Vector Databases for Retrieval Augmented Generation (RAG)

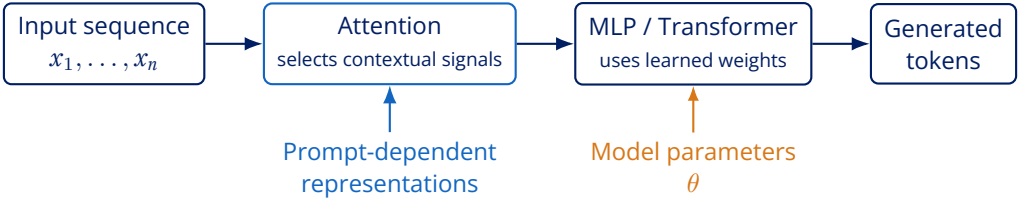
CS 594: Vector Databases

A. Asudeh, asudeh@uic.edu

Fall 2026



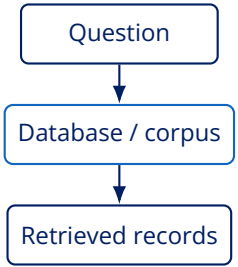
LLMs at a high level: context versus learned parameters



- The model learns statistical associations during training and uses them to predict tokens.
- Factual information is not stored as a directly queryable database of facts.

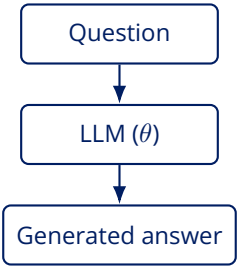
Retrieval-only versus parametric generation

Retrieval-only system



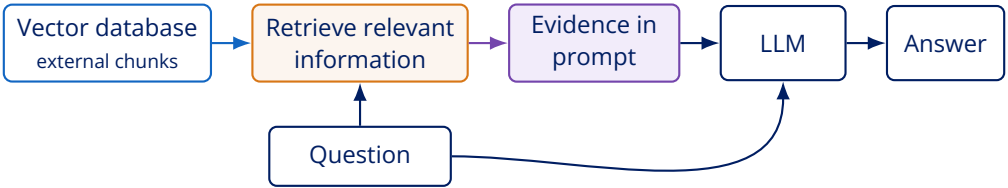
+ Can expose the stored evidence.
– Limited synthesis of information.

LLM-only system



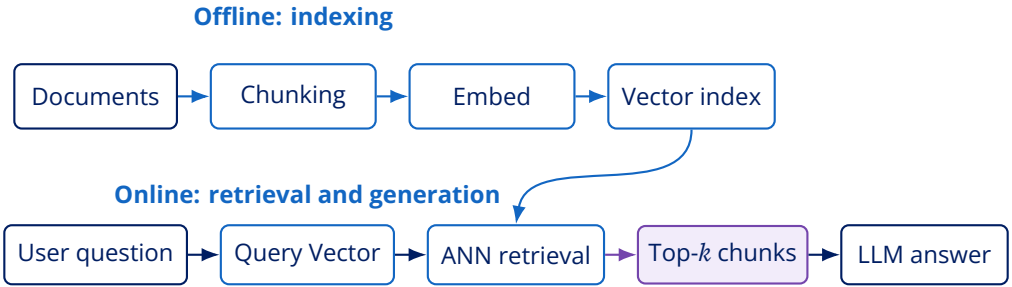
+ Can reason and integrate/explain information.
– May hallucinate: generate unsupported claims.

RAG combines LLMs with vector databases



- External passages may contain facts missing from the model's learned knowledge.
- The LLM still uses its parameters to interpret the question and generate text.

Indexing, retrieval, generation



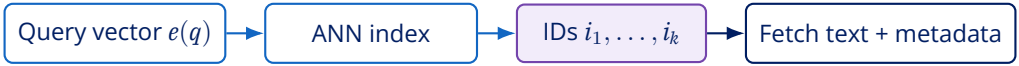
The vector database implements the search stage. The LLM receives the selected chunks, not the vector index itself.

Gao et al., arXiv:2312.10997, Fig. 2 (redrawn as a simplified pipeline).

What does the vector database do?

