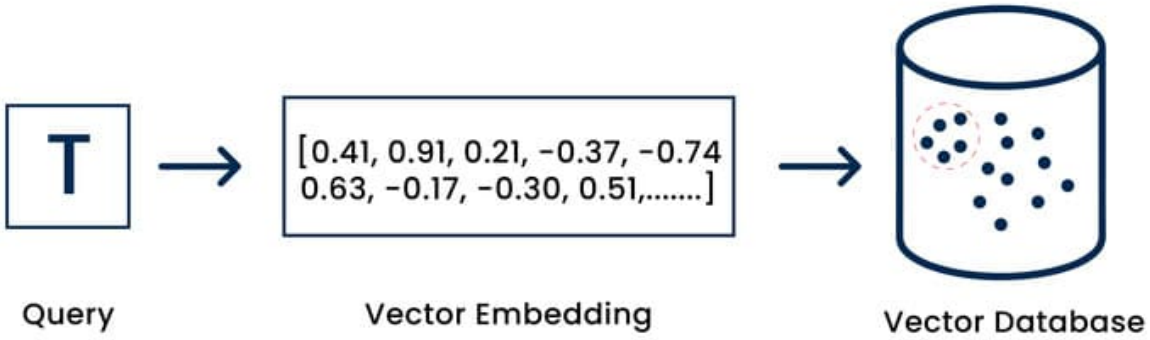
Benefits

- Reduced '**hallucinations**' of models.
- Updated LLMs
- Increased accuracy and reliability as the output is based on data sources.

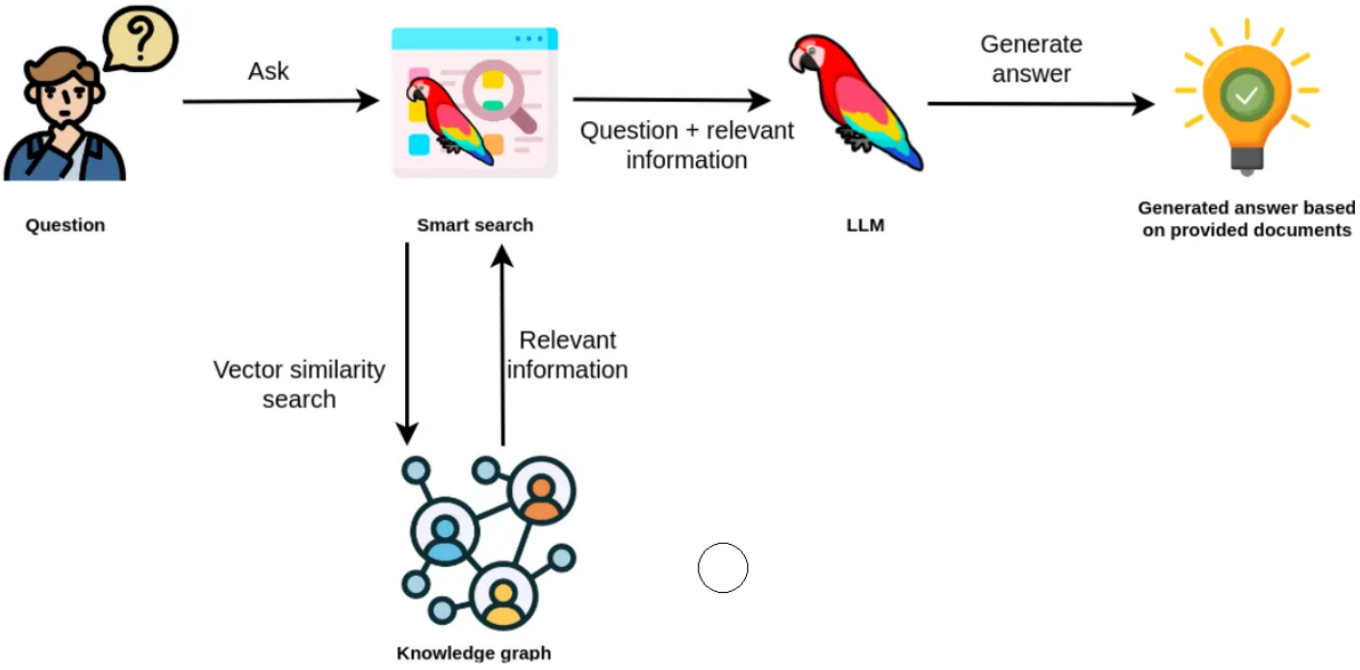
Other methods of retrieval

Vector Databases

- Vector databases is a collection of data stored as mathematical representations. A vector database indexes and stores vector embeddings for fast retrieval and similarity search, with capabilities like CRUD operations, metadata filtering, horizontal scaling, and serverless.
- A vector database uses a combination of different algorithms that all participate in Approximate Nearest Neighbour (ANN) search. These algorithms optimize the search through hashing, quantization, or graph-based search.

