Retrieval-Augmented Generation

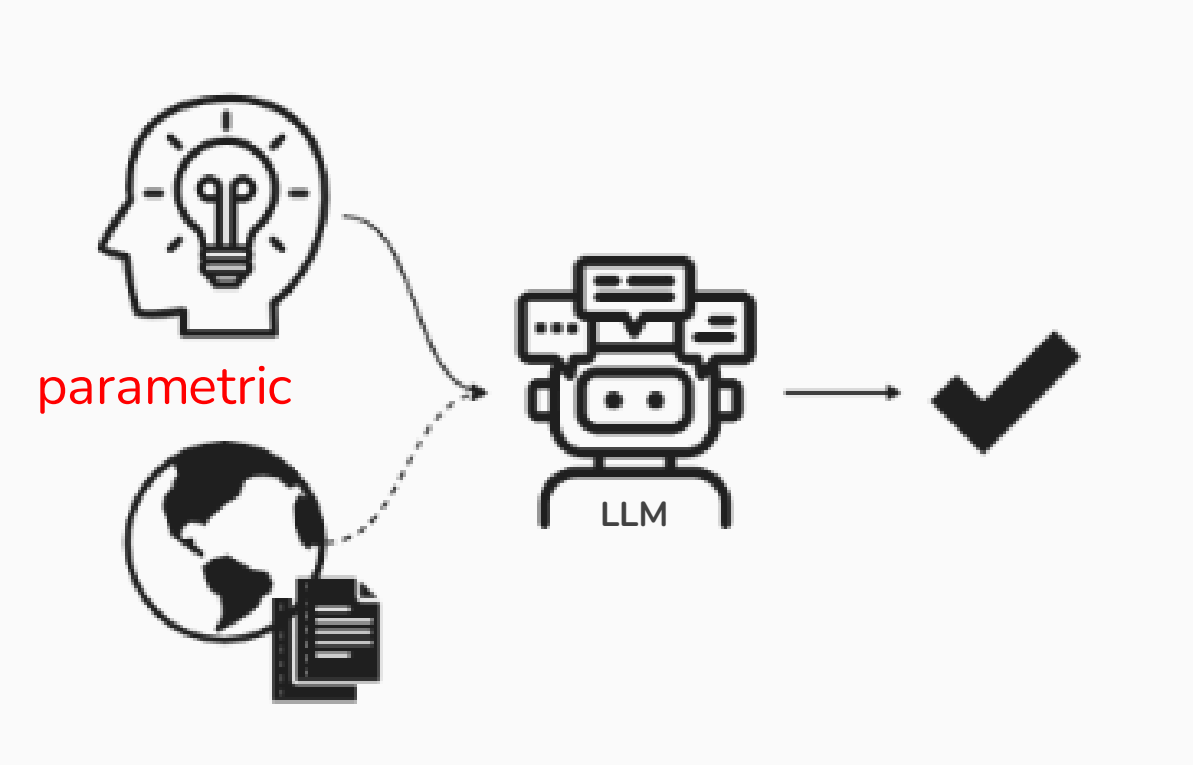
By Aliza Samad & Sofia Lendahl

Outline

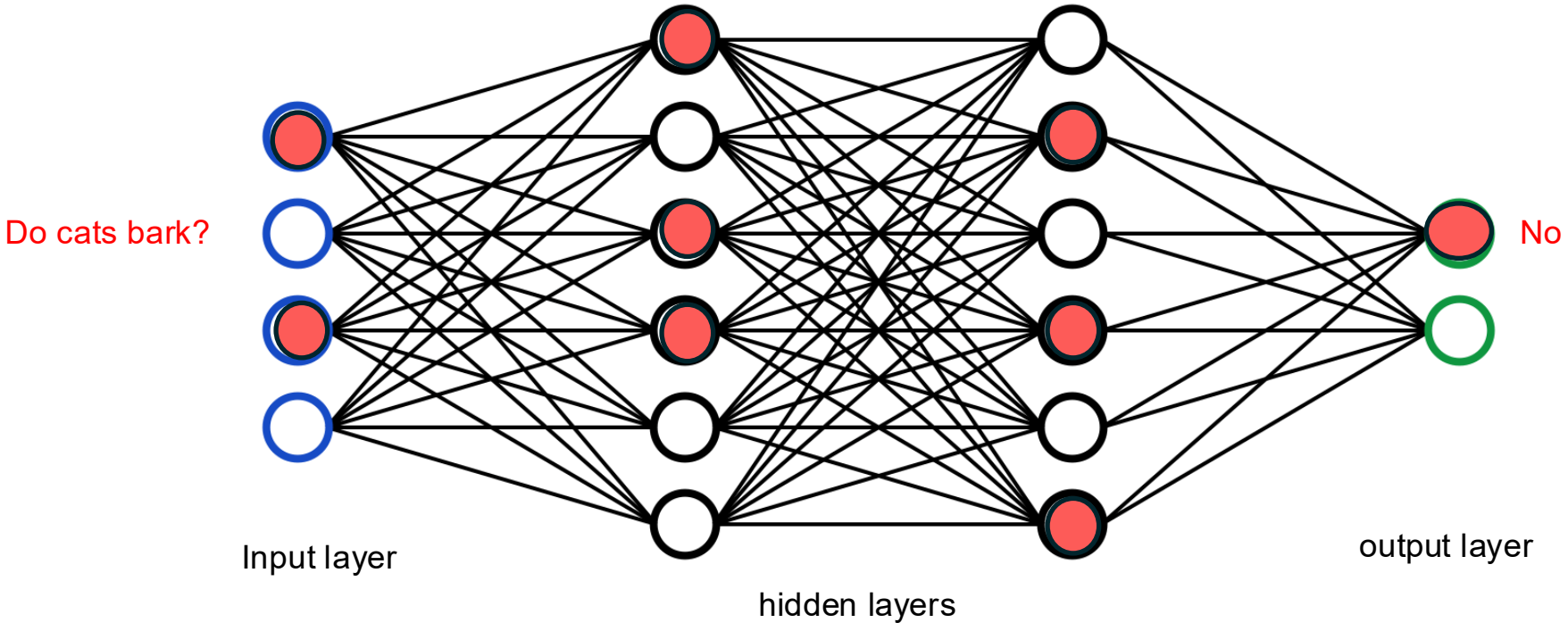
1. General Overview
2. Research Timeline
3. Optimal Parameters
4. Evaluation Metrics
5. Next Steps

RAG General Overview

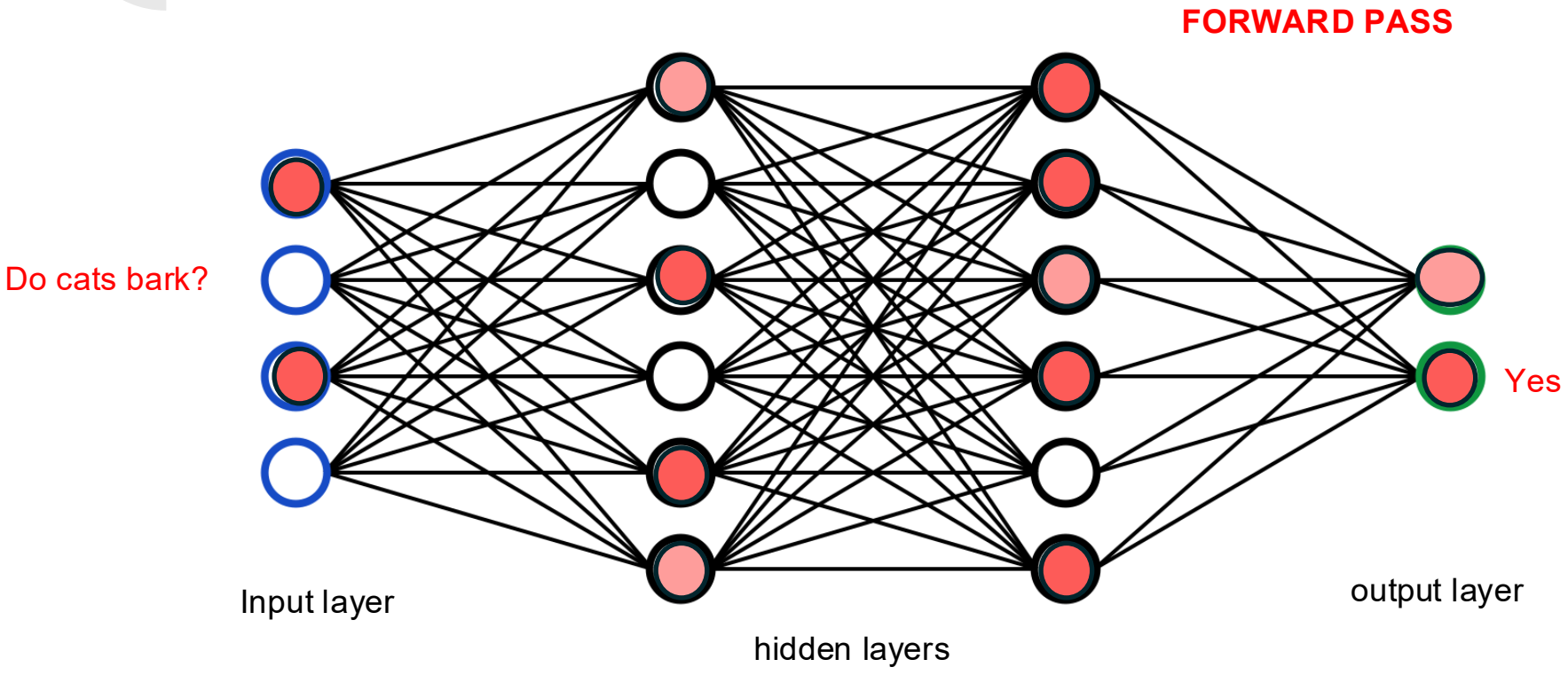
What is Retrieval-Augmented Generation?



