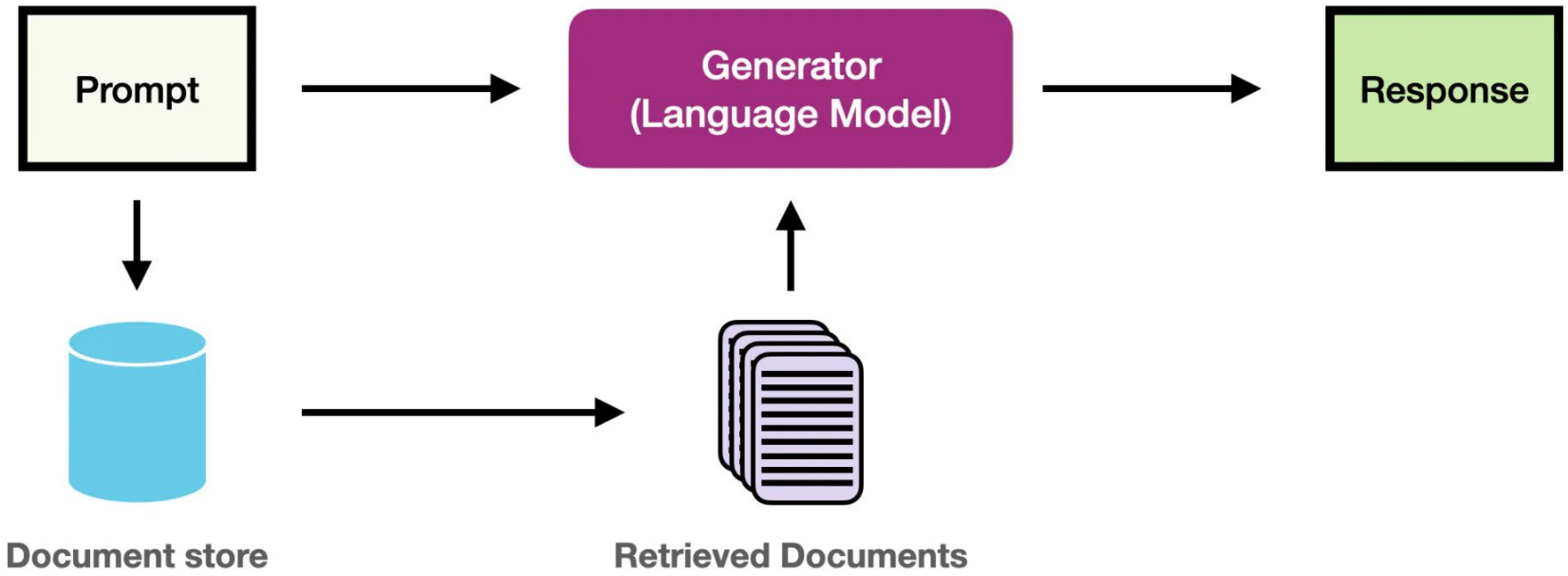
CSCE 689-609 Retrieval Augmented Generation

Jeff Huang
jeff@cse.tamu.edu
o2lab.github.io

Retrieval Augmented Generation (RAG)

- Intro: https://www.youtube.com/watch?v=Y08Nn23o_mY

Retrieval Augmented Generation



Sentence Embedding

A numeric representation of a sentence in the form of a **vector** of **real numbers** to encode meaningful **semantic information**

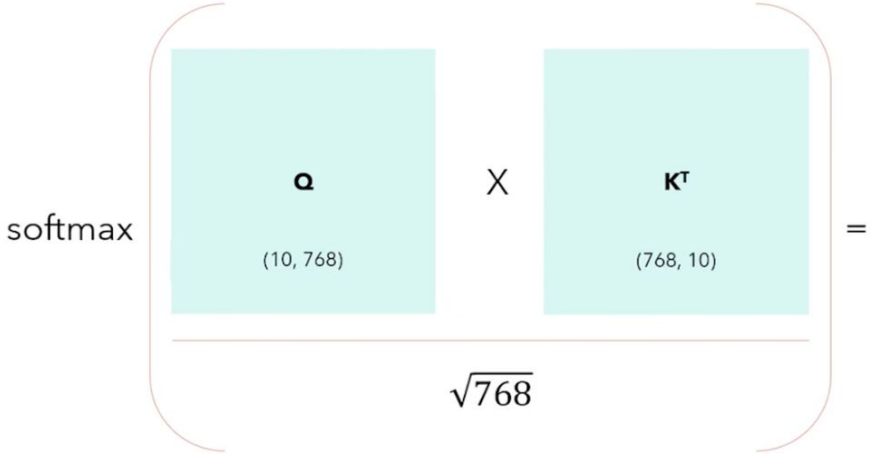
- **BERT (encoder-only)** uses the final hidden state vector of the **[CLS] token** to encode information about the sentence
  - In practice, however, BERT's sentence embedding with [CLS] achieves **poor performance**
  - Often worse than simply averaging non-contextual word embeddings
- **Sentence-BERT** improves performance of BERT by fine-tuning it using a siamese architecture
- **LLM (GPT decoder-only) embeddings**
  - GPT seems to outperform BERT for embedding quality
  - NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models
  - https://arxiv.org/pdf/2405.17428

Sentence Transformers: https://huggingface.co/sentence-transformers

[CLS] token in BERT

The **[CLS]** token always interacts with all the other tokens, as we do not use any mask.
So, we can consider the **[CLS]** token as a token that "captures" the information from all the other tokens.

