AI 2025
SUMMER SCHOOL
ai.uni-jena.de

Retrieval-Augmented Generation (RAG)

Zhe Wang

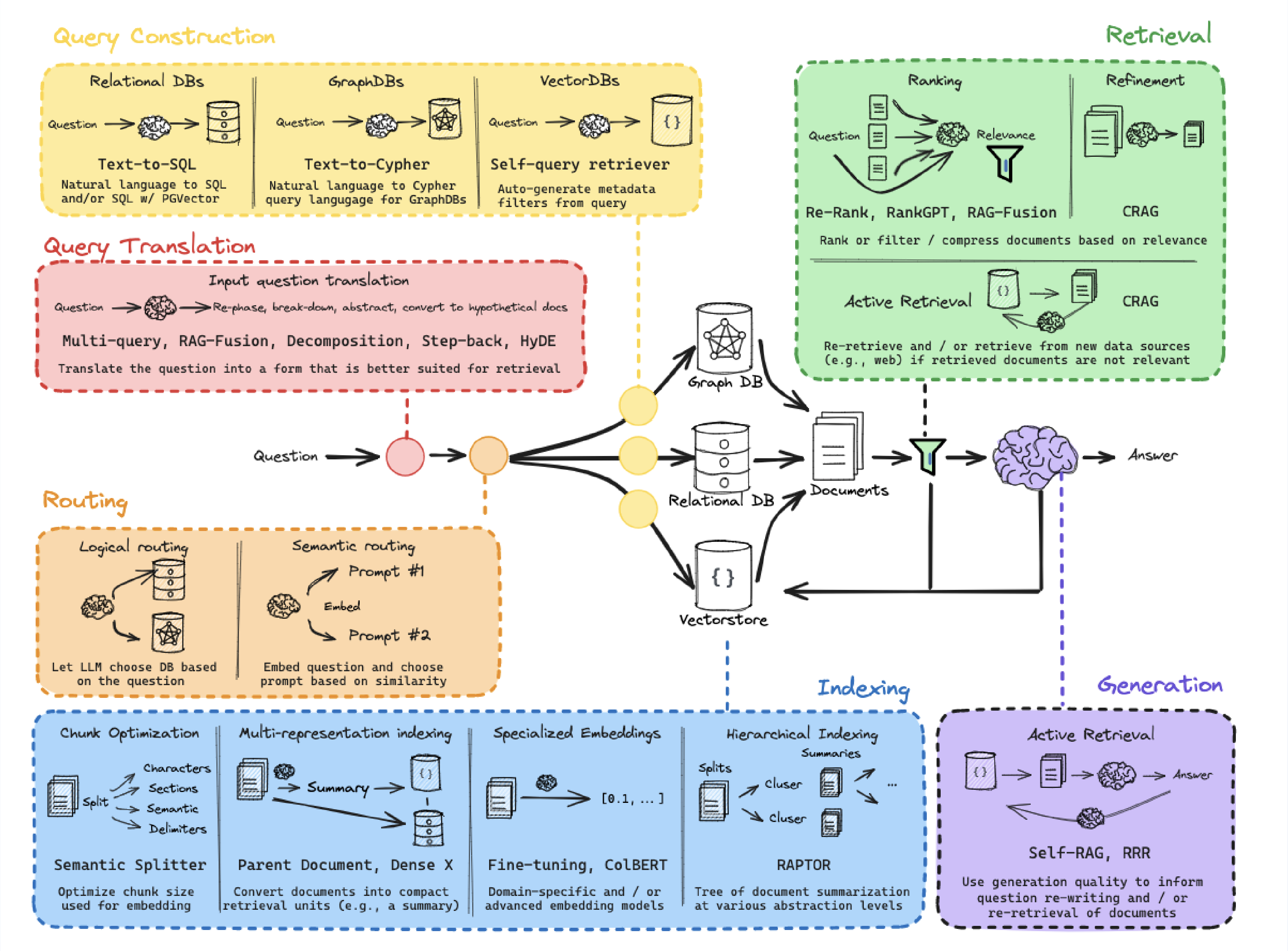
Ludwig-Maximilians-Universität München

Motivation

RAG was developed to create a more robust and reliable system for knowledge-intensive NLP tasks by integrating the strengths of both parametric (the pre-trained model) and non-parametric (the retrievable knowledge base) memory. (Lewis et al., 2020)