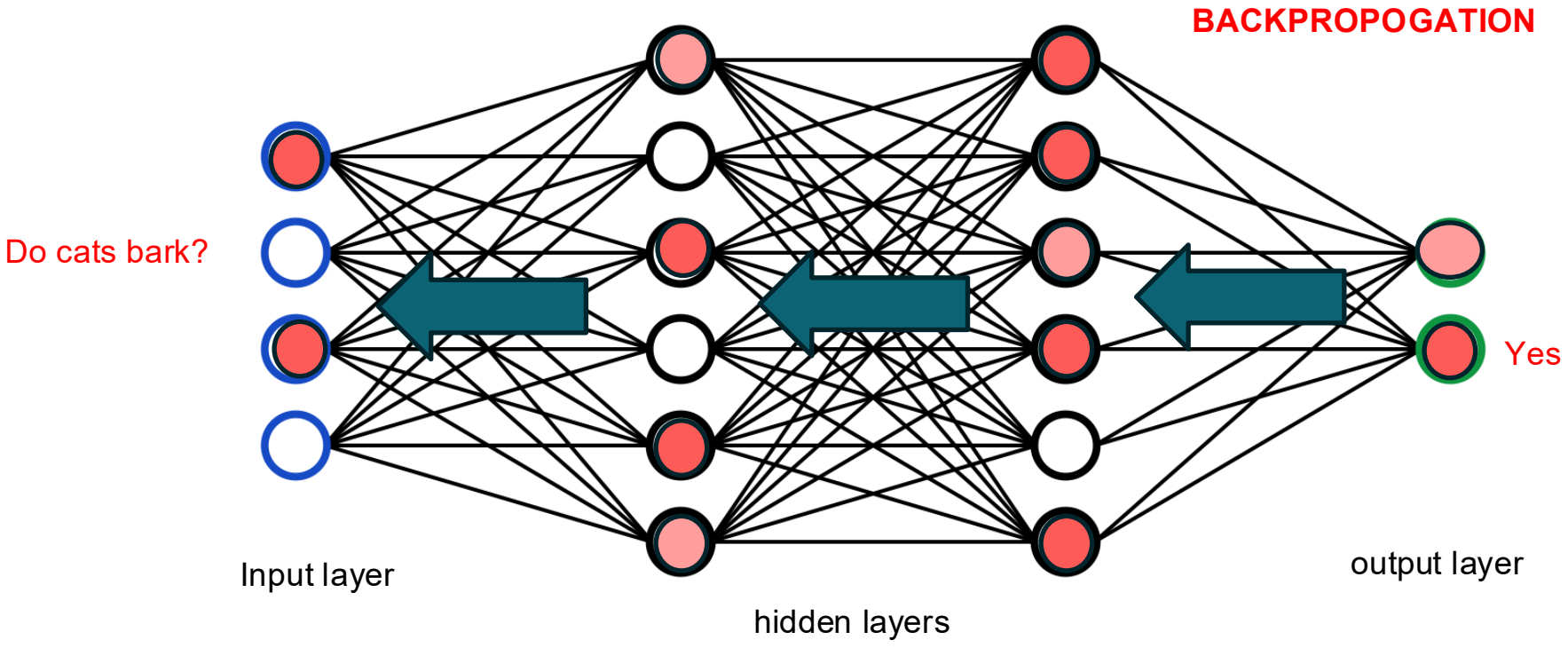
Neural Networks



Training Neural Networks

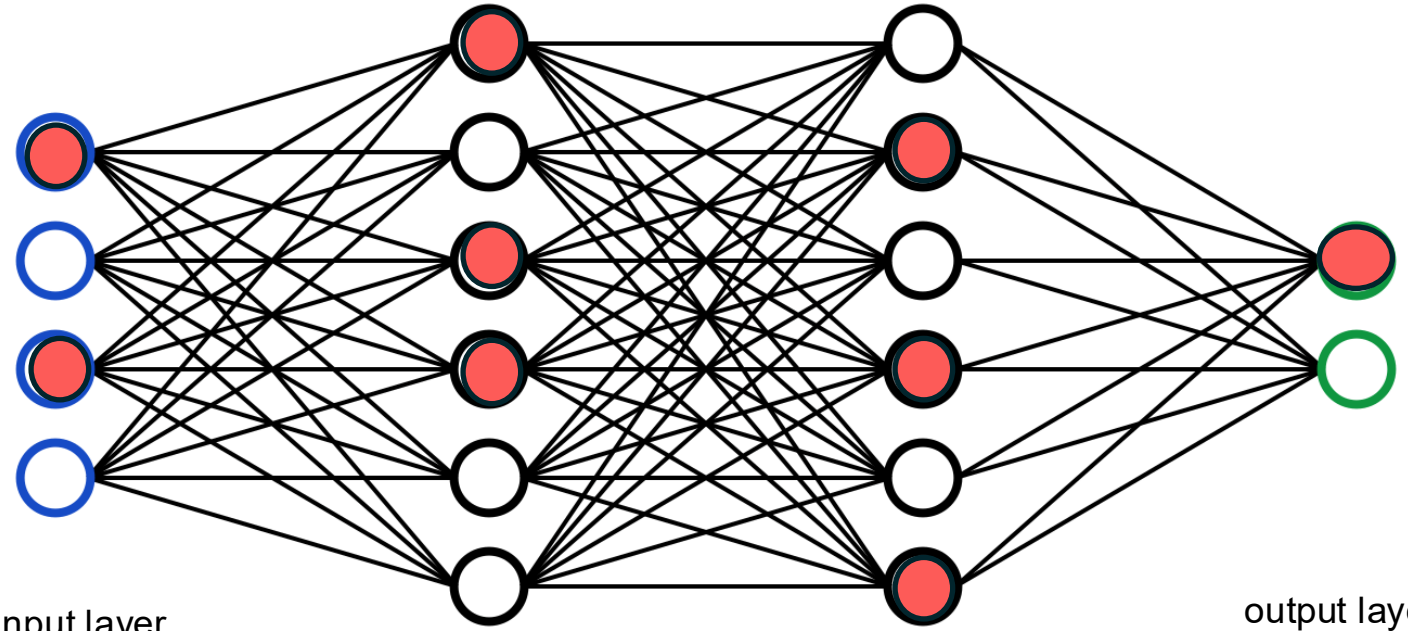


Training Neural Networks