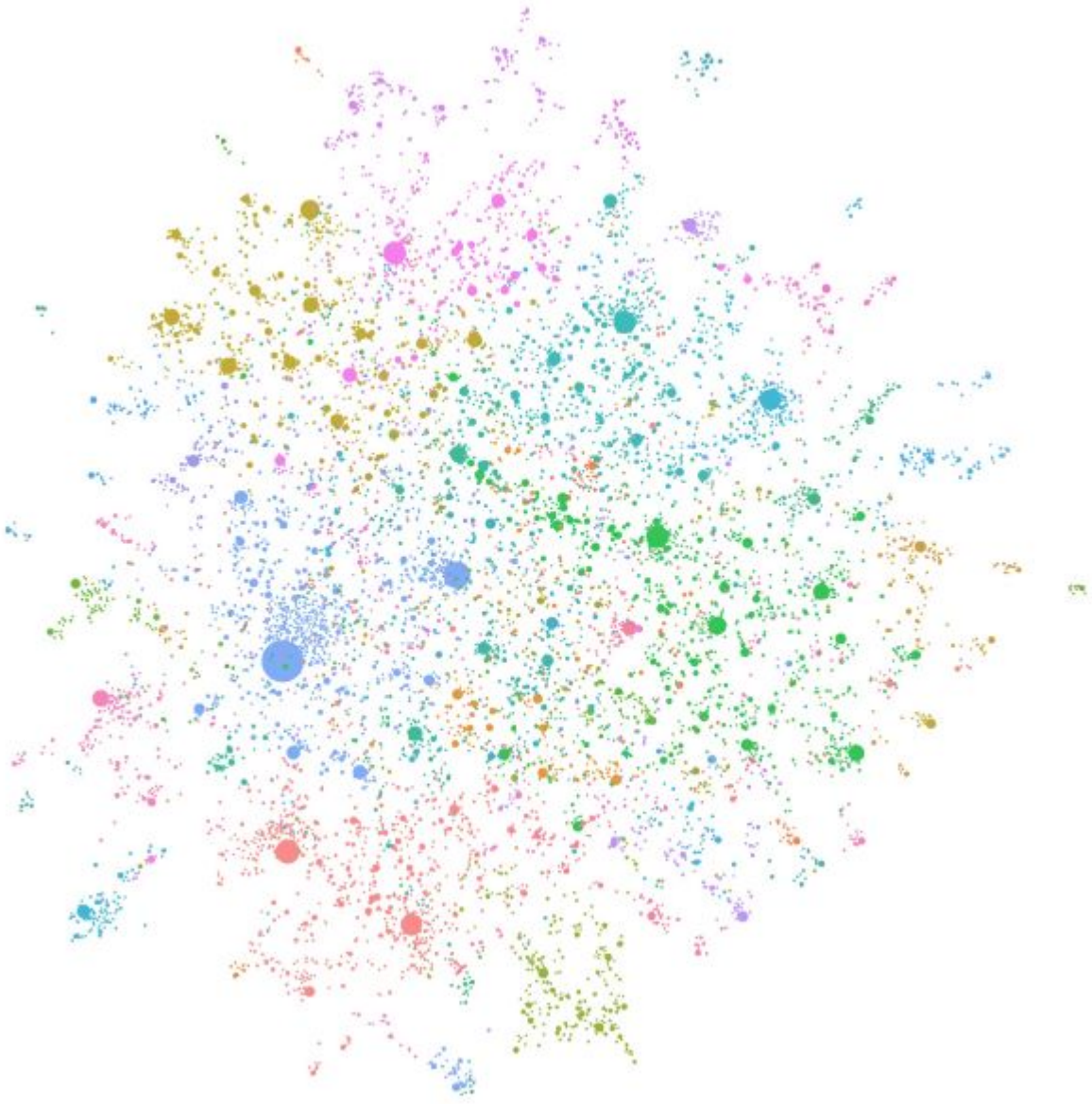


Graph Representations of RAG



- Graph RAG process involves extracting knowledge graphs out of raw text, and leveraging the knowledge graphs to perform RAG Tasks.
- The knowledge graph is created using LLMs
- This graph, along with community summaries and graph machine learning outputs, are used to augment prompts at query time.

- The hierarchical clustering of Graph are performed using a method called Leiden Technique.
- **Query Methods:**
  * *Global Search*: for reasoning about holistic questions about the corpus by leveraging the community summaries.
  * *Local Search*: for reasoning about specific entities by fanning-out to their neighbors and associated concepts.



Knowledge Graph Generated by GPT-4 Turbo

Links Referred:

- https://research.ibm.com/blog/retrieval-augmented-generation-RAG
- https://bradsol.com/resources/blogs/retrieval-augmented-generation-rag-for-llms-a-guide-to-enhanced-accuracy-and-trust/
- https://www.promptingguide.ai/research/rag
- https://srk.ai/blog/004-ai-llm-retrieval-eval-llamaindex

- https://www.llamaindex.ai/blog/boosting-rag-picking-the-best-embedding-reranker-models-42d079022e83
- https://medium.com/@nelson.miranda_40644/training-and-decoding-in-rag-models-bridging-knowledge-and-language-235be2bf4830
- https://www.rungalileo.io/blog/mastering-rag-how-to-architect-an-enterprise-rag-system
- https://www.microsoft.com/en-us/research/blog/graphrag-unlocking-llm-discovery-on-narrative-private-data/
- https://www.pinecone.io/learn/vector-database/
- https://microsoft.github.io/graphrag/
- https://youtu.be/sVcwVQRHIc8?si=UE6fpSDnGXg1Izct

Papers worth reading:
https://arxiv.org/pdf/2409.11136