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



	[CLS]	Before	my	bed	lies	a	pool	of	moon	bright
[CLS]	0.62	0.19	0.02	0.02	0.04	0.01	0.00	0.09	0.00	0.02
Before	0.15	0.00	0.00	0.01	0.00	0.00	0.17	0.00	0.67	0.00
my	0.09	0.02	0.56	0.02	0.01	0.08	0.11	0.02	0.05	0.03
bed	0.10	0.06	0.03	0.00	0.53	0.12	0.01	0.11	0.00	0.04
lies	0.02	0.00	0.00	0.05	0.80	0.00	0.02	0.04	0.01	0.06
a	0.01	0.00	0.02	0.02	0.00	0.03	0.68	0.16	0.03	0.06
pool	0.00	0.16	0.02	0.00	0.03	0.56	0.00	0.00	0.22	0.01
of	0.22	0.00	0.01	0.05	0.19	0.44	0.00	0.00	0.04	0.04
moon	0.00	0.67	0.01	0.00	0.02	0.03	0.23	0.01	0.00	0.03
bright	0.06	0.00	0.03	0.03	0.43	0.21	0.03	0.06	0.13	0.03

(10, 10)

Umar Jamil - https://github.com/hkproj/bert-from-scratch

[CLS] token: output sequence

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

	[CLS]	Before	my	bed	lies	a	pool	of	moon	bright
[CLS]	0.62	0.19	0.02	0.02	0.04	0.01	0.00	0.09	0.00	0.02
Before	0.15	0.00	0.00	0.01	0.00	0.00	0.17	0.00	0.67	0.00
my	0.09	0.02	0.56	0.02	0.01	0.08	0.11	0.02	0.05	0.03
bed	0.10	0.06	0.03	0.00	0.53	0.12	0.01	0.11	0.00	0.04
lies	0.02	0.00	0.00	0.05	0.80	0.00	0.02	0.04	0.01	0.06
a	0.01	0.00	0.02	0.02	0.00	0.03	0.68	0.16	0.03	0.06
pool	0.00	0.16	0.02	0.00	0.03	0.56	0.00	0.00	0.22	0.01
of	0.22	0.00	0.01	0.05	0.19	0.44	0.00	0.00	0.04	0.04
moon	0.00	0.67	0.01	0.00	0.02	0.03	0.23	0.01	0.00	0.03
bright	0.06	0.00	0.03	0.03	0.43	0.21	0.03	0.06	0.13	0.03

(10, 10)

X

V

(10, 768)

=

Attention Output

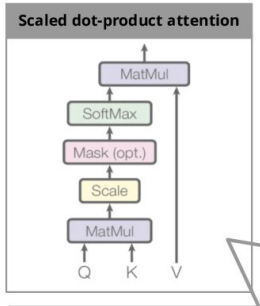
(10, 768)

Each row of the "Attention Output" matrix represents the embedding of the output sequence: it captures not only the meaning of each token, not only its position, but also the interaction of each token with all the other tokens, but only the interactions for which the softmax score is not zero. All the 512 dimensions of each vector only depend on the attention scores that are non-zero.

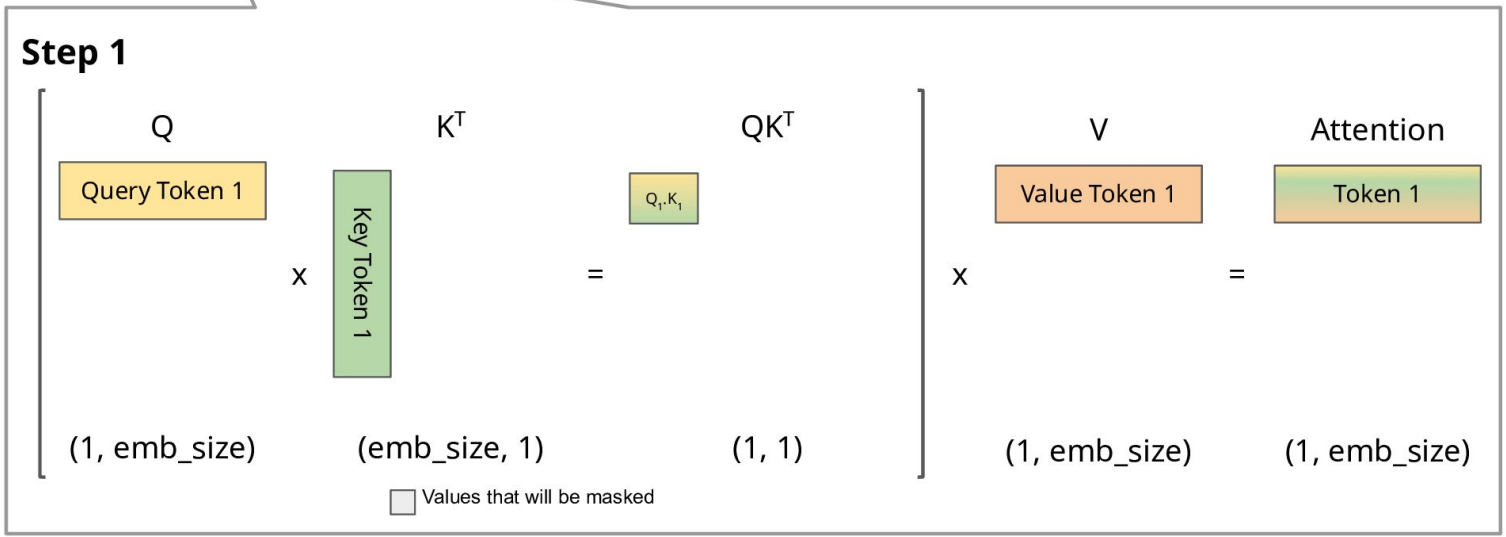
Umar Jamil - https://github.com/hkproj/bert-from-scratch

QKV

https://medium.com/@joaolages/kv-caching-explained-276520203249



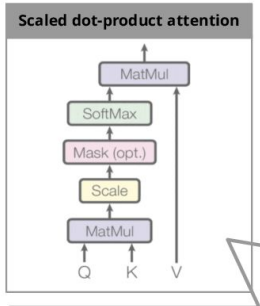
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