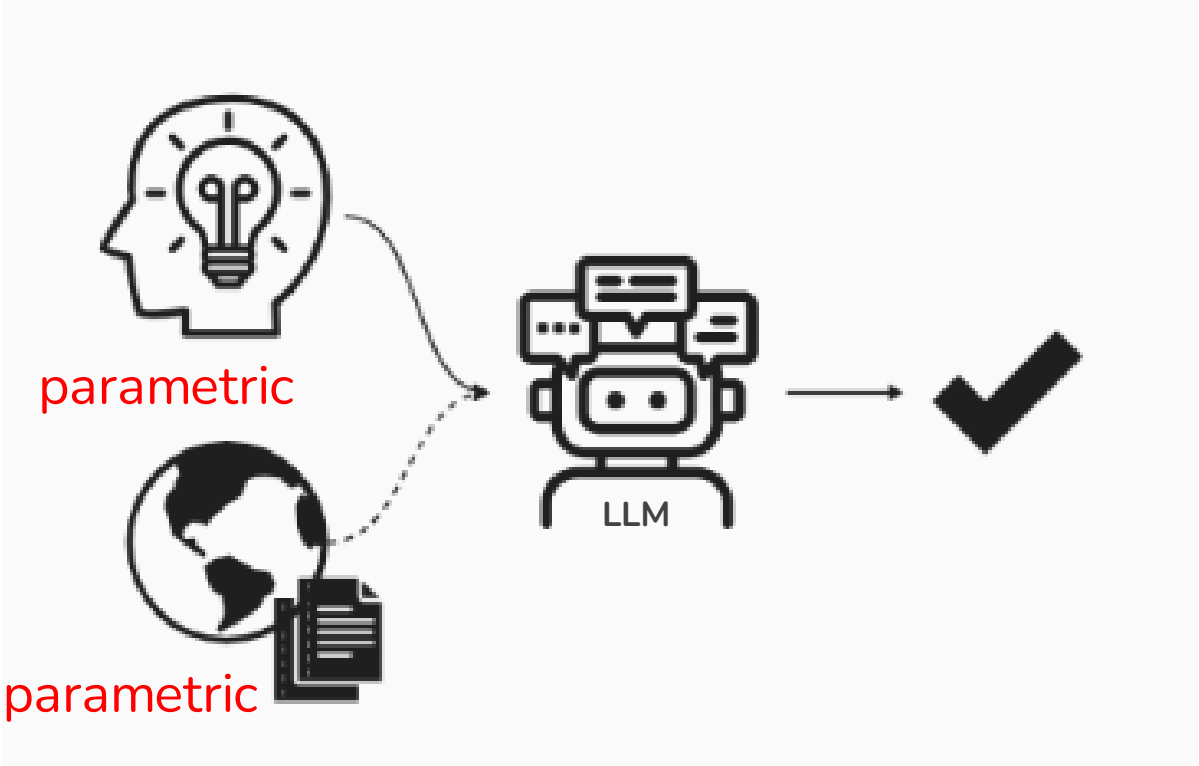


Training Neural Networks

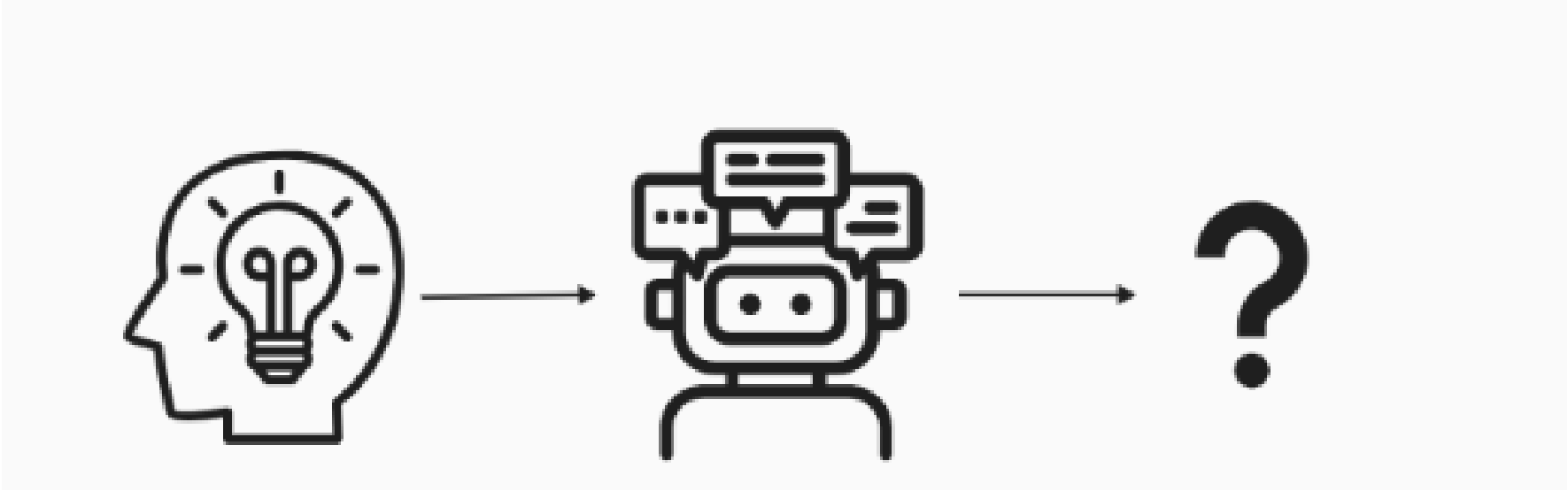
SEVERAL ITERATIONS LATER



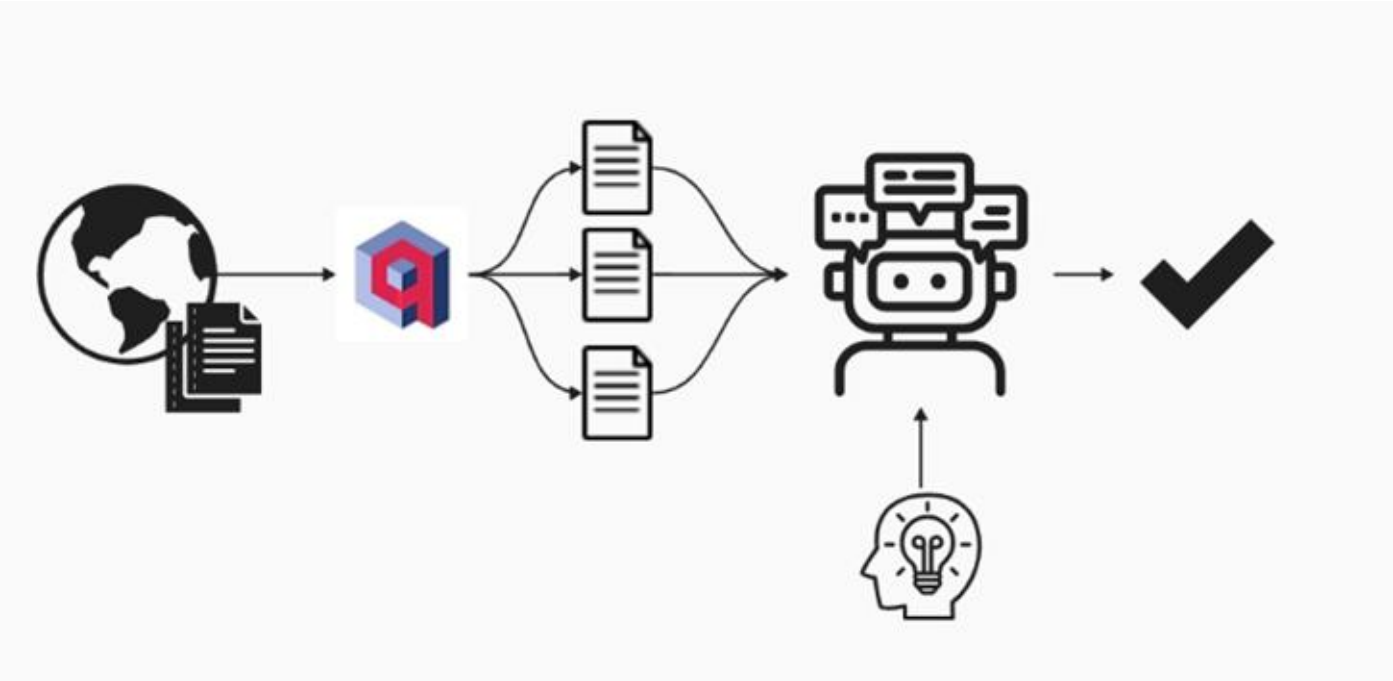
What is RAG?