- **Addressing Knowledge Gaps:** The performance of large pre-trained language models lags behind task-specific architectures on knowledge-intensive tasks. RAG addresses this by combining a pre-trained retriever with a pre-trained sequence-to-sequence model to provide external, up-to-date information.
- **Overcoming Static Knowledge:** The world knowledge of pre-trained models is fixed at the time of their training. RAG allows for updating the knowledge base by simply replacing the document index, without needing to retrain the entire model.
- **Providing Provenance:** It is difficult to understand the reasoning behind the decisions of large language models. RAG offers a degree of interpretability by showing which retrieved documents were used to generate a response.
- **Reducing Hallucinations:** Large language models may generate text that is factually incorrect, a phenomenon known as "hallucination". By grounding the generation process in retrieved documents, RAG can produce more factual and specific language.

RAG Pipeline

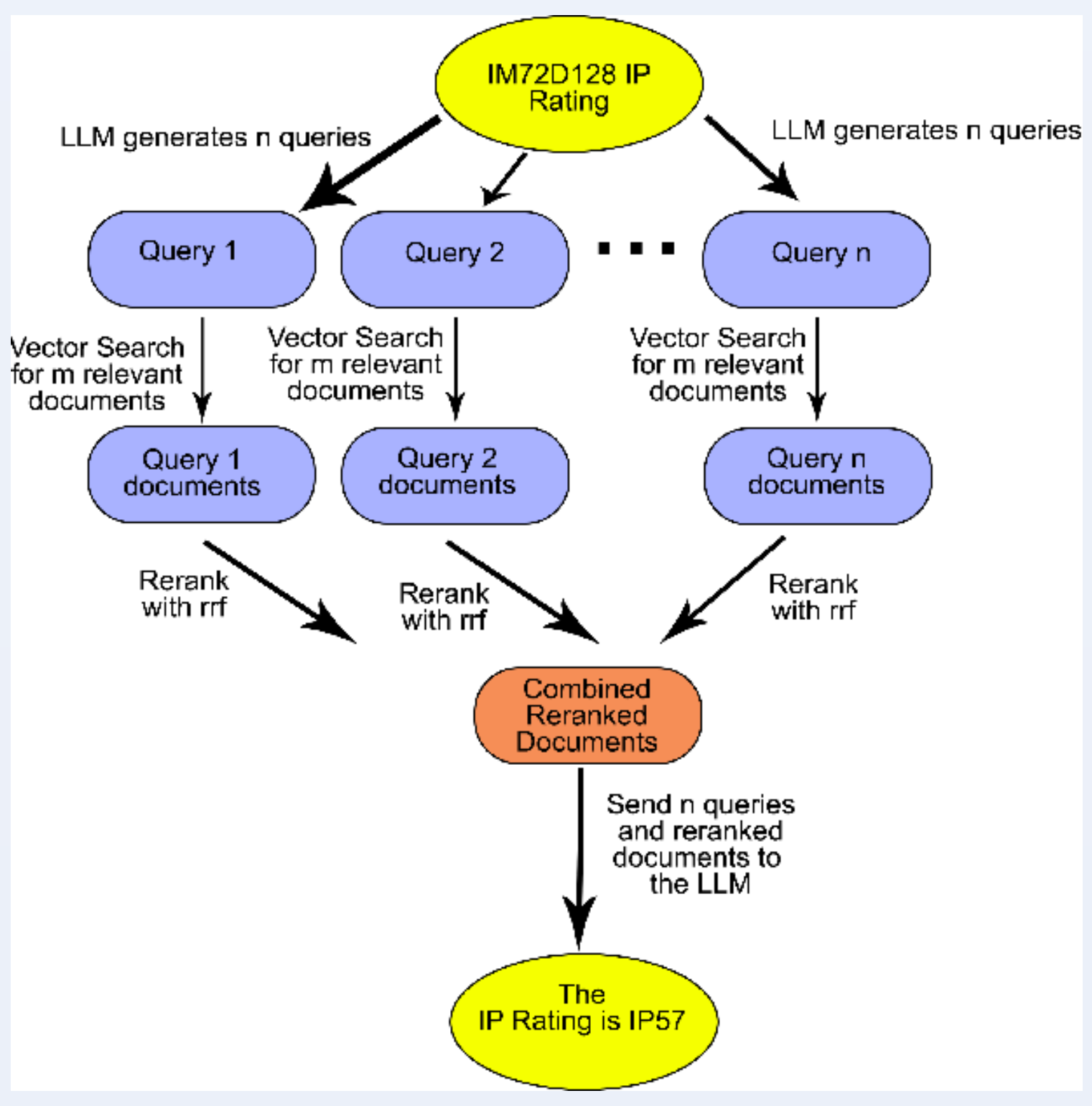

Query Construction
Relational DBs
Question
Text-to-SQL
Natural language to SQL and/or SQL w/ PGVector
GraphDBs
Question
Text-to-Cypher
Natural language to Cypher query langugage for GraphDBs
VectorDBs
Question
Self-query retriever
Auto-generate metadata filters from query