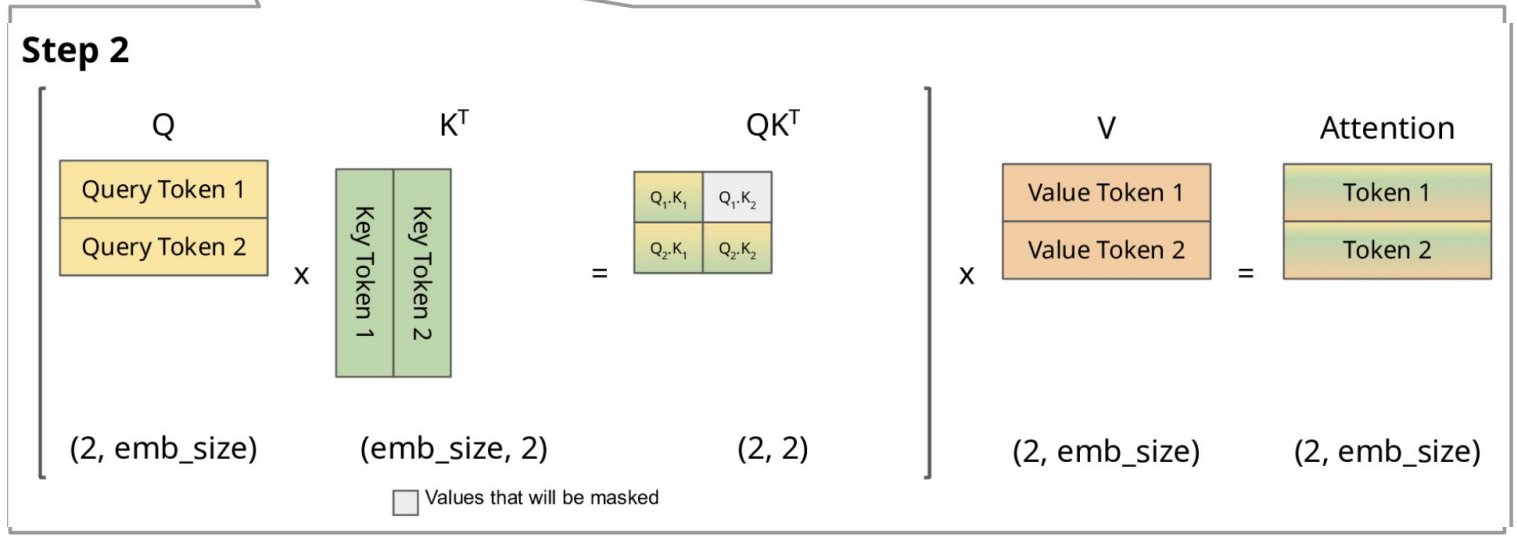
Zoom-in! (simplified without Scale and Softmax)

QKV

https://medium.com/@joaolages/kv-caching-explained-276520203249



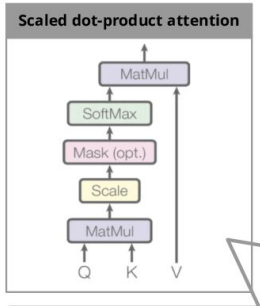
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



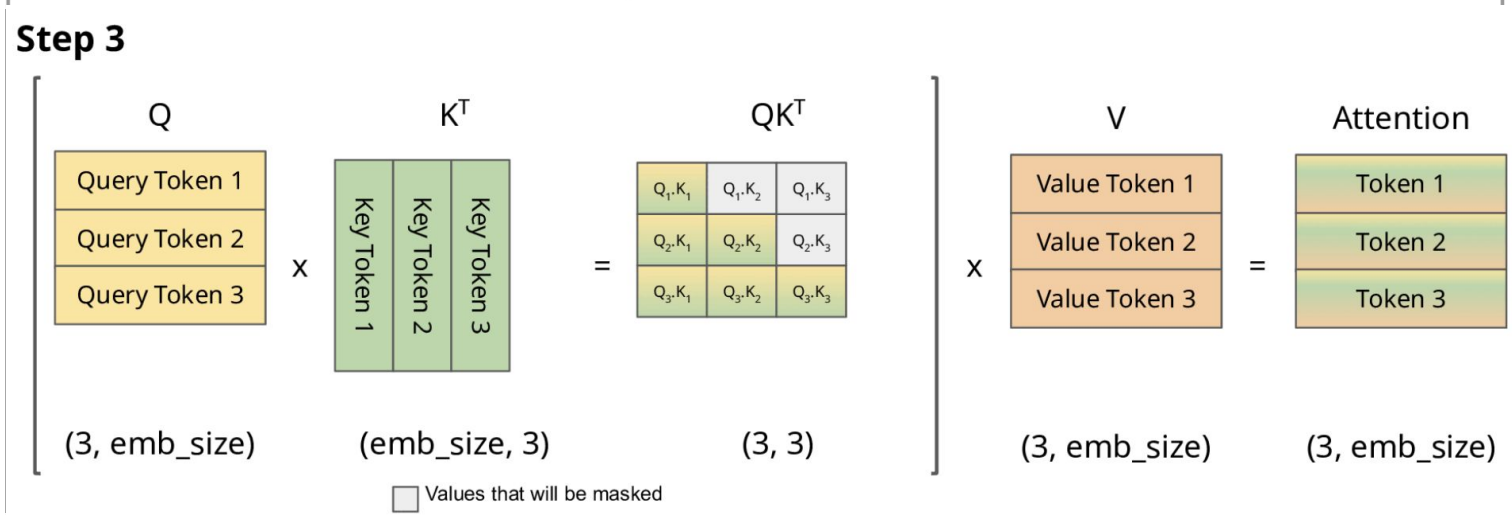
Zoom-in! (simplified without Scale and Softmax)

QKV

https://medium.com/@joaolages/kv-caching-explained-276520203249



$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Zoom-in! (simplified without Scale and Softmax)