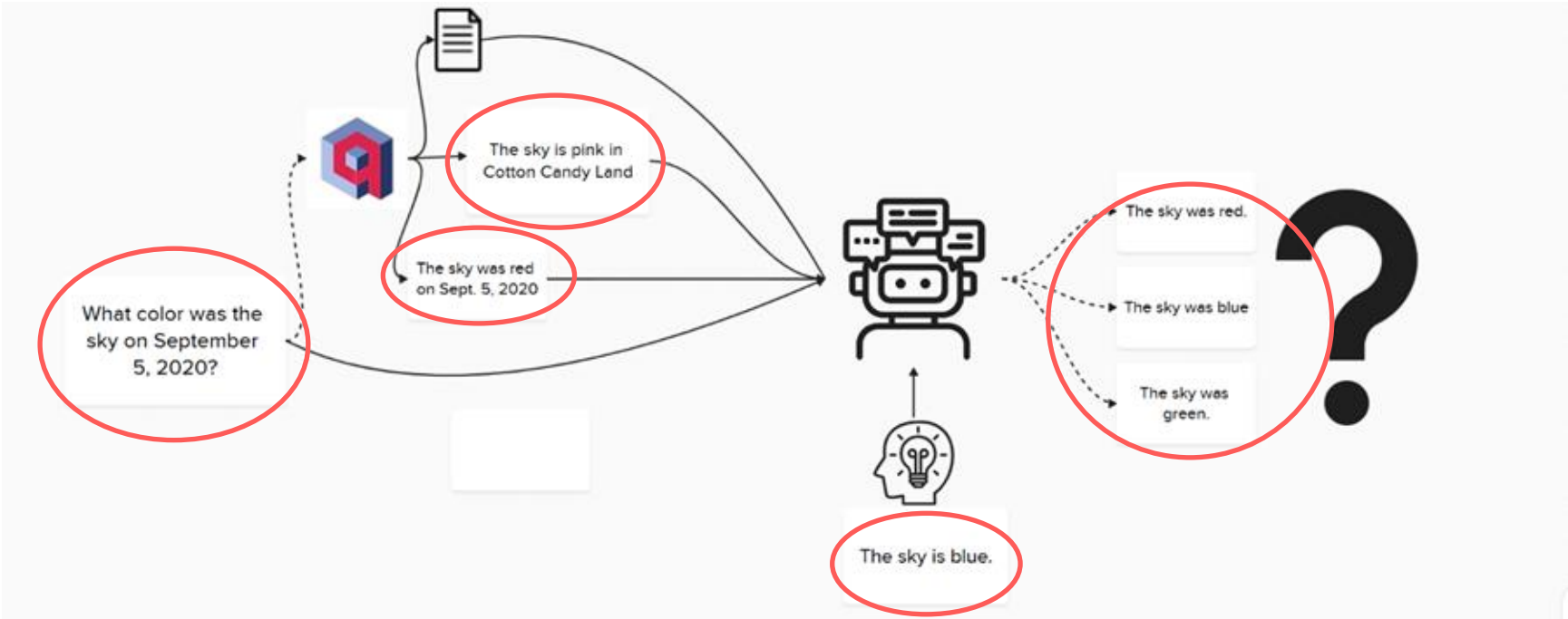
What Problems Does RAG Address?



What Solutions Does RAG Offer?

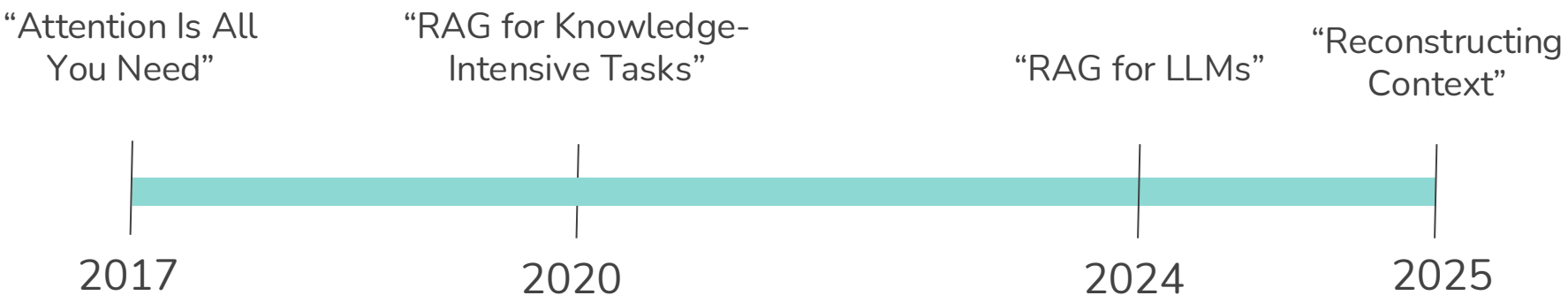


Current Limitations

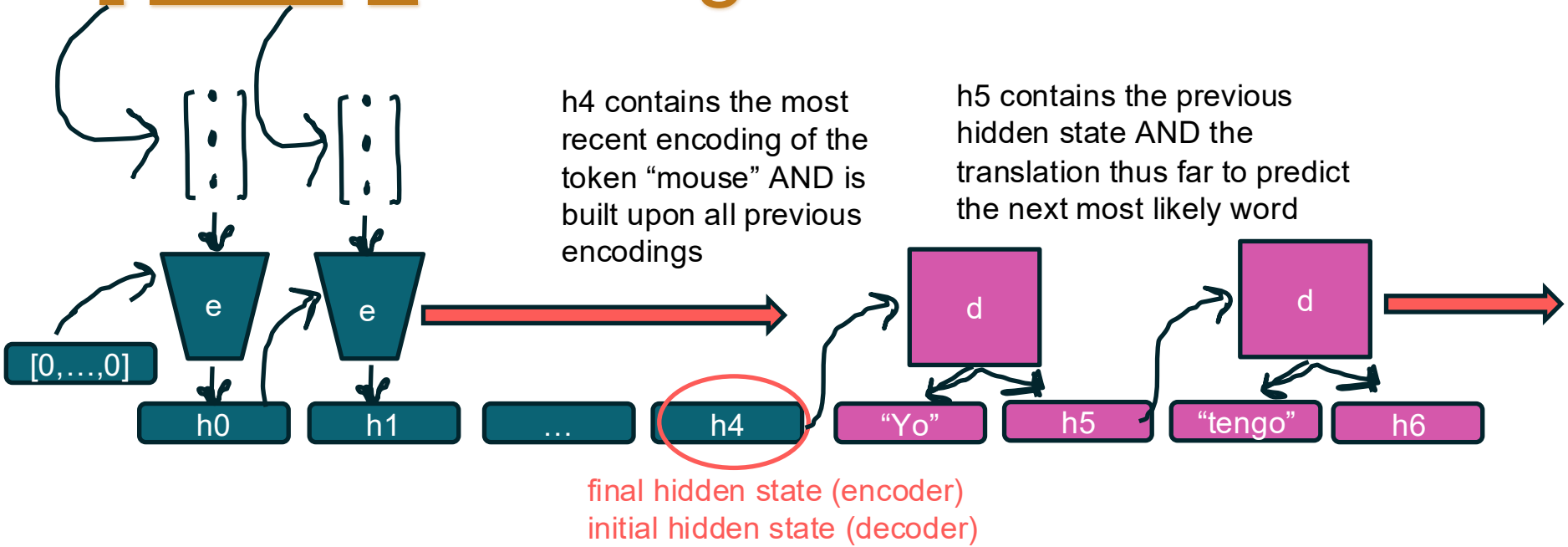


RAG Research Timeline

Timeline Overview



“I have a red dog”



“Yo tengo un perro rojo”