Query Translation
Input question translation
Question
Re-phase, break-down, abstract, convert to hypothetical docs
Multi-query, RAG-Fusion, Decomposition, Step-back, HyDE
Translate the question into a form that is better suited for retrieval
Routing
Logical routing
Let LLM choose DB based on the question
Semantic routing
Prompt #1
Embed
Prompt #2
Embed question and choose prompt based on similarity
Question
Graph DB
Relational DB
Vectorstore
Documents
Answer
Retrieval
Ranking
Question
Relevance
Re-Rank, RankGPT, RAG-Fusion
Refinement
CRAG
Rank or filter / compress documents based on relevance
Active Retrieval
CRAG
Re-retrieve and / or retrieve from new data sources (e.g., web) if retrieved documents are not relevant
Indexing
Chunk Optimization
Split
Characters
Sections
Semantic
Delimiters
Semantic Splitter
Optimize chunk size used for embedding
Multi-representation indexing
Summary
Parent Document, Dense X
Convert documents into compact retrieval units (e.g., a summary)
Specialized Embeddings
[0.1, ...]
Fine-tuning, ColBERT
Domain-specific and / or advanced embedding models
Hierarchical Indexing
Summaries
Splits
Cluser
Cluser
...
RAPTOR
Tree of document summarization at various abstraction levels
Generation
Active Retrieval
Answer
Self-RAG, RRR
Use generation quality to inform question re-writing and / or re-retrieval of documents