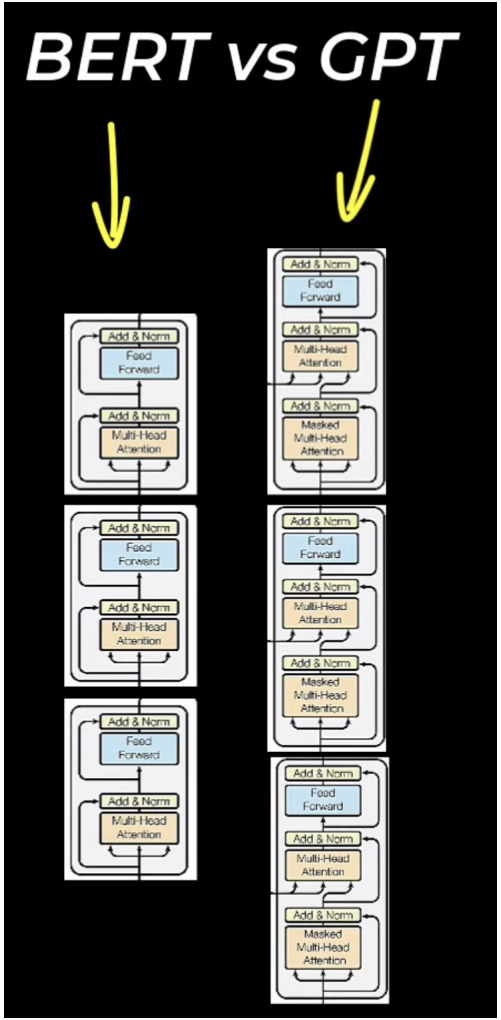
Encoder-Only vs Decoder-Only

Encoder-Only Models (e.g., BERT, RoBERTa)

- Bidirectional Context Understanding
- Heavy Computational Load for Long Sequences

Decoder-Only Models (e.g., GPT, GPT-2, GPT-3)

- Decoders are far far easier to train. You can just take any text and use it predict the next token
- Unidirectional and Efficient for Generative Tasks
- Limited Context Understanding



Encoder-Only vs Decoder-Only

Aspect	Encoder-Only (e.g., BERT)	Decoder-Only (e.g., GPT)
Primary Use	Understanding tasks (e.g., classification, NER, QA)	Text generation (e.g., dialogue, text completion, translation)
Context Handling	Bidirectional (sees both past and future context)	Unidirectional (only sees past context)
Generation Capabilities	Not suitable for generation	Designed for generation tasks
Best Use Cases	Sentiment analysis, text classification, NER, QA	Text generation, dialogue systems, storytelling
Training Objective	Masked Language Modeling (MLM)	Causal Language Modeling (CLM)
Performance on NLU Tasks	High performance on understanding tasks	Weaker performance on understanding tasks
Efficiency for Long Sequences	Can be computationally expensive for very long sequences	Efficient for sequential tasks like generation
Use in Generation Tasks	Ineffective for text generation	Strong in text generation and sequential tasks

https://github.com/embeddings-benchmark/mteb

Massive Text Embedding Benchmark (MTEB) https://arxiv.org/abs/2210.07316

Leaderboard (Sept 2024) https://huggingface.co/spaces/mteb/leaderboard

Rank	Model	Model Size (Million Parameters)	Memory Usage (GB, fp32)	Embedding Dimensions	Max Tokens	Average (56 datasets)	Classification Average (12 datasets)	Clustering Average (11 datasets)
1	NV-Embed-v2	7851	29.25	4096	32768	72.31	90.37	58.46
2	bge-en-icl	7111	26.49	4096	32768	71.67	88.95	57.89
3	stella_en_1.5B_v5	1543	5.75	8192	131072	71.19	87.63	57.69
4	SFR-Embedding-2_R	7111	26.49	4096	32768	70.31	89.05	56.17
5	gte-Qwen2-7B-instruct-Q8_0-GG					70.24	86.58	56.92

https://platform.openai.com/docs/guides/embeddings/embedding-models

OpenAI Embedding Models

Embedding models

OpenAI offers two powerful third-generation embedding model (denoted by `-3` in the model ID). You can read the embedding v3 announcement blog post for more details.

Usage is priced per input token, below is an example of pricing pages of text per US dollar (assuming ~800 tokens per page):

MODEL	~ PAGES PER DOLLAR	PERFORMANCE ON MTEB EVAL	MAX INPUT
text-embedding-3-small	62,500	62.3%	8191
text-embedding-3-large	9,615	64.6%	8191
text-embedding-ada-002	12,500	61.0%	8191

https://cookbook.chromadb.dev/integrations/ollama/embeddings/#ollama-embedding-models

Ollama Embedding Models

Models:

Model	Pull	Ollama Registry Link
`nomic-embed-text`	`ollama pull nomic-embed-text`	nomic-embed-text
`mxbai-embed-large`	`ollama pull mxbai-embed-large`	mxbai-embed-large
`snowflake-arctic-embed`	`ollama pull snowflake-arctic-embed`	snowflake-arctic-embed
`all-minilm-l6-v2`	`ollama pull chroma/all-minilm-l6-v2-f32`	all-minilm-l6-v2-f32

./llama-embedding -m ./path/to/model --pooling mean --log-disable -p "Hello World!" 2>/dev/null

Sentence Embedding

But how exactly are these embeddings computed? Are these the last layer hidden states?

- The embedding outputs would be the same as **the output of 'norm'** in this case.

```
vector = model.embed(text)
eos_vector = vector[-1]
```

```
    // final rmsnorm
    rmsnorm(x, x, w->rms_final_weight, dim);

    // classifier into logits
    matmul(s->logits, x, w->wcls, p->dim, p->vocab_size);
    // printf("number_of_matmuls: %d\n", number_of_matmuls);

    return s->logits;
```

```
enum llama_pooling_type {
    LLAMA_POOLING_TYPE_UNSPECIFIED = -1,
    LLAMA_POOLING_TYPE_NONE = 0,
    LLAMA_POOLING_TYPE_MEAN = 1,
    LLAMA_POOLING_TYPE_CLS  = 2,
};
```

https://github.com/ggerganov/llama.cpp/tree/master/examples/embedding

RAG Demo

```
import chromadb
from chromadb.utils.embedding_functions import OllamaEmbeddingFunction

client = chromadb.PersistentClient(path="ollama")

# create EF with custom endpoint
ef = OllamaEmbeddingFunction(
    model_name="nomic-embed-text",
    url="http://localhost:11434/api/embeddings",
)

print(ef(["Here is an article about llamas..."]))
```