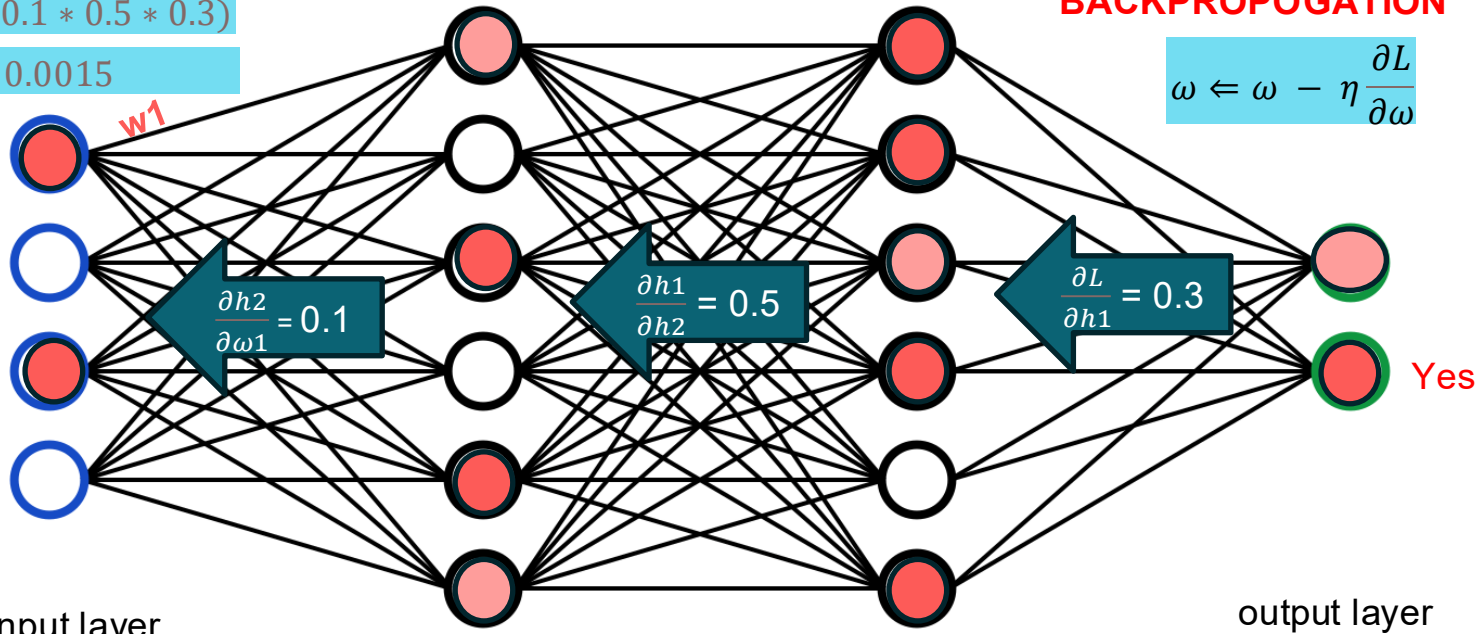
The “Vanishing Gradient” Problem

$$\omega \Leftarrow \omega - 0.01 * (0.1 * 0.5 * 0.3)$$

$$\omega \Leftarrow \omega - 0.0015$$

BACKPROPOGATION

$$\omega \Leftarrow \omega - \eta \frac{\partial L}{\partial \omega}$$



Do cats bark?

Yes

Input layer

hidden layers

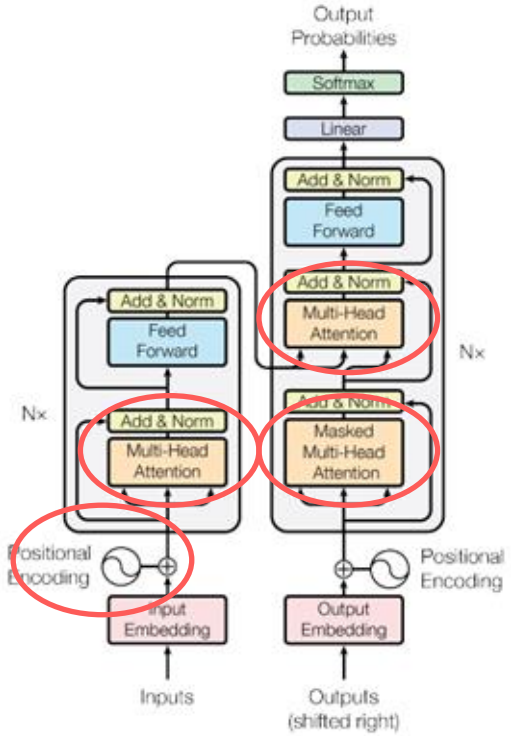
output layer

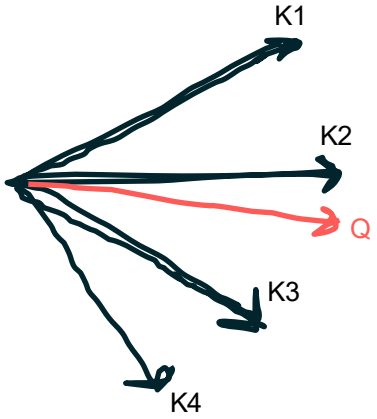
“Vanishing Gradient” & RNNs

“I” “I have” “I have a” “I have a red” “I have a red dog”

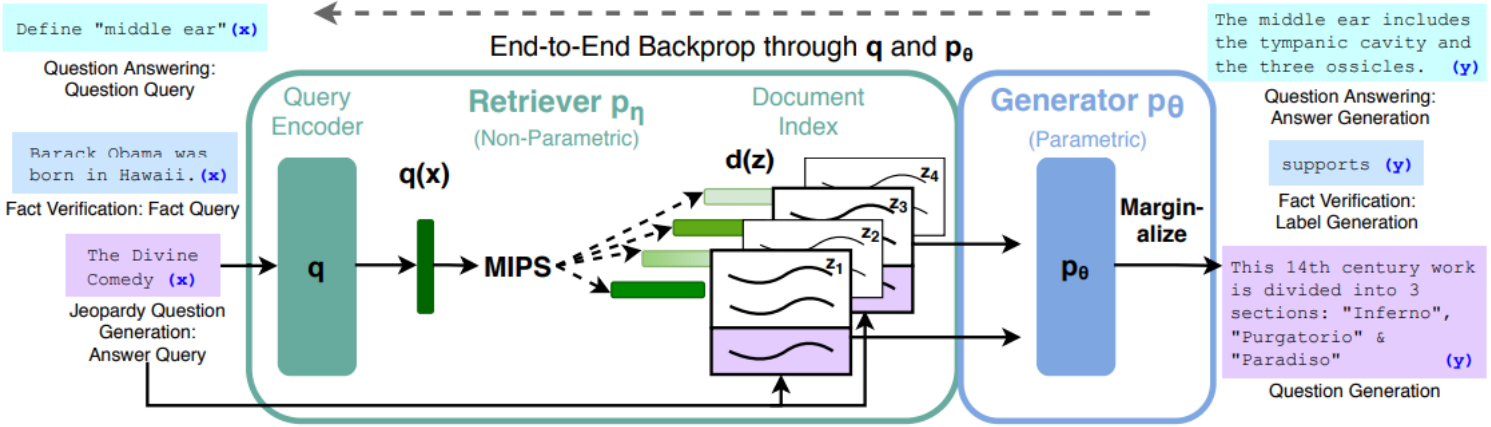
Ten? Tengo? Tienes? Tenemos??

“Attention is All You Need” (2017)





“RAG for Knowledge-Intensive Tasks” (2020)



Generation Models:

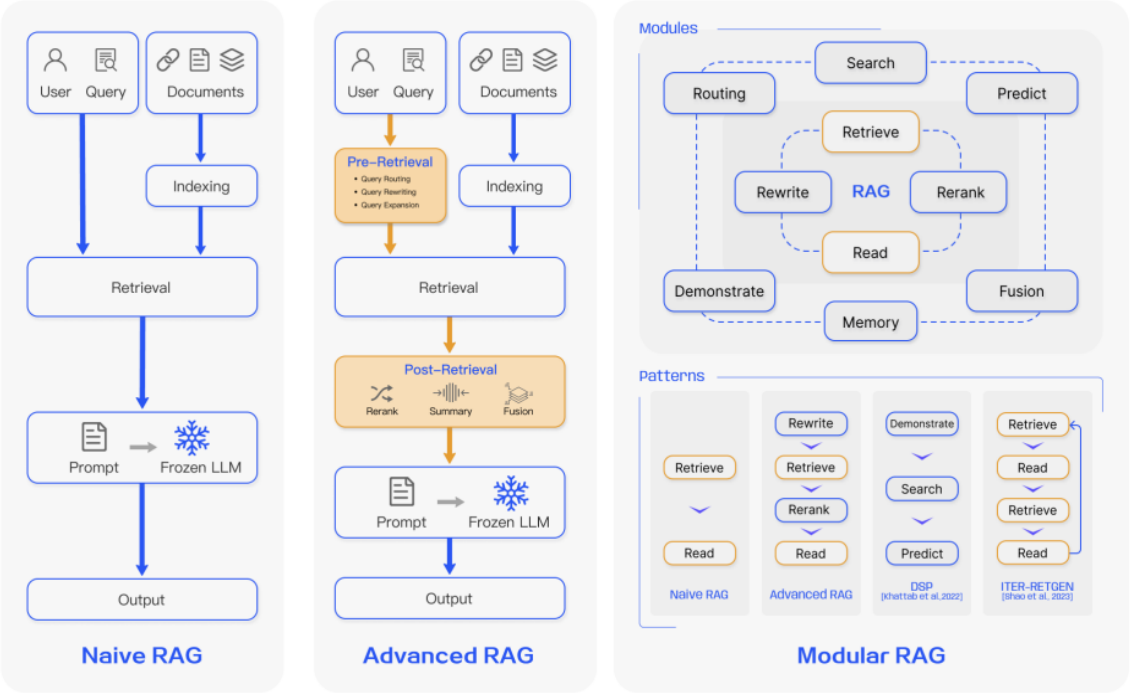
- Rag-Sequence: same document to generate the complete sequence
- Rag-Token: different documents to generate each token