Source: “RAG Decomposition”, an open source project provided by LangChain

Query Translation Methods

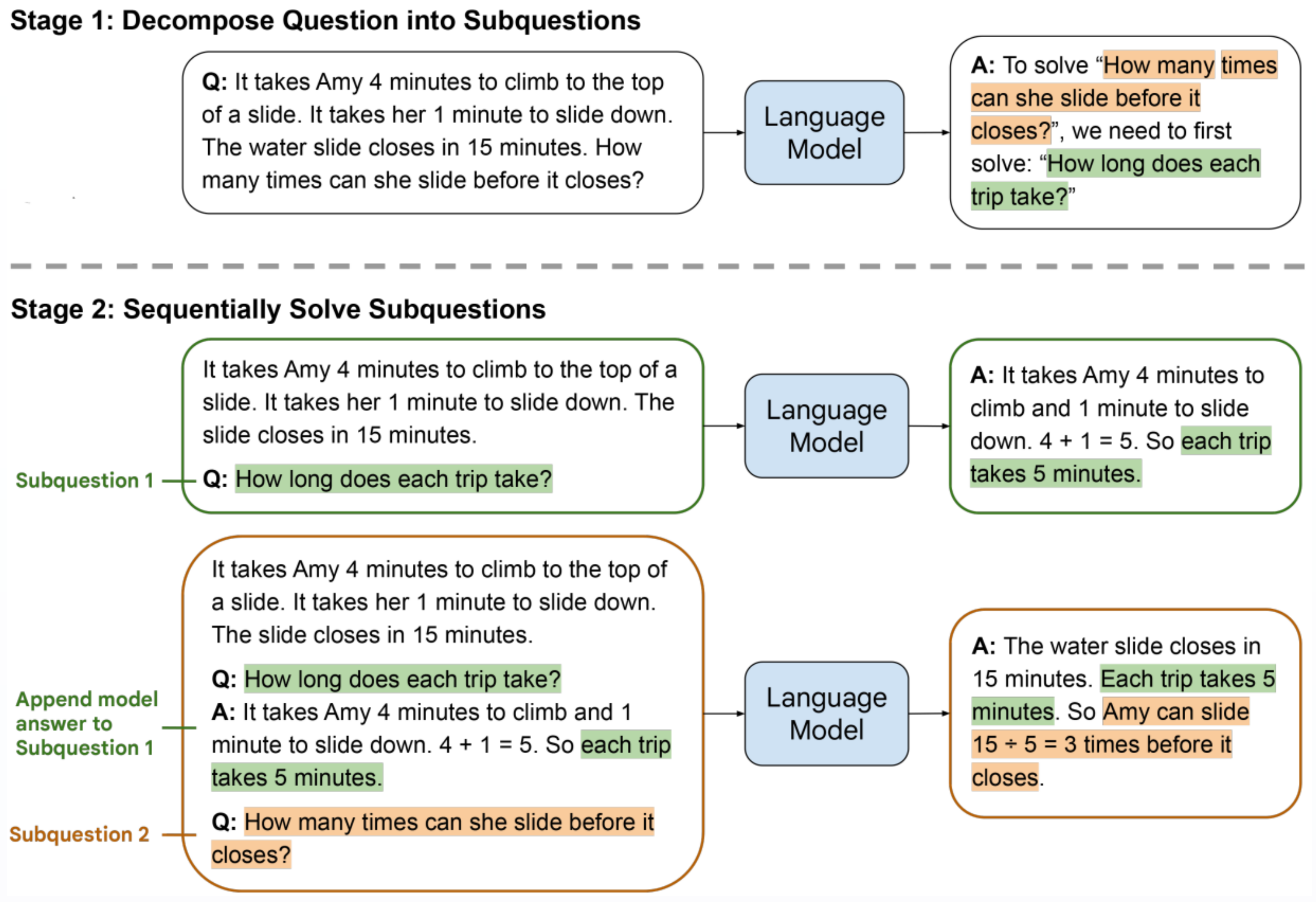
Ambiguous queries lead to poor document retrieval, causing LLMs to hallucinate answers based on irrelevant information. Therefore, the goal for this step is to transfer the user query to improve retrievals.

To address this issue Rackauckas (2024) proposed RAG-Fusion. The idea is to first generate multiple queries from the user query using LLM, conduct parallel retrieval, and then start reciprocal rank fusion (RRF) that is to select documents that consistently appear in the top results across multiple queries.



Source: RAG-Fusion: A New Take on Retrieval-Augmented Generation. (Rackauckas, Z. 2024)

Alternatively, the query decomposition method was proposed by Google. It decompose problem into sub-problems and solve them sequentially.


Stage 1: Decompose Question into Subquestions
Language Model
Stage 2: Sequentially Solve Subquestions
Subquestion 1
Language Model
Append model answer to Subquestion 1
Subquestion 2
Language Model

Q: It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The water slide closes in 15 minutes. How many times can she slide before it closes?