RAG Limitations and Enhancements

Key Issues

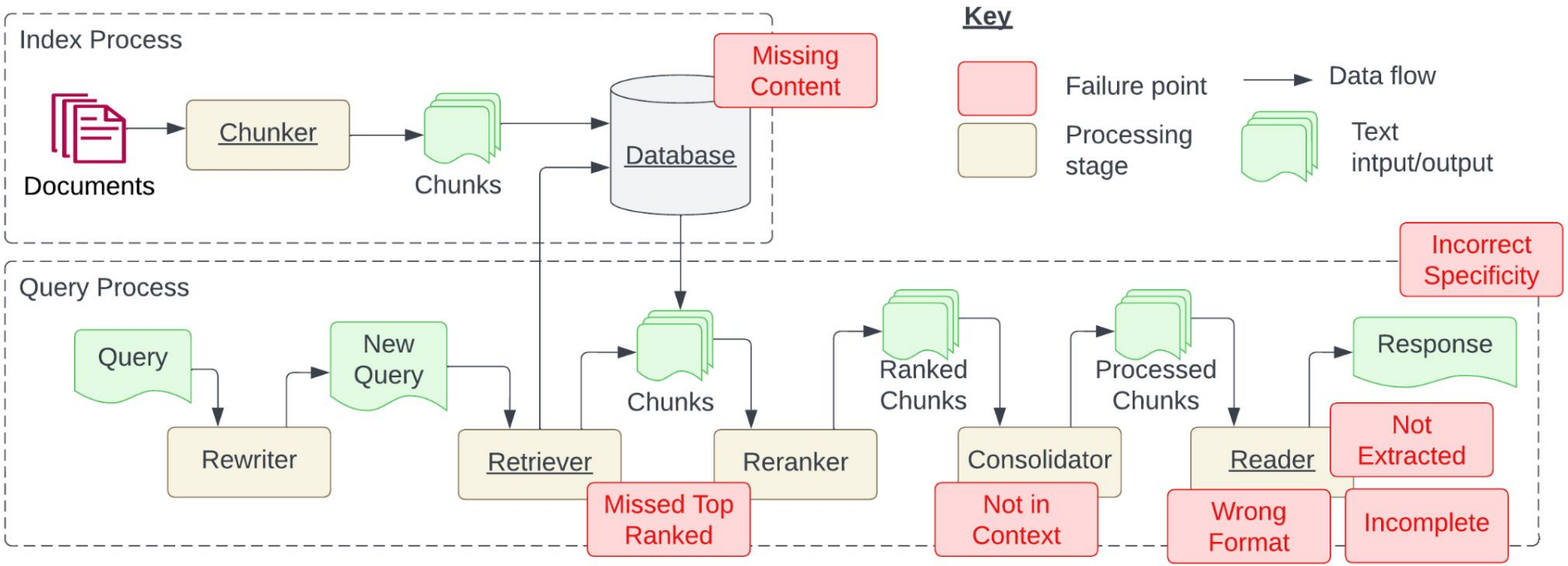
- Quality and relevance of retrieved data
  - individual chunks lack sufficient context
- Mismatch between retrieval and generation
- Dependency on retrieval database
- Increased computational complexity
- Model hallucination
- …

Advancements:

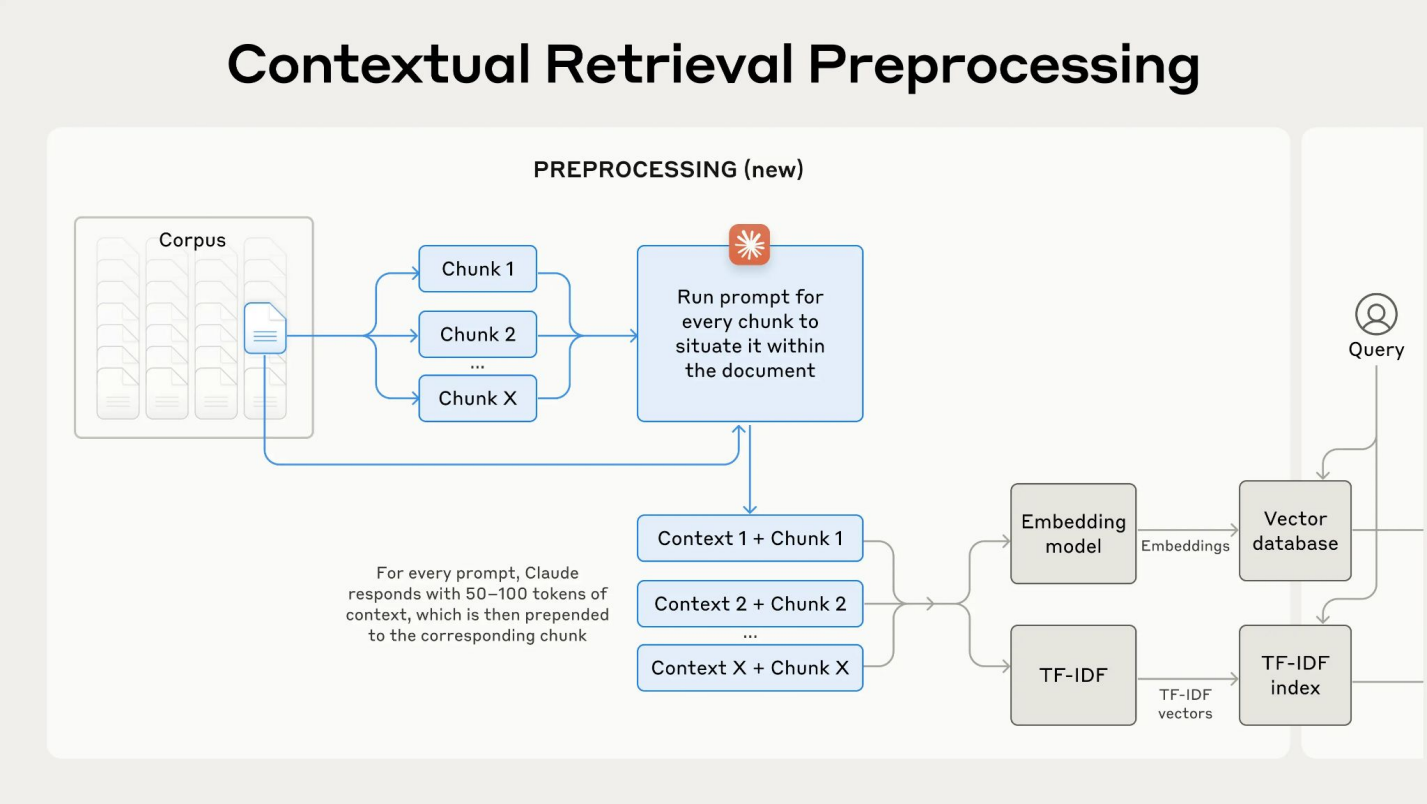
- Contextual Retrieval (by Anthropic): https://www.anthropic.com/news/contextual-retrieval
- GraphRAG: https://arxiv.org/pdf/2404.16130