“RAG for Knowledge-Intensive Tasks” (2020)

Observations:

- Documents with clues about the answer but do not contain the exact answer can still contribute towards a correct answer being generated
- Model performance varies for specific tasks
- RAG models obtain state of the art results on open-domain QA
- RAG is more factual and specific than parametric models such as BART

"RAG for LLMs" (2024)



“Reconstructing Context” (2025)

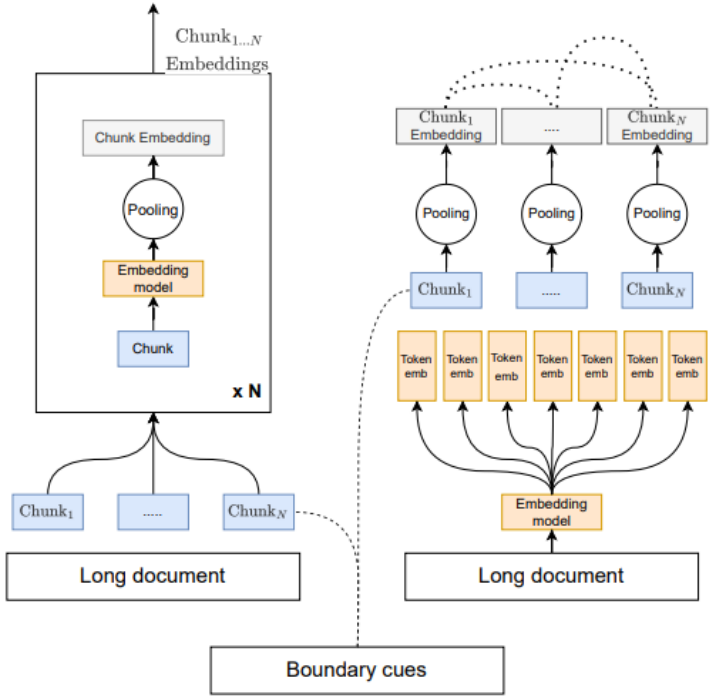
Late Chunking

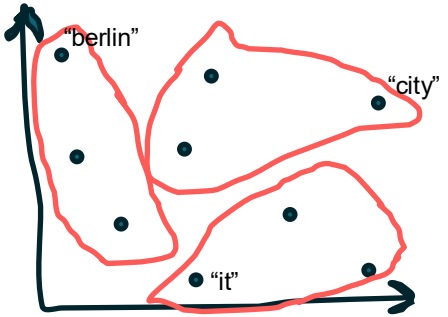
- Annotate the document with where you want to chunk
- Embed full document (or as much as possible) via long-context embedder
- Chunk documents by embeddings

Contextual Chunking

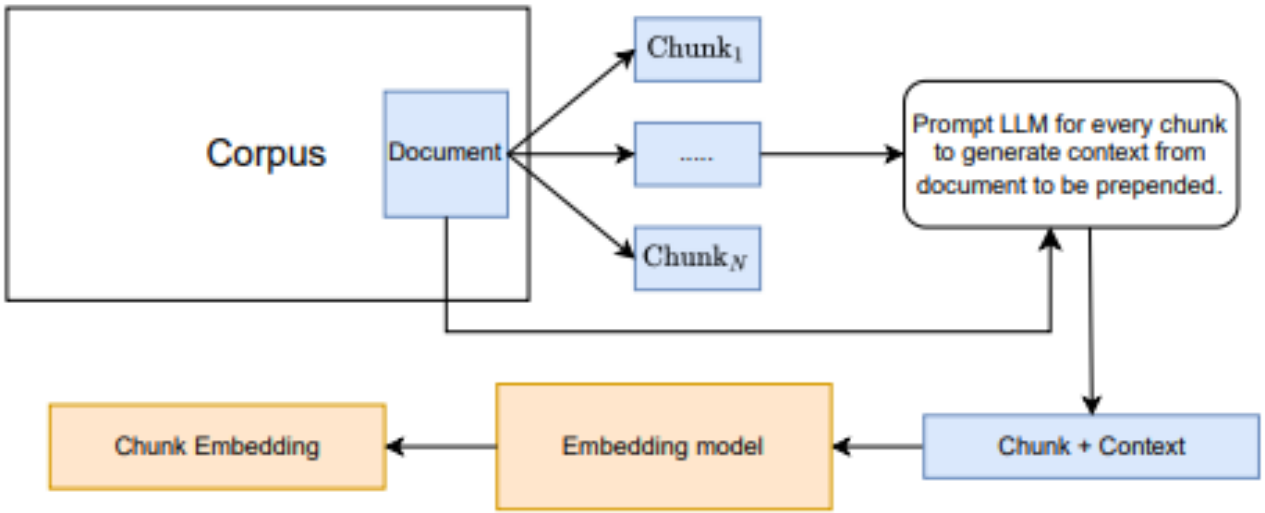
- Contextualization: use an LLM to contextualize each chunk in context of doc
- Rank fusion:
  - final_fused_score = 1.0 * dense_score + 0.25 * bm25_score
- Reranking: cross-encode cross-encode chunks (top 50 from final_fused_score) and rerank

“Reconstructing Context” (2025)





“Reconstructing Context” (2025)



Optimizing RAG Parameters

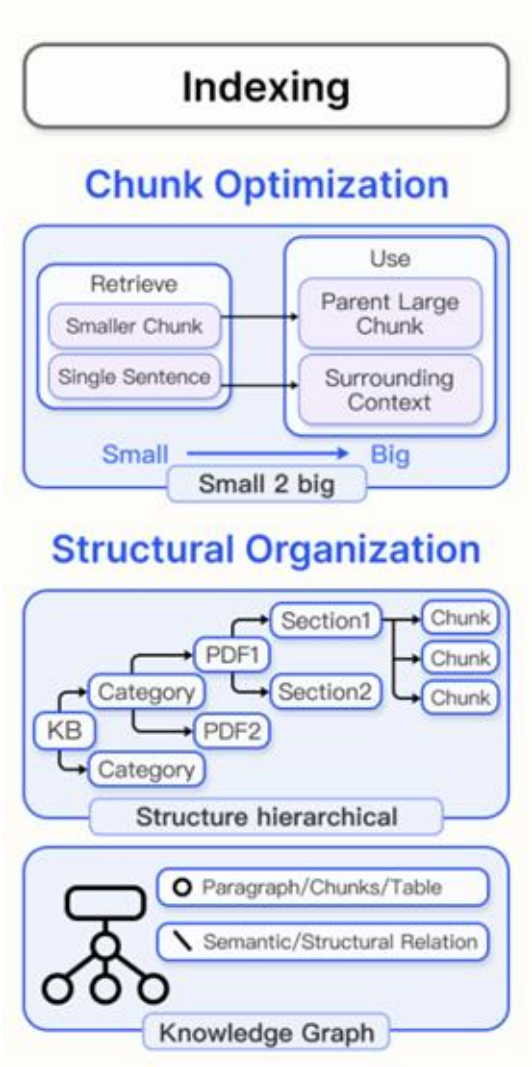
Goal

Balanced Performance and Efficiency :

Achieve a balance between performance and efficiency for OCHCO Chat users.

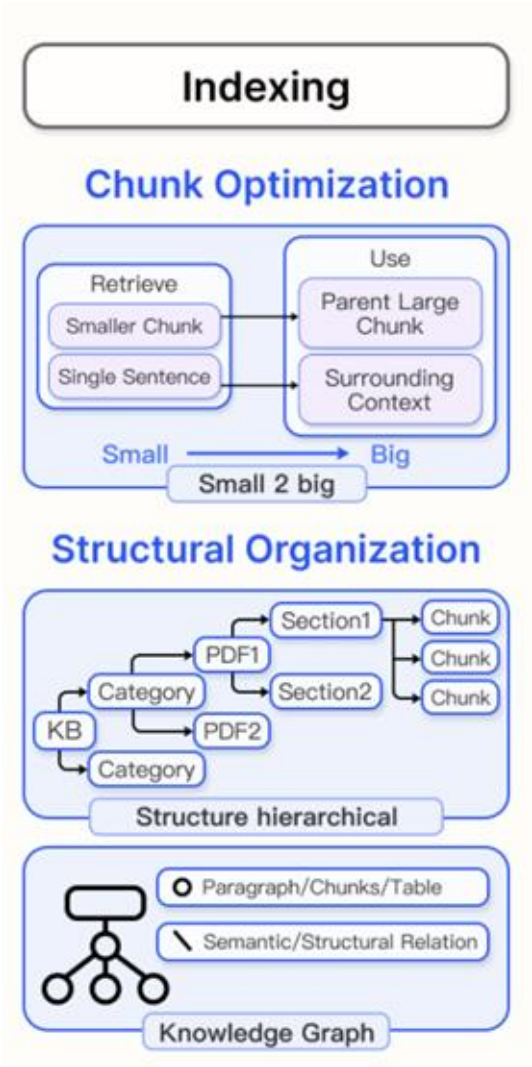
Evaluate parameters in 6 main RAG modules:

Indexing, Pre-retrieval, Retrieval, Post-Retrieval, Generation, and Orchestration



Chunking Methods:

- **Naive Fixed Length Chunking:** chunks independent & of pre-determined length
- **Small-to-Big:** retrieve smaller summarized chunks and reference their parent larger chunks
- **Sliding Window:** last few tokens from previous chunk overlap with beginning few of following chunk
- **Late-Chunking:** embed entire document then chunk embeddings
- **Contextual Chunking:** use an LLM to generate summary of chunk in context of entire doc & embed chunk with summary
- **Element-Based Chunking:** parse document and use JSON elemental map to guide chunking

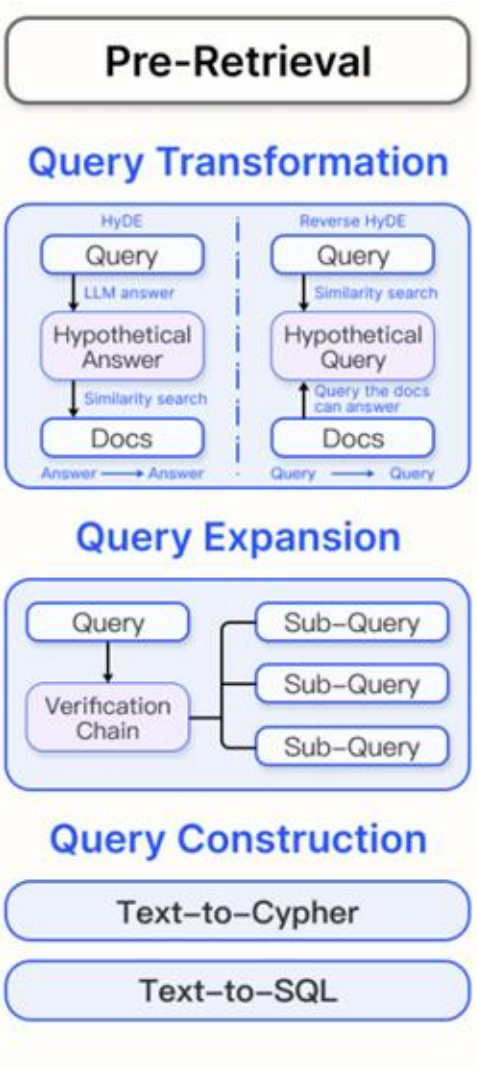


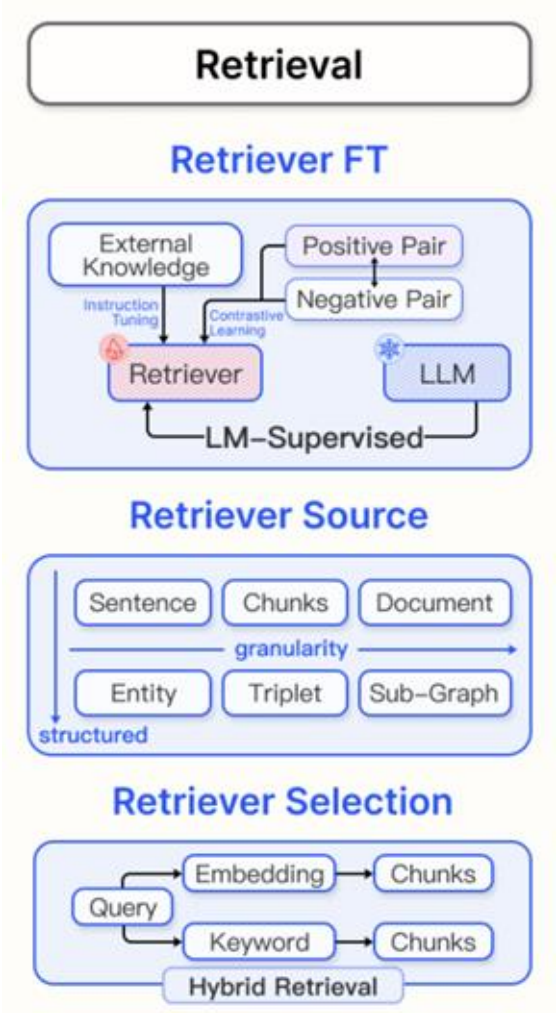
Structure:

- **Hierarchical Index:** documents arranged in parent child relationships with chunks linked to them
- **KG Index:** Knowledge Graphs (KGs) to structure documents

Query Processing:

- **Query Rewriting:** specialized smaller models to rewrite query
- **Query Expansion:** break down query into subqueries
- **HyDE:** constructs hypothetical documents (assumed answers) when responding to queries
- **Query Construction:** Text-to-SQL or Text-to-Cypher to accommodate data types





Retriever Selection

- **Hybrid Retriever**: uses both sparse (exact word match) and dense retrievers (semantic match) simultaneously ($\alpha = 0.3$)

Retriever Fine-tuning

- **Adapter:** a small, trainable set of new parameters inserted into a pre-trained LLM's architecture, allowing the model to adapt its performance to specific downstream tasks without altering the original model weights

Reranking

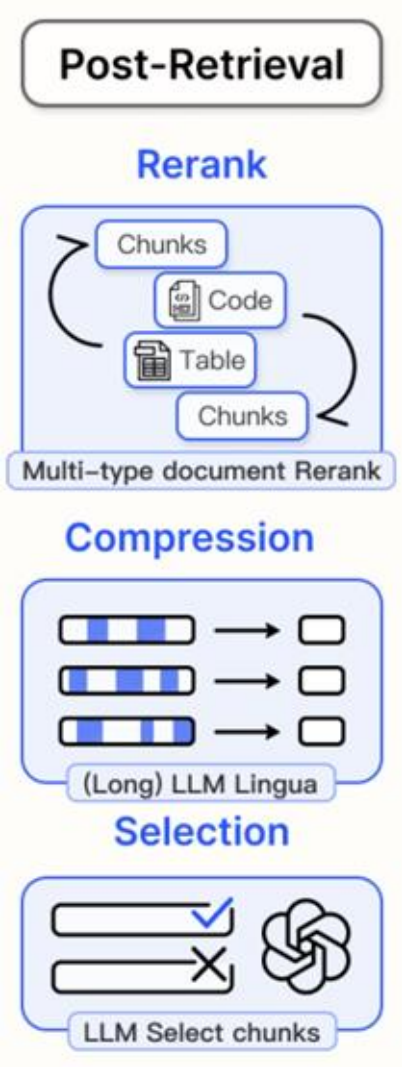
- **TILDE Reranking:** reorders retrieval results using a BERT-based sparse model that estimates relevance by modeling the importance of query terms in documents

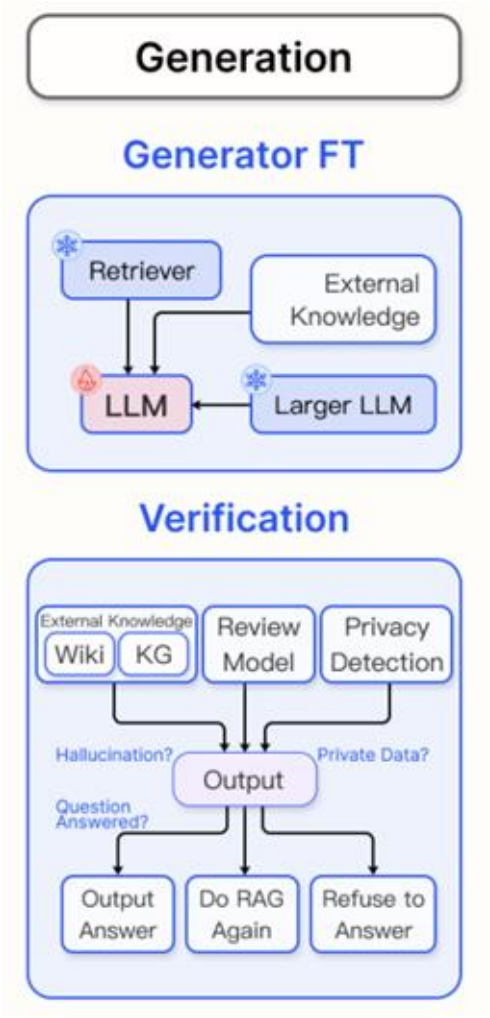
Compression

- **(Long)LLMLingua:** detects and removes of unimportant tokens from the prompt
- **Recomp (abstractive):** synthesizes information from multiple documents