A: To solve “How many times can she slide before it closes?”, we need to first solve: “How long does each trip take?”

It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The slide closes in 15 minutes.
Q: How long does each trip take?

A: It takes Amy 4 minutes to climb and 1 minute to slide down. 4 + 1 = 5. So each trip takes 5 minutes.

It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The slide closes in 15 minutes.
Q: How long does each trip take?
A: It takes Amy 4 minutes to climb and 1 minute to slide down. 4 + 1 = 5. So each trip takes 5 minutes.
Q: How many times can she slide before it closes?

A: The water slide closes in 15 minutes. Each trip takes 5 minutes. So Amy can slide 15 ÷ 5 = 3 times before it closes.

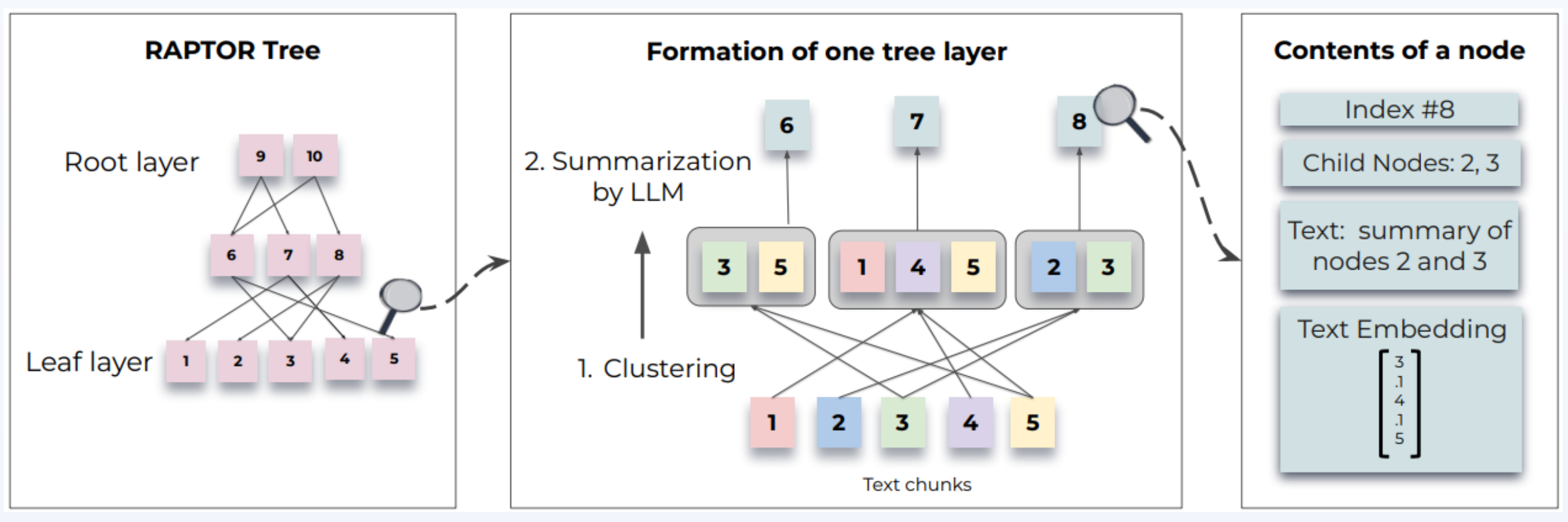
Source: Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. (Zhou et al., 2022)