- Each document chunk d_i is encoded into $e(d_i) \in \mathbb{R}^d$ and stored with a chunk ID and metadata.
- A question q is encoded into $e(q)$; the system retrieves chunks with high similarity.

$$D_k(q) = \mathrm{TopK}_{d_i \in D} \ \mathrm{sim}\big(e(q), e(d_i)\big)$$



Three challenges

1. **Retrieval quality:** Do we return the right information?
   High precision and high recall are both desirable.

2. **Integrating retrieved documents into the prompt:**
   Noise, context-window limits, and loss of information in the middle.

3. **Reliance on learned parameters:**
   The LLM may prefer its parametric knowledge over the supplied evidence.

Challenge 1: Retrieval quality

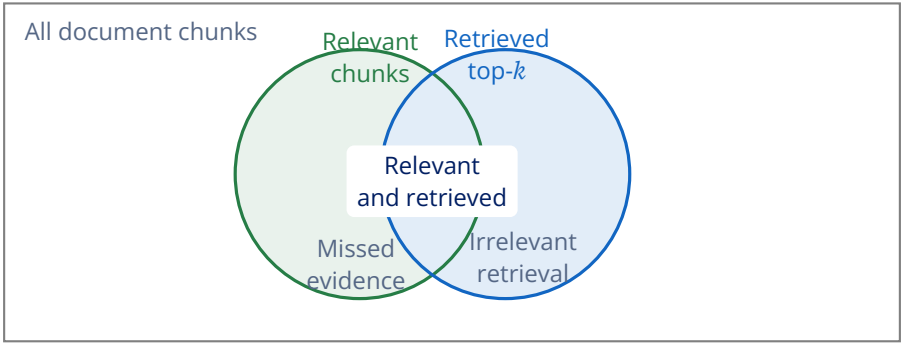
High precision

- Most returned chunks are relevant to the query.
- Avoid injecting irrelevant evidence into the prompt.

High recall

- The retriever does not miss relevant chunks.
- The necessary evidence is in the candidate set.

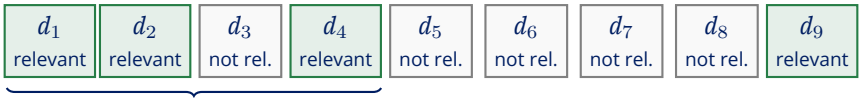
What precision and recall measure



- Precision concerns what *we return*; recall concerns what *we miss*.

The top-k tradeoff: a small example

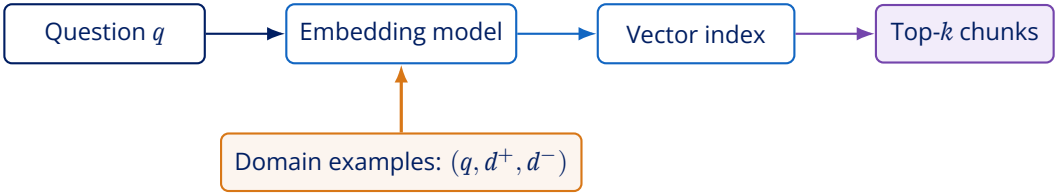
Suppose there are **four** relevant chunks in the collection. The ranked list begins:



k	2	4	6
Precision@k	1	$3/4$	$1/2$
Recall@k	$1/2$	$3/4$	$3/4$

Increasing k cannot decrease recall@k, but precision@k need not be monotone.

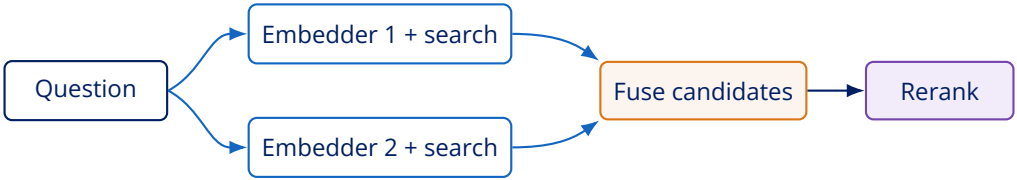
High precision: Improve the embedding model



- Fine-tune an embedder so relevant chunks rank above confusing negatives.

High precision: Multiple retrieval models

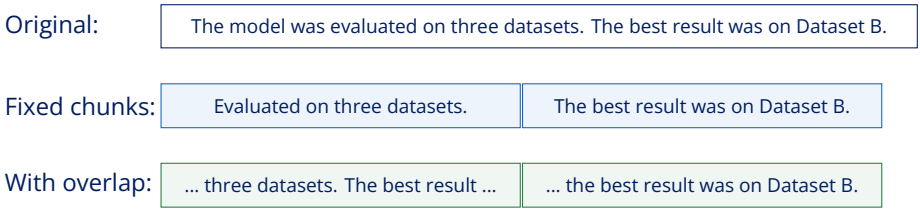
Different embedders can capture different notions of relevance.



Candidate fusion may improve coverage, but it adds computation cost and can bring in more irrelevant chunks. Scores from distinct models are not automatically comparable.

High recall: Chunking affects what can be found

Evidence can cross a chunk boundary.



- Smaller chunks reduce noise but may split facts across boundaries.
- Larger or overlapping chunks preserve context but require more tokens.

Gao et al., Section III-B.

High recall: Increase the number of candidates

Larger k retrieves more candidates before generation.



- **Benefit:** a larger candidate set can include previously missed evidence.
- **Cost:** more irrelevant chunks, reranking work, and prompt tokens.

We can retrieve many candidates, then select a much smaller context for the LLM.

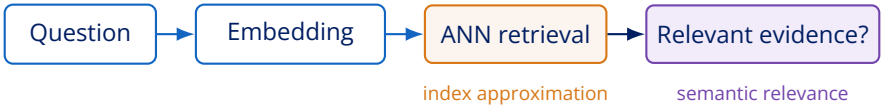
Two different sources of retrieval error

Vector index error: ANN misses exact nearest neighbors under the selected similarity function.

$$\text{ANNRecall@}k = \frac{|A_k(q) \cap E_k(q)|}{k}.$$

A_k: approximate top-k; E_k: exact top-k in embedding space.

Semantic error: Exact nearest neighbors may still lack the evidence needed to answer the question.



Challenge 2: Integrating retrieved documents into the prompt

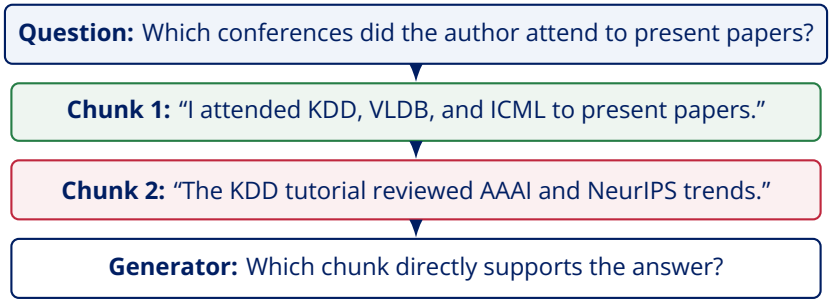
Even if the correct passage is retrieved, the final answer can fail.

1. **Noise:** Irrelevant or conflicting content can mislead the LLM.
2. **Limited context window:** Only some retrieved text fits in the prompt.
3. **Lost in the middle:** The model may use evidence differently depending on its location.

Asudeh, RAG notes, p. 2; Gao et al., Sections II and IV.

Noise can mislead the generator

Illustration: the query asks for a conference, but the retrieved context mixes evidence and noise.



A chunk may share query terms without supporting the requested claim.

The context window is a resource constraint

The prompt budget must accommodate instructions, the question, retrieved evidence, and generated tokens.

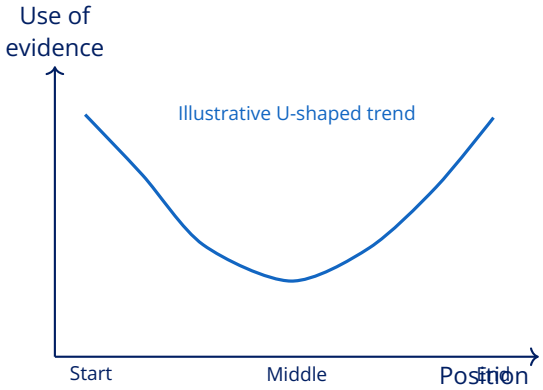
$$T_{\text{instructions}} + T_q + \sum_{i \in S} T_{d_i} + T_{\text{answer}} \leq B$$



Example budget allocation (not to scale)

A larger context window does not eliminate retrieval costs, distraction, or evidence-selection decisions.

Lost in the middle



A model can perform worse when the necessary passage appears in the middle of a long prompt.

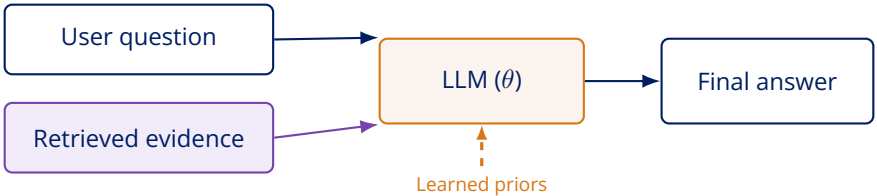
Implication: Context selection and ordering can change answer quality.

The shape depends on model, task, and prompt. The curve is schematic, not experimental data.

Liu et al., “Lost in the Middle: How Language Models Use Long Contexts,” TACL 12 (2024).

Challenge 3: The LLM may rely on its parameters

Even with retrieved evidence in its prompt, the generator need not follow that evidence.



The model may ignore, misinterpret, or contradict relevant passages. A correct retrieval is not sufficient for a correct generated answer.

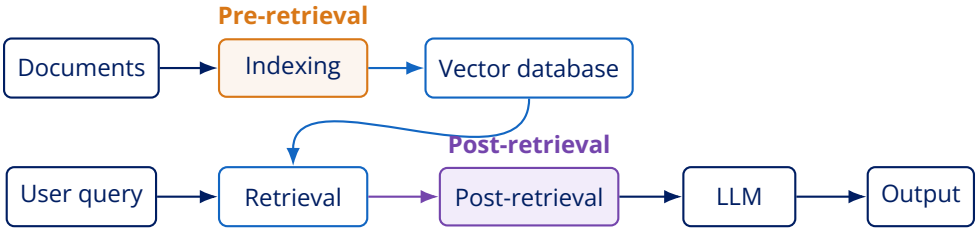
Reduce reliance on unsupported knowledge

- **Grounding and abstention:** Instruct the model to cite supporting passages and to say when evidence is missing.
- **Fine-tune the LLM for RAG:** Train it to answer from the supplied context, including examples with conflicting priors.
- **Verification:** Check whether each generated factual statement is supported by its cited source.

Prompting or fine-tuning can improve evidence use, but neither guarantees faithfulness.

Gao et al., Sections IV–V and VII.

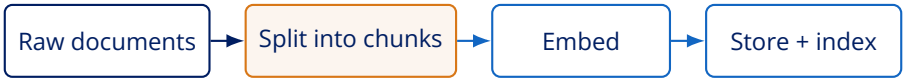
Where can we improve the RAG pipeline?



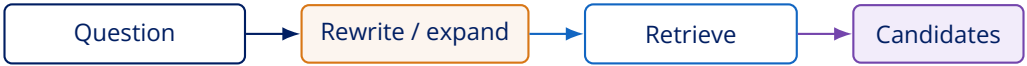
Before retrieval:
Chunking, embedding, metadata, query rewriting.

After retrieval:
Reranking, filtering, context compression.

Pre-retrieval: Build a searchable collection



Pre-retrieval: Improve the question before search



- Rewrite a vague query to make entities, constraints, or terminology explicit.
- Use **reasoning** and multiple reformulations when no single query covers the information need.

Query rewriting introduces its own failure mode: it may change the user's intended question.

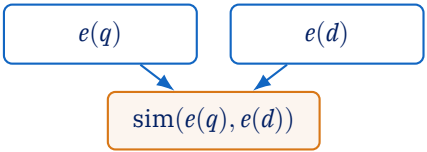
Post-retrieval: Improve the selected context



- **Reranking:** order the candidates using a more precise relevance score.
- **Filtering:** discard chunks that do not support the question.
- **Compression:** keep the relevant parts of the remaining chunks.

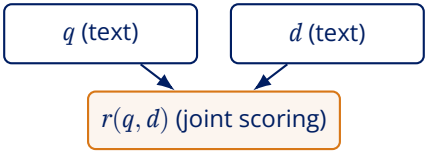
Similarity search versus reranking

Dense retrieval



Embeddings can be precomputed and indexed.
Suitable for large candidate sets.

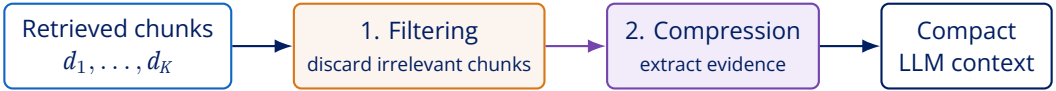
Query–document reranker



Can jointly inspect question and chunk.
Usually more expensive per candidate.

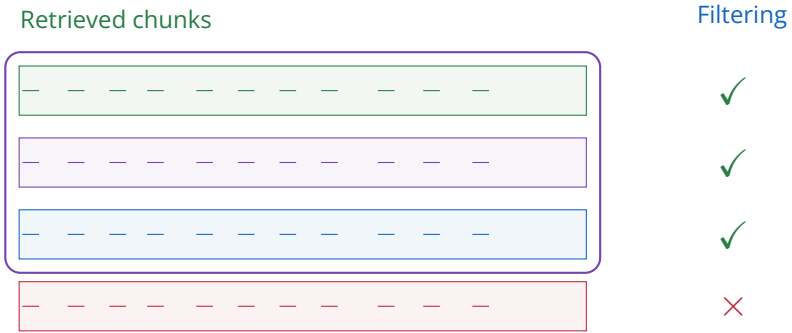
Filtering and compression

The goal is to remove noise from retrieved documents and retain the information needed to answer the question.

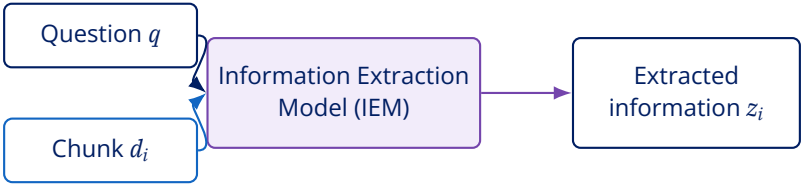


Two different decisions: Which chunks should remain? Which parts of each remaining chunk should be included?

Step 1: Filter irrelevant chunks



Step 2: Extract the relevant parts



- Use the question to identify the evidence-bearing sentences or spans in each chunk.
- Preserve the meaning of the source; extracting only keywords is not necessarily sufficient.

Compression must be query-dependent

Source chunk

The author attended KDD, VLDB, and ICML to present their papers. The trips took place in different months. The author also collaborated on a course project.

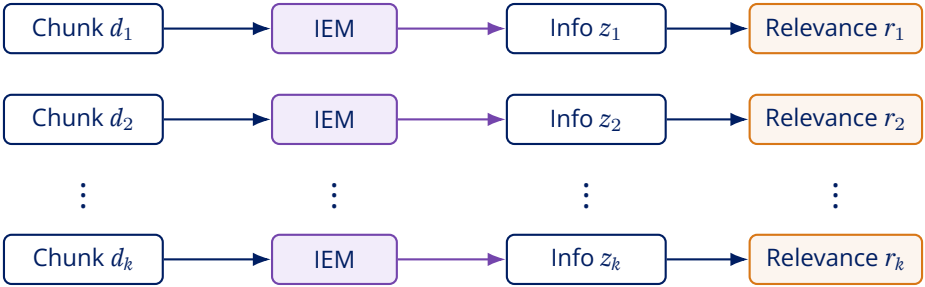
Question

Which conferences did the author attend to present papers?

Query-specific extraction

“Attended KDD, VLDB, and ICML to present papers.”

Apply the IEM to each retrieved chunk



After extraction, rank the retained evidence or estimate how useful each extracted item is for answering q.

Recreated from Asudeh, RAG notes, p. 5.

RAG pipeline: components evaluation

- **Retrieval:** Did we find the required chunks? Measure precision@k, recall@k, and ANN recall.
- **Evidence selection:** Did reranking and extraction keep sufficient, faithful source text?
- **Generation:** Is the answer correct, supported by the selected context, and appropriately abstaining?
- **Efficiency:** Measure search latency, reranking time, prompt tokens, and LLM runtime.



References

1. Y. Gao et al. *Retrieval-Augmented Generation for Large Language Models: A Survey*. arXiv:2312.10997, v5, 2024.
   https://arxiv.org/pdf/2312.10997

2. N. F. Liu et al. *Lost in the Middle: How Language Models Use Long Contexts*. TACL 12:157–173, 2024.
   https://aclanthology.org/2024.tacl-1.9/

Thank you!