Document Repacking

- **Reverse:** arranges documents in ascending order in terms of their relevance



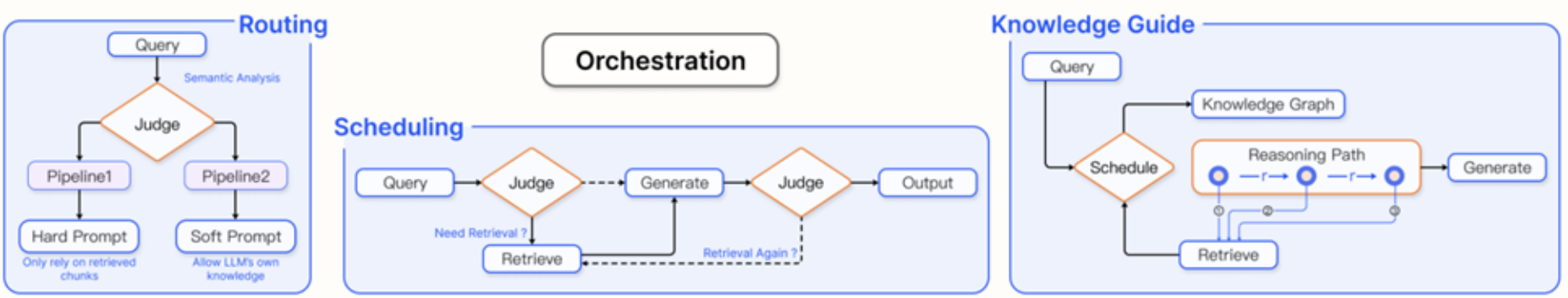


Generator Fine-tuning

- **Instruct-Tuning**: provide LLM additional knowledge
- **Reinforcement learning:** aligning LLM outputs with human or retriever preferences
- **Dual Fine-tuning:** fine-tuning both generator and retriever simultaneously

Verification

- **Knowledge-base verification:** validate LLM responses through external knowledge
- **Model-based verification**: use a small language model to verify the responses generated by LLMs



Routing

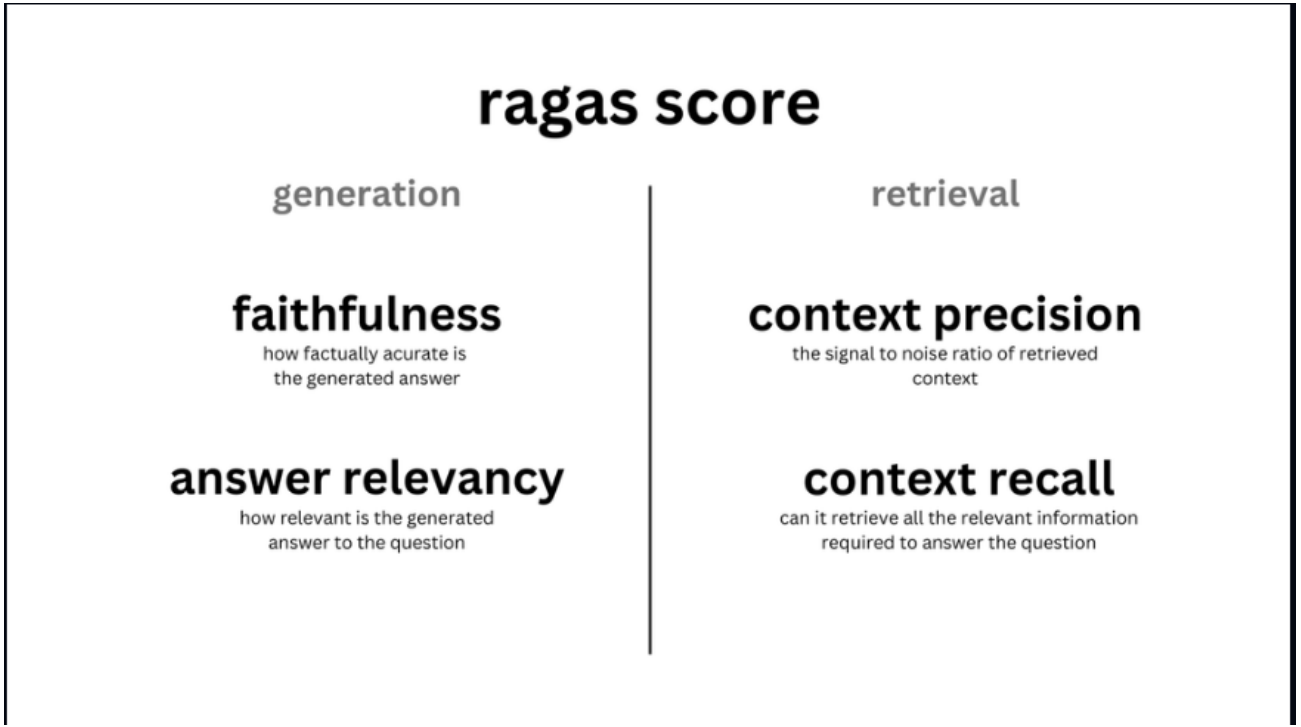
- **Metadata routing:** extracting key terms, or entities, from the query
- **Semantic routing:** routes to different modules based on the semantic information
- **Hybrid Routing:** integrates semantic analysis and metadata-based approaches

Scheduling

- **Rule judge:** system evaluates the quality of answers through scoring mechanisms
- **LLM judge:** LLM independently determines the subsequent course of action

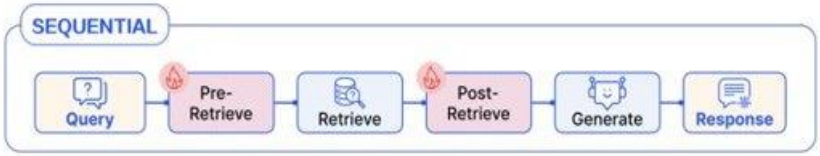
Evaluation Metrics

RAGAS


ragas score
generation
retrieval
faithfulness
how factually acurate is the generated answer
answer relevancy
how relevant is the generated answer to the question
context precision
the signal to noise ratio of retrieved context
context recall
can it retrieve all the relevant information required to answer the question

Next Steps

Experiment Setup



Retrieval
- Late-chunking
- Sliding Window
- Contextual Chunking
- Element-Based Chunking
- Hybrid Search

Metrics
- precision@K / context precision
- recall@K / context recall
- MAP
- MRR
- nDCG
- F-score
- latency

Generation
- Reranking
- System prompt engineering
- Temperature
- Repacking

Metrics
- ROUGE (non-LLM based)
- BLEU (non-LLM based)
- Faithfulness
- Answer Relevancy
- latency

Experiment Setup

Indexing

1. late-chunking vs. contextual chunking vs elemental chunking vs OWUI → benchmark results
2. late-chunking + contextual chunking + elemental chunking vs OWUI → benchmark results
3. Based on benchmark results, try different combinations of chunking methods & select highest score

Note: to control for external factors utilize OWUI's in-built retrieval function & try to keep all other variables (ex: metadata tagging) the same

Pre-retrieval

1. Query expansion vs query rewriting vs OWUI → benchmark results & select highest score

Retrieval

1. Hybrid model 1 vs. Hybrid model 2 (etc.) vs OWUI → benchmark results & select highest score

Post-Retrieval

1. Reranking vs repacking vs OWUI → benchmark results & select highest score
2. Recomp vs OWUI → benchmark & select highest score

Note: control for potential confounding variables (chunking method, etc.)

Sources

- Attention Is All You Need (2017)
- Meta Research Paper (2020)
- Retrieval Augmentation Helps Reduce "Hallucinations" (2021)
- RAG for LLMs: A Survey (2023)
- Advancements in RAG Systems (2024)
- Retrieval Strategies (contextual retrieval) (2025)
- Late Chunking (2024)
- Best practices (2024)
- Hyperparameter Analysis (2025)
- Modular RAG (2024)
- Modular RAG + Hypster (check out 56:00)
- RAGAS