RAG Limitations and Enhancements

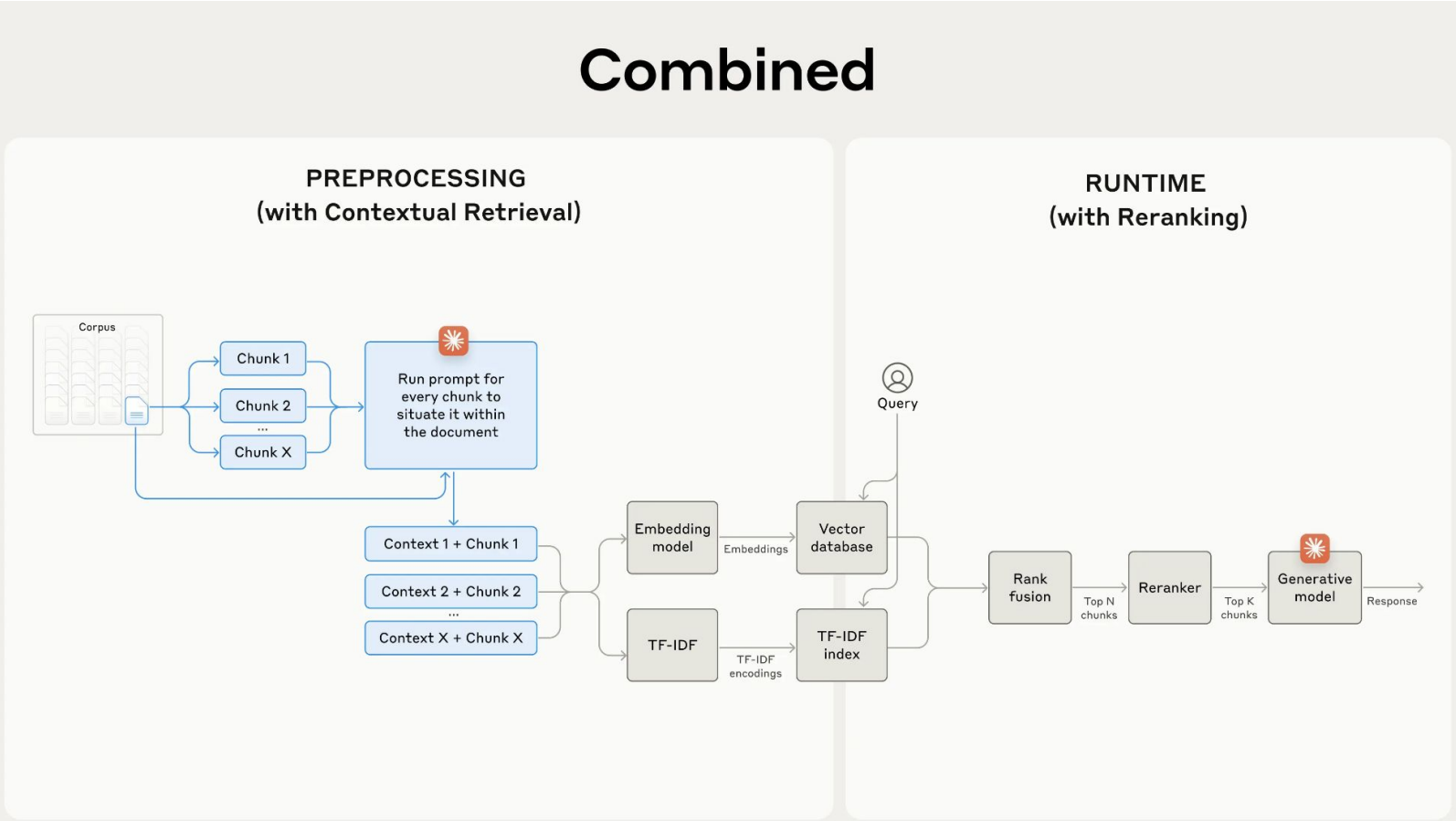
- Seven Failure Points When Engineering a Retrieval Augmented Generation System
  https://arxiv.org/abs/2401.05856

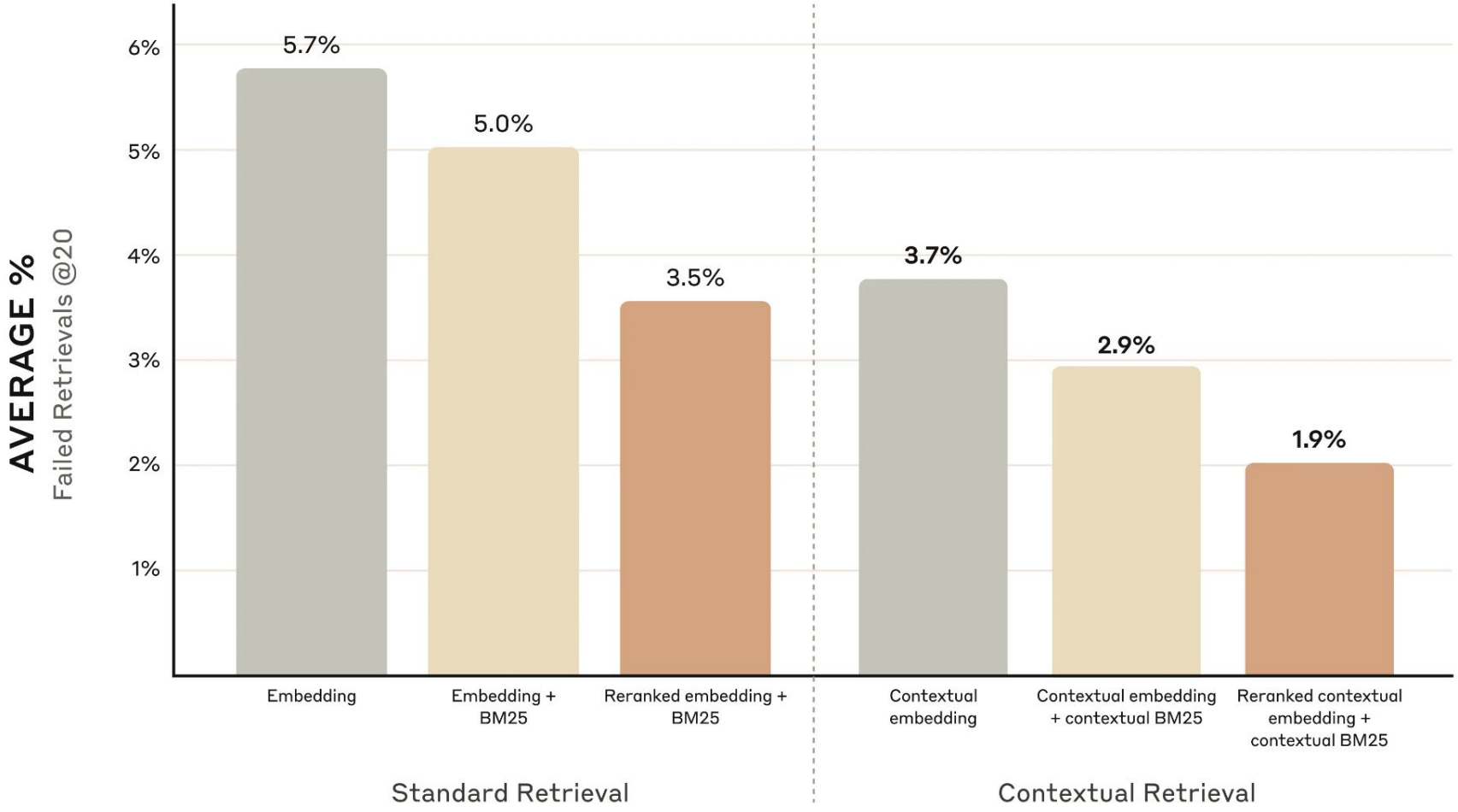


Contextual Retrieval https://www.anthropic.com/news/contextual-retrieval



Contextual Retrieval




AVERAGE %
Failed Retrievals @20
6%
5%
4%
3%
2%
1%
5.7%
5.0%
3.5%
3.7%
2.9%
1.9%
Embedding
Embedding + BM25
Reranked embedding + BM25
Contextual embedding
Contextual embedding + contextual BM25
Reranked contextual embedding + contextual BM25
Standard Retrieval
Contextual Retrieval

GraphRAG https://arxiv.org/pdf/2404.16130

https://www.microsoft.com/en-us/research/blog/graphrag-unlocking-llm-discovery-on-narrative-private-data/

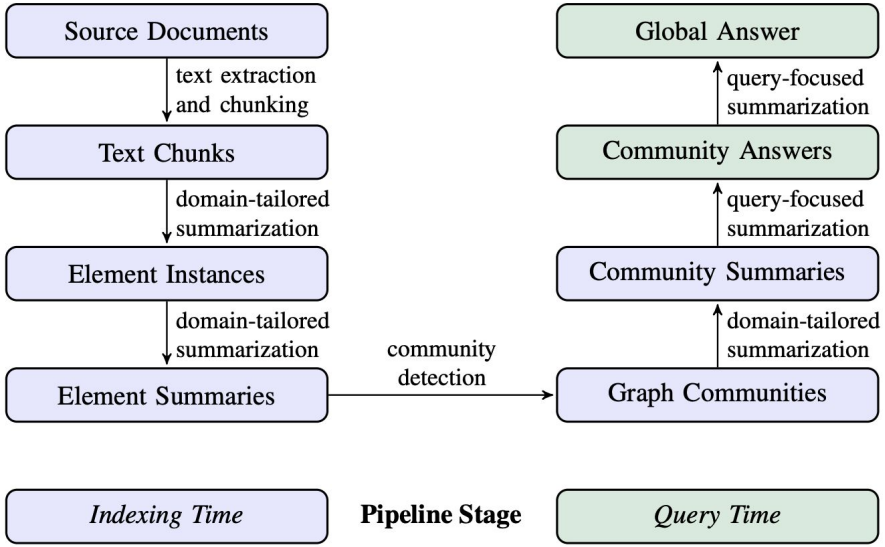
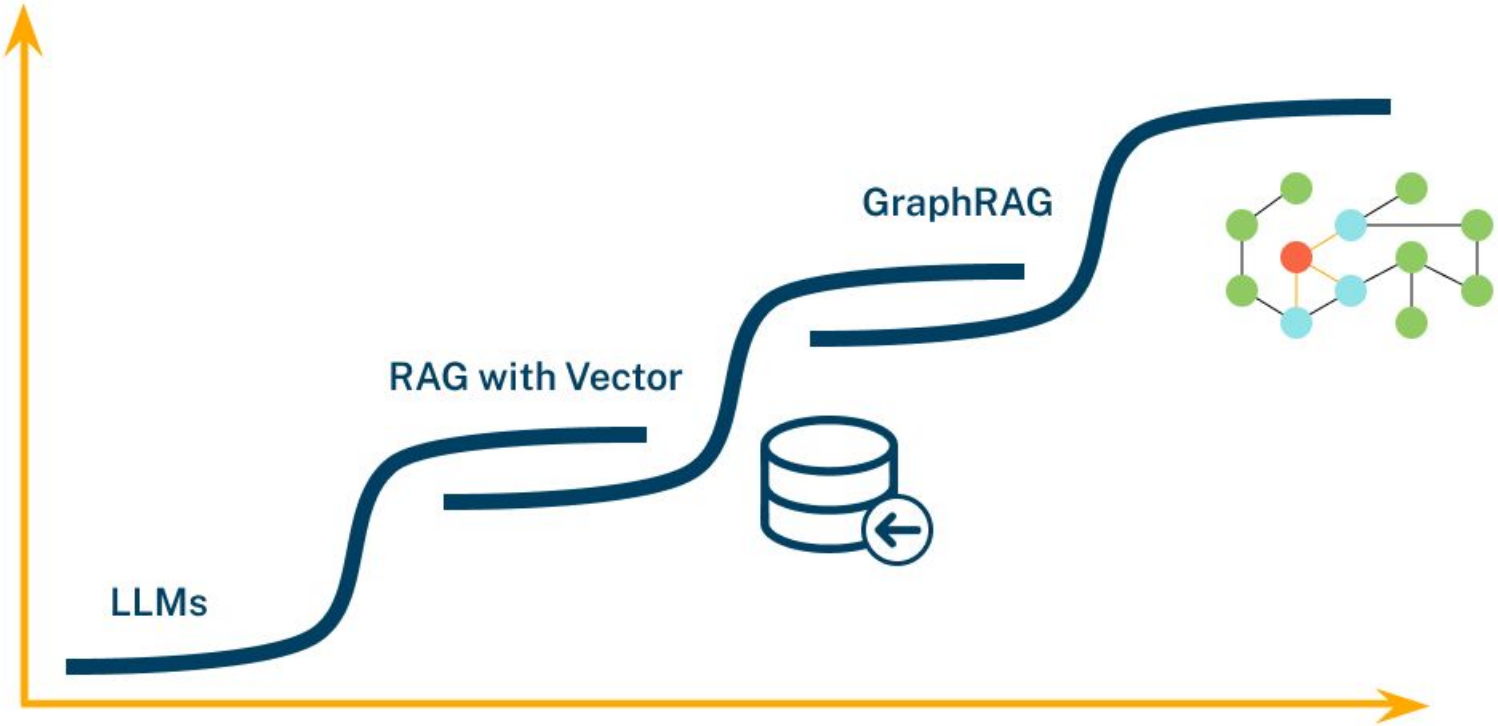


Figure 1: Graph RAG pipeline using an LLM-derived graph index of source document text. This index spans nodes (e.g., entities), edges (e.g., relationships), and covariates (e.g., claims) that have been detected, extracted, and summarized by LLM prompts tailored to the domain of the dataset. Community detection (e.g., Leiden, Traag et al., 2019) is used to partition the graph index into groups of elements (nodes, edges, covariates) that the LLM can summarize in parallel at both indexing time and query time. The "global answer" to a given query is produced using a final round of query-focused summarization over all community summaries reporting relevance to that query.


GraphRAG
RAG with Vector
LLMs

Knowledge Graph and Vector Database Integration

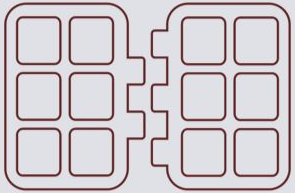
Input Prompt

Graph Querying

Information Retrieval

Vector DB

Prompt Enrichment



The retrieved information from the knowledge graph and vector database is combined with the original user prompt to create an enriched prompt. This can be done in via concatenation: The retrieved information is appended to the original user prompt as additional text context.

LLM-Based Answer Generation

Output

RAG vs Fine-Tuning

Feature Comparison	RAG	Fine-Tuning
Knowledge Updates	Directly updating the retrieval knowledge base ensures that the information remains current without the need for frequent retraining, making it well-suited for dynamic data environments.	Stores static data, requiring retraining for knowledge and data updates.
External Knowledge	Proficient in leveraging external resources, particularly suitable for accessing documents or other structured/unstructured databases.	Can be utilized to align the externally acquired knowledge from pretraining with large language models, but may be less practical for frequently changing data sources.
Data Processing	Involves minimal data processing and handling.	Depends on the creation of high-quality datasets, and limited datasets may not result in significant performance improvements.
Model Customization	Focuses on information retrieval and integrating external knowledge but may not fully customize model behavior or writing style.	Allows adjustments of LLM behavior, writing style, or specific domain knowledge based on specific tones or terms.
Interpretability	Responses can be traced back to specific data sources, providing higher interpretability and traceability.	Similar to a black box, it is not always clear why the model reacts a certain way, resulting in relatively lower interpretability.
Computational Resources	Depends on computational resources to support retrieval strategies and technologies related to databases. Additionally, it requires the maintenance of external data source integration and updates.	The preparation and curation of high-quality training datasets, defining fine-tuning objectives, and providing corresponding computational resources are necessary.
Latency Requirements	Involves data retrieval, which may lead to higher latency.	LLM after fine-tuning can respond without retrieval, resulting in lower latency.
Reducing Hallucinations	Inherently less prone to hallucinations as each answer is grounded in retrieved evidence.	Can help reduce hallucinations by training the model based on specific domain data but may still exhibit hallucinations when faced with unfamiliar input.
Ethical and Privacy Issues	Ethical and privacy concerns arise from the storage and retrieval of text from external databases.	Ethical and privacy concerns may arise due to sensitive content in the training data.

Figure Source

RAG Best Practices

https://behitek.com/blog/2024/07/18/rag-in-production/

- Try the simple method first. eg. BM25
- Maintain the hierarchical structure of your documents
- Take a look at the chunking strategy
- Consider using text summarization for long documents
- Take a look at the format of (search_query, document) pairs
- Moving tasks from RAG-frontend to RAG-backend