Indexing Method

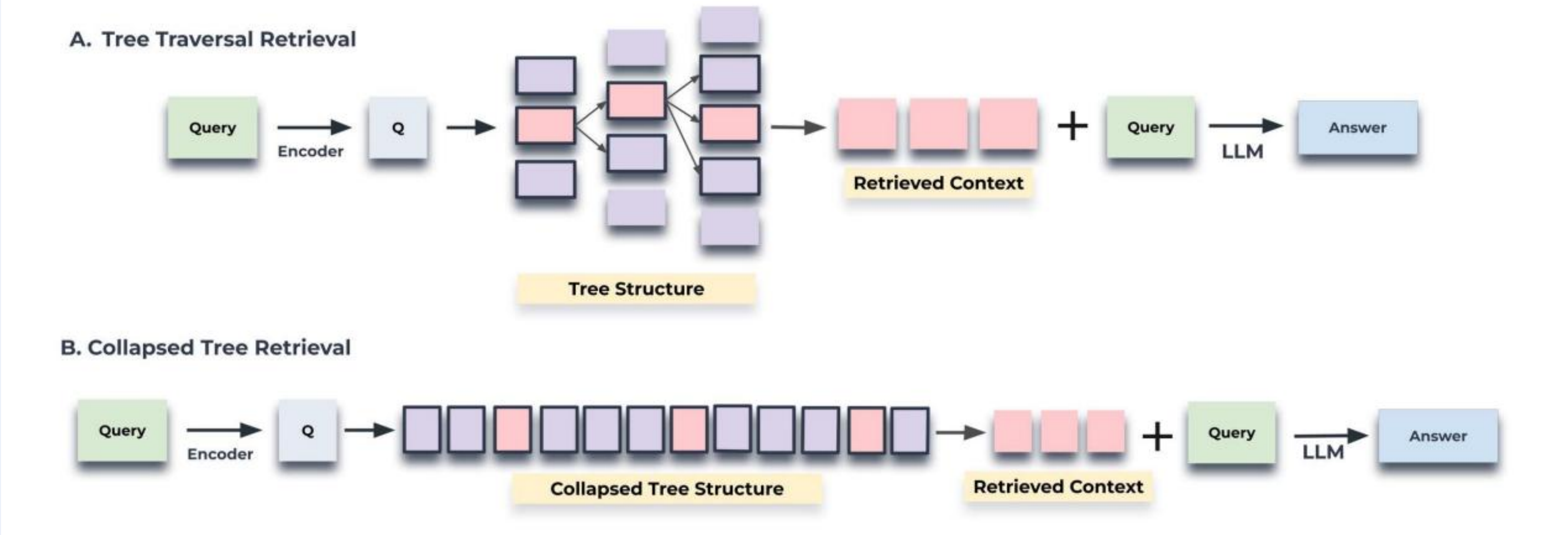
The goal for indexing is to transform external documents to retrievers which is responsible for fetching the most relevant pieces of data given a user query. In simple words, it uses embedding to transform both the user query and documents to vectors and then use method like KNN to select the ones that are close to the query.

However, one issue occurs which is what if the number of required retrievals exceed the number of K specified in the simple KNN model? Sarthi (2024) proposed the idea of Recursive Abstractive Processing for Tree-Organized Retrieval (PATOR).

The tree is constructed by first clustering the raw documents (leaf layer) and generate summaries for each cluster using LLM. This process is conducted recursively until a certain pre-defined depth or a single cluster is reached.



There are two retrieval mechanisms: A. Tree traversal starts at the root level of the tree and retrieves the top-k (here, top-1) node(s) based on cosine similarity to the query vector.
B. Collapsed tree collapses the tree into a single layer and retrieves nodes until a threshold number of tokens is reached. The benefit of this is both the detailed documents and high-level summaries are indexed together. This means if a higher-level question is asked, it will retrieve more higher level of layers, and vice versa.



Source: RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval.

References

- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.
- Rackauckas, Z. (2024). RAG-Fusion: A New Take on Retrieval-Augmented Generation.
- Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q., & Chi, E. (2022). Least-to-Most Prompting Enables Complex Reasoning in Large Language Models.
- LangChain. (2023, October 20). *Semi-structured & multi-modal RAG*. LangChain Blog. https://blog.langchain.com/semi-structured-multi-modal-rag/
- Sarthi, P., Abdullah, S., Tuli, A., Khanna, S., Goldie, A., & Manning, C. D. (2024). RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval.