

# From Idea to Impact: Building RAG Beyond POC

*Maria Khalusova*  
*Staff Developer Advocate at Unstructured.io*

*Feb 2025*

# LLMs can hallucinate when information is not in their training data

```
1 import ollama
2
3 response = ollama.chat(model='llama3:latest', messages=[
4     {
5         'role': 'user',
6         'content': 'Who is Maria Khalusova?',
7     },
8 ])
9
10 print(response['message']['content'])
```

Executed at 2025.02.05 09:36:34 in 14s 262ms

✓ Maria Khalusova was a Russian aristocrat and socialite who lived in the 18th century. She is often referred to as "the most beautiful woman in Russia" due to her stunning physical appearance and her reputation for being kind, gentle, and generous. :

Khalusova was born in 1746 into a noble family and grew up with all the privileges and luxuries that came with her social status. However, she is said to have been a humble and unassuming person who used her beauty and charm to bring joy and comfort to those around her.

Maria Khalusova's life was marked by tragedy, including the death of her husband at a young age, but , despite these challenges, she continued to live with kindness and compassion. Her reputation for , being one of the most beautiful women in Russia has endured through the centuries, and she remains a , beloved figure in Russian history and culture.

# Modern LLMs hallucinate less. But this answer is not useful

```
1 import ollama
2
3 response = ollama.chat(model='deepseek-r1:8b', messages=[
4     {
5         'role': 'user',
6         'content': 'Who is Maria Khalusova?',
7     },
8 ])
9
10 print(response['message']['content'])
    Executed at 2025.02.05 09:34:31 in 1s 614ms
```

<think>

</think>

As of my last update in July 2024, there is no widely recognized public figure or notable individual named Maria Khalusova. It's possible that she could be a private individual, a lesser-known personality, or someone who has gained prominence after my last update. If you have more context or specific details about her, I might be able to provide more accurate information.

# LLM-based systems over pure LLMs

## Response augmented with web search results:

Who is Maria Khalusova?



Maria Khalusova is a Staff Developer Advocate at Unstructured.io, focusing on preprocessing complex unstructured data for Generative AI applications, such as Retrieval-Augmented Generation (RAG). Prior to this, she contributed to the open-source team at Hugging Face. Maria has co-authored courses on building applications with open-source models and on Audio Transformer models, both available to the public.

CONFOO.CA





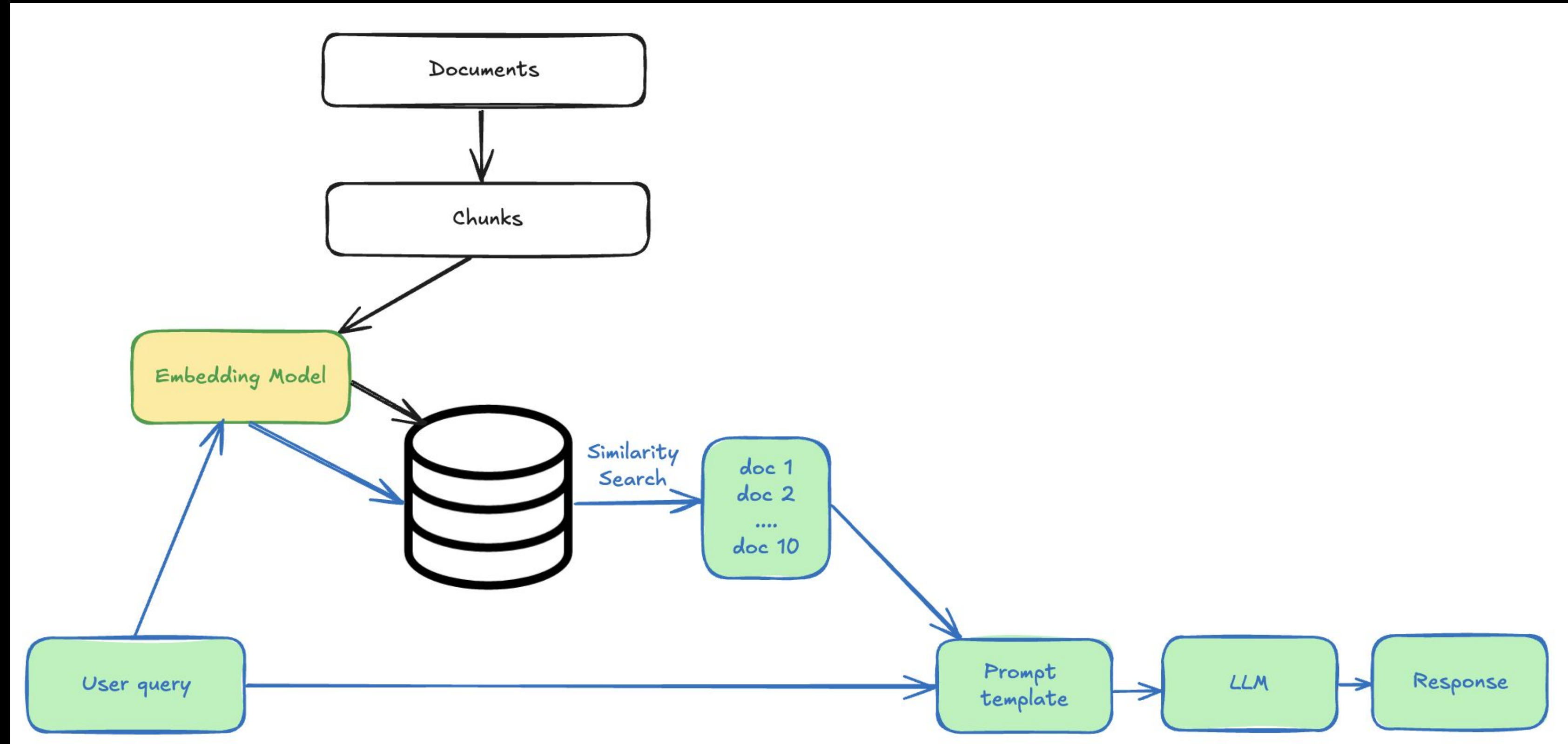
# Retrieval-Augmented Generation (RAG)

```
prompt = "Who is Maria Khalusova?"
```

```
user_query = "Who is Maria Khalusova"

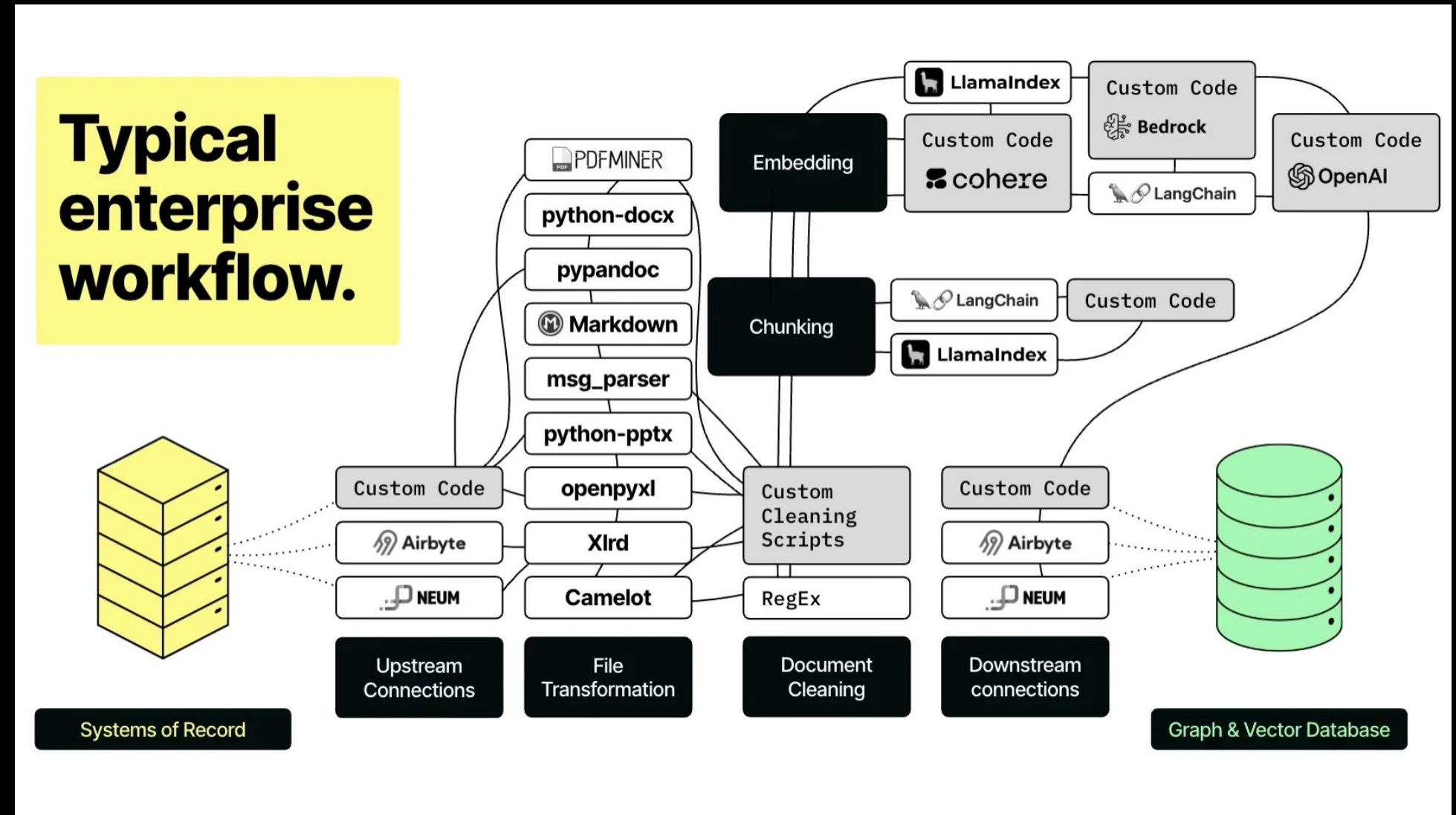
prompt = f"""
User prompt: {user_query}
Relevant context: {retrieved_documents}
Instructions:
* Provide a comprehensive, accurate, and coherent response to the user query,
  using the provided context.
* If the retrieved context is sufficient, deliver precise and relevant
  information.
If the retrieved context is insufficient, acknowledge the gap and suggest
  potential sources or steps for obtaining more information.
* Avoid speculating or generalizing.
"""
```

# Vanilla RAG (aka Naive RAG)



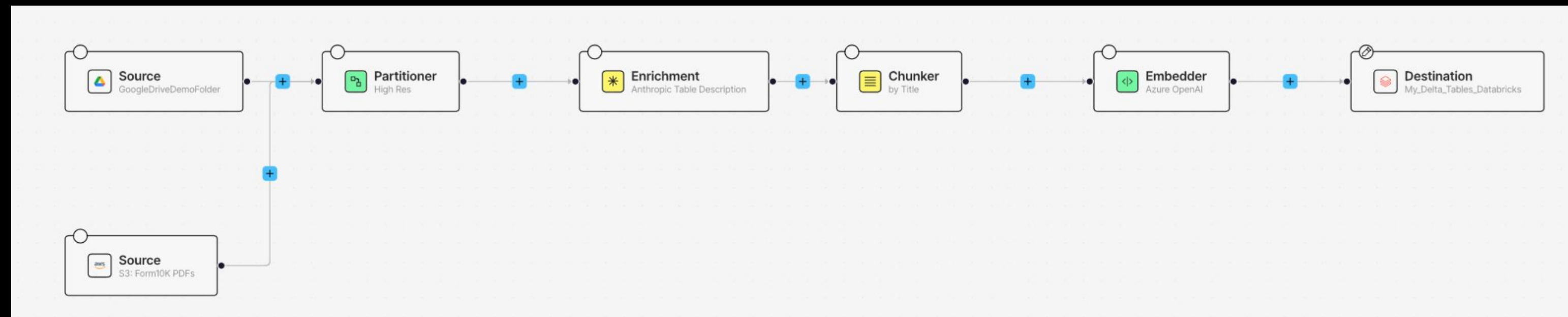
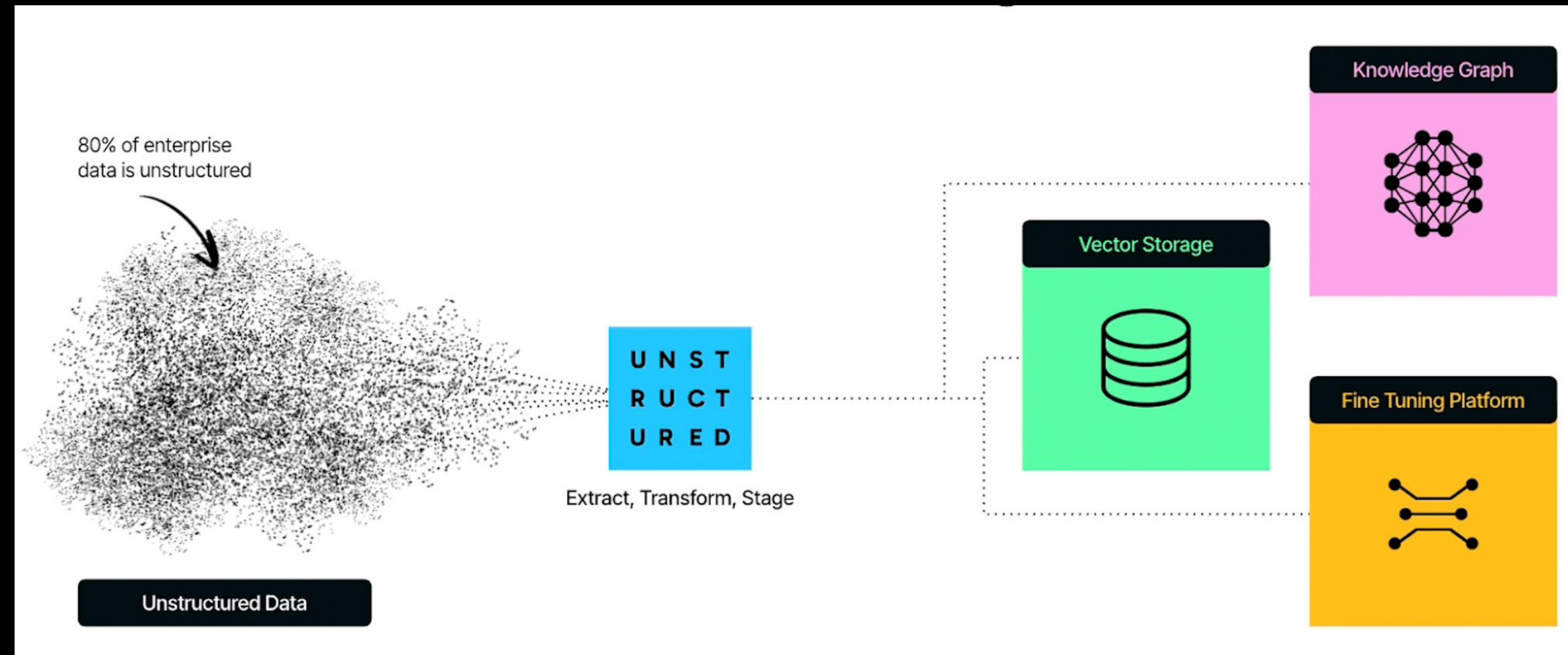
# It starts with the data

- 80% of data is unstructured
- Contained in silos
- Complex to process due to native formats
- May require cleaning (e.g. PII)
- Needs to be chunked
- Needs to be embedded



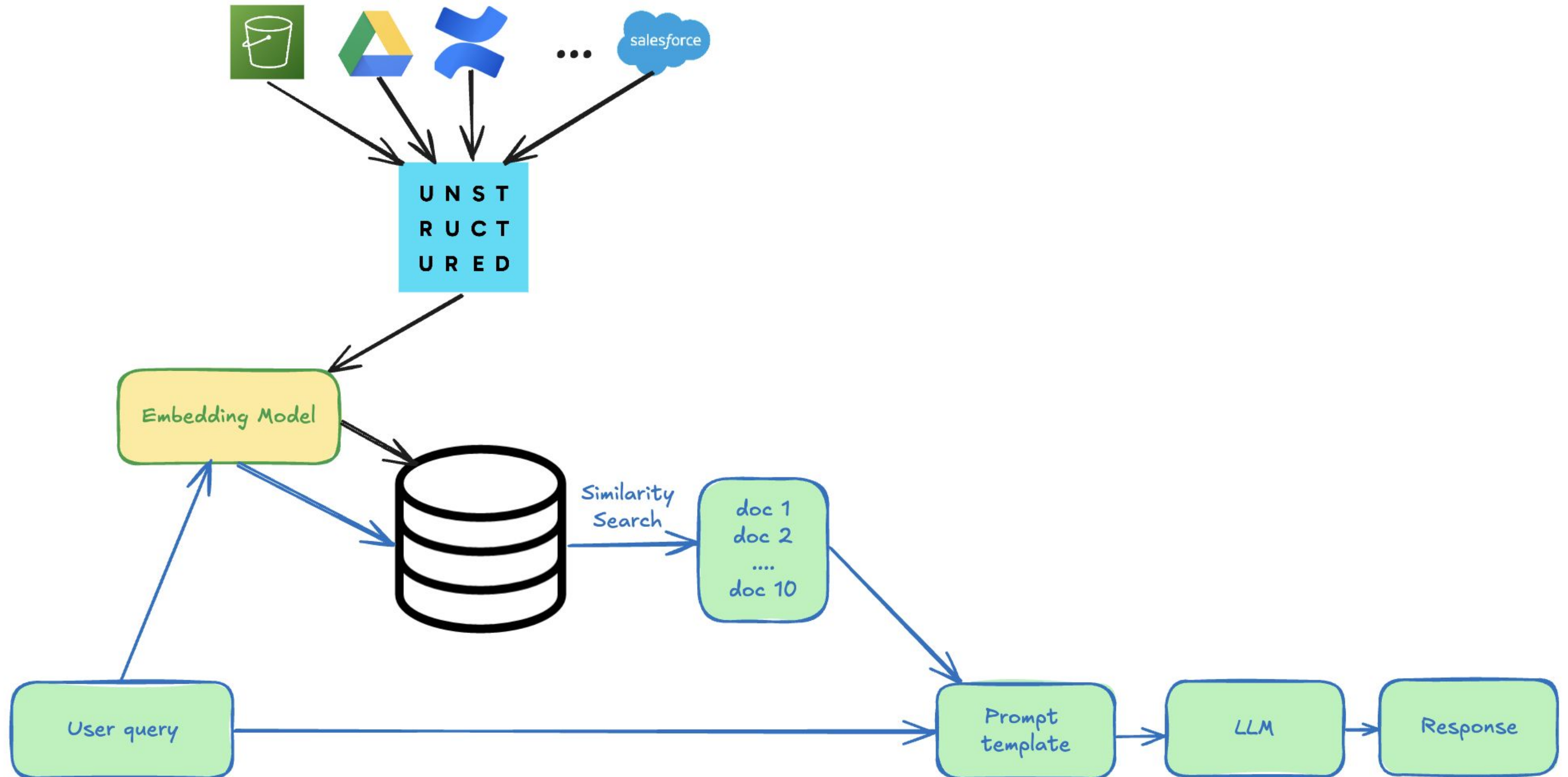


# With Unstructured





# Realistic RAG POC



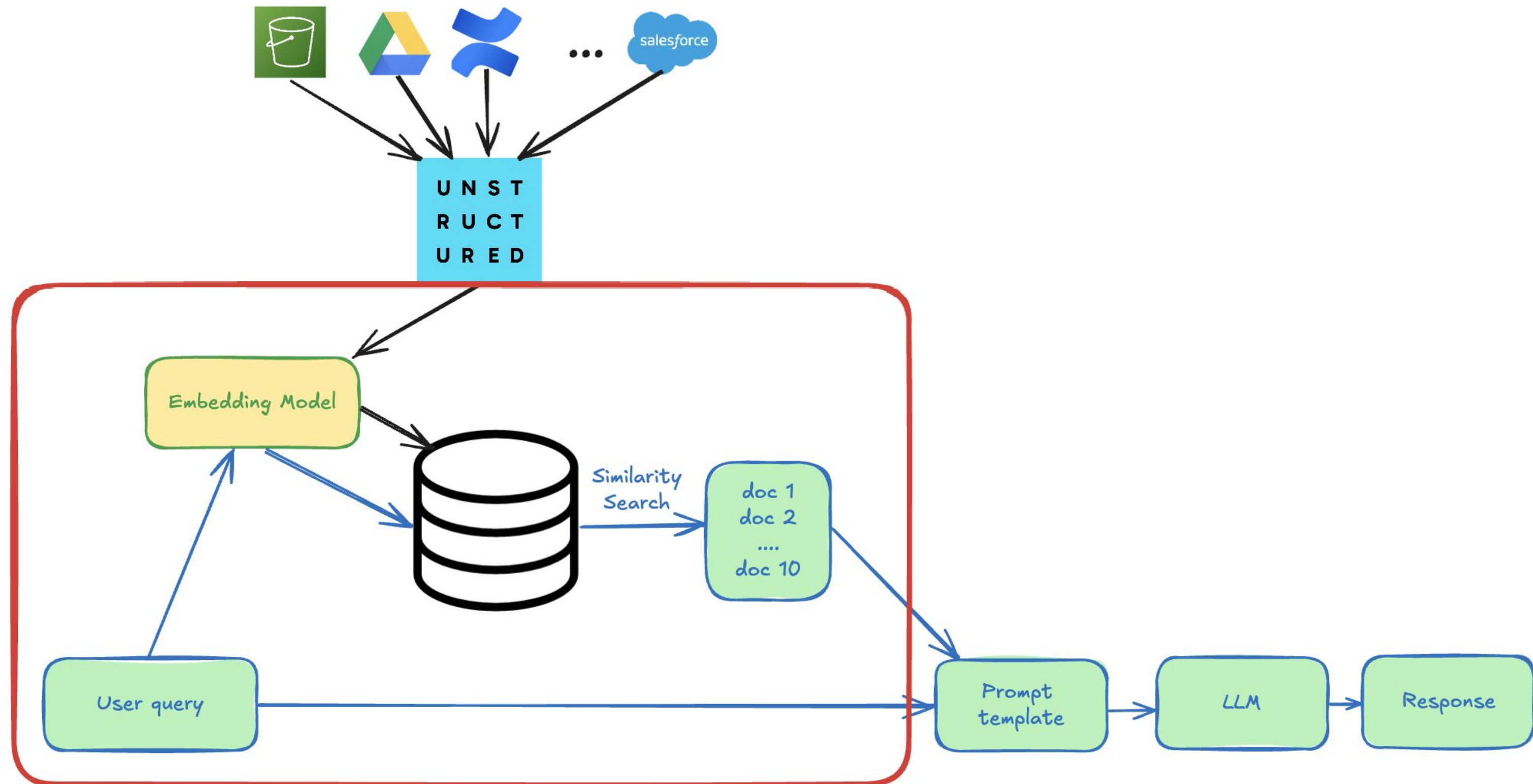
# R in RAG is critical

If you can't retrieve relevant context, no amount of prompt engineering will help.

How to think about retrieval:

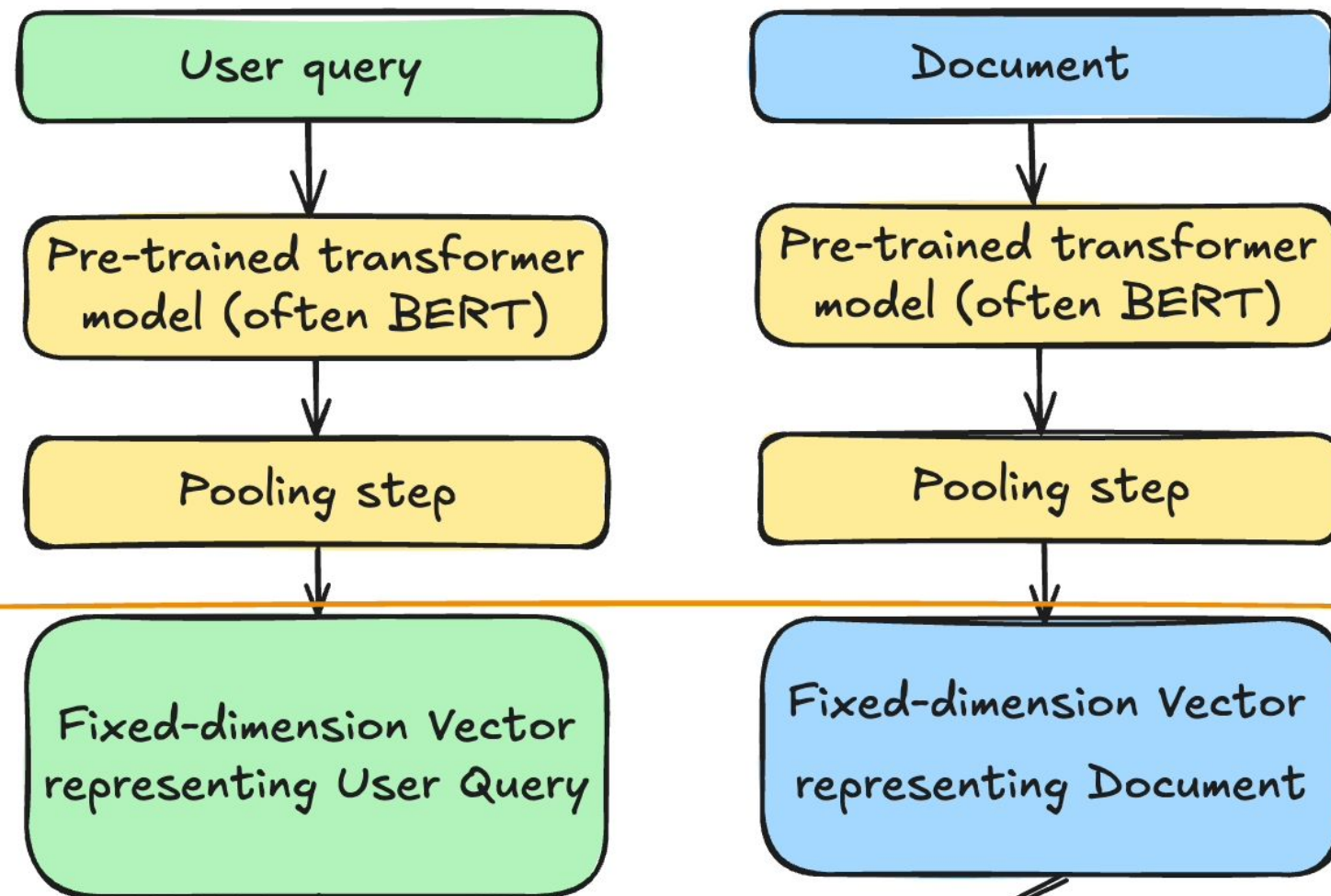
1. Make sure relevant context exists in the retriever.
2. Optimize to retrieve all relevant docs to ensure complete and accurate answers
3. Don't pass to an LLM too many irrelevant docs to avoid needle-in-haystack and unnecessary costs
4. Retrieved docs order doesn't matter.

# Similarity search in RAG



# Bi-Encoder

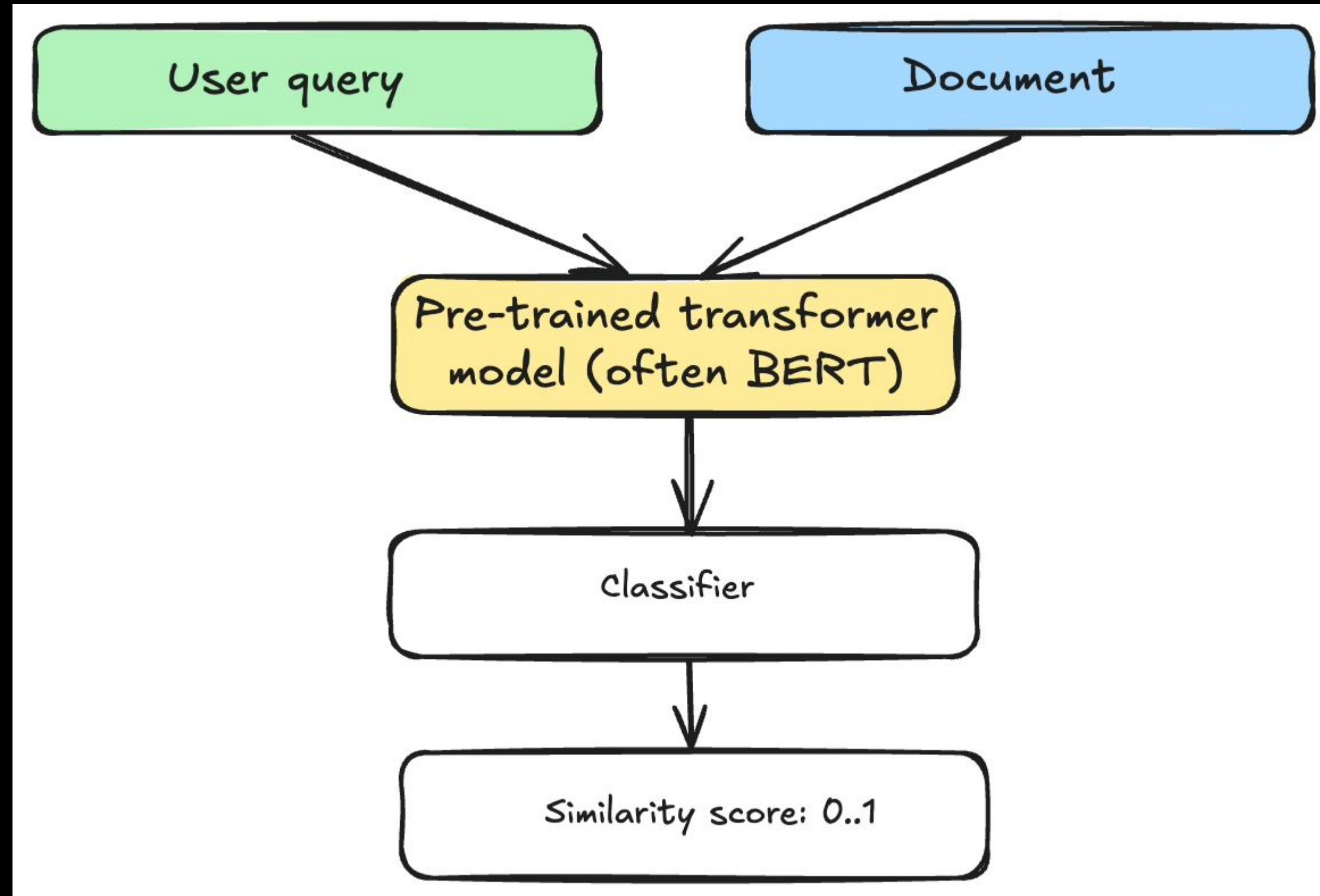
STEP 1: creating embeddings



STEP 2: calculating similarity



# Cross-Encoder



# Cross vs Bi-Encoders

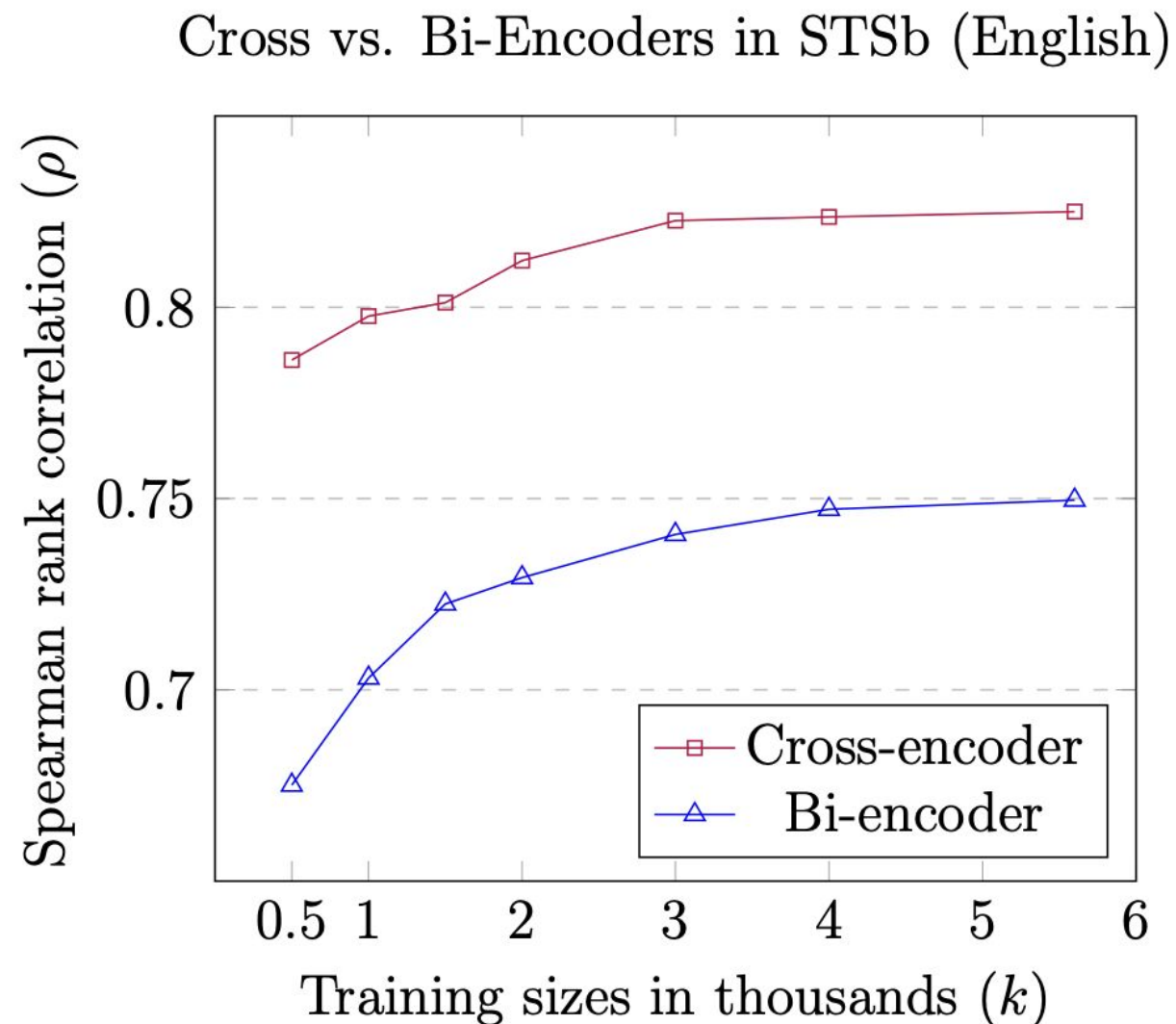


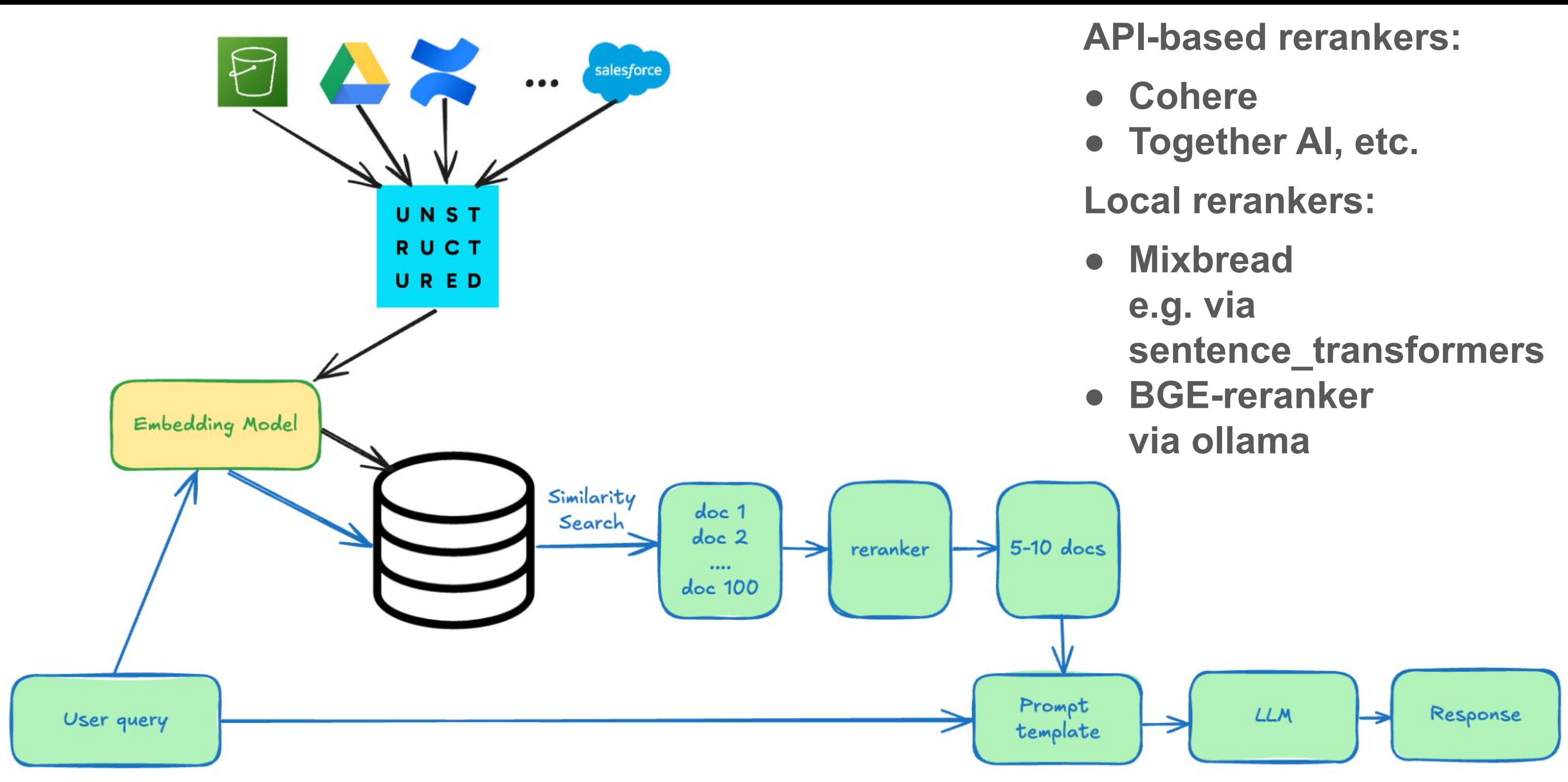
Figure 1: Spearman rank correlation ( $\rho$ ) test scores for different STS Benchmark (English) training sizes.

<https://arxiv.org/pdf/2010.08240>

**Cross-encoders are better at matching similar documents, but are wildly inefficient:**

- **1000x+ slower than bi-encoders**
- **require significantly more computational resources**
- **do not scale**

# Cross-encoder as a reranker



# Similarity search is not a silver bullet

**Embeddings compress information into a dense vector which leads to information loss**

**Embedding model's training data is never 100% representative of your data (custom product names, internal acronyms, etc.)**

**People love using keywords when searching**



# BM25: an oldie but a goodie

**Keyword search (aka full-text search) based on term frequency and inverse document frequency**

**Strong baseline on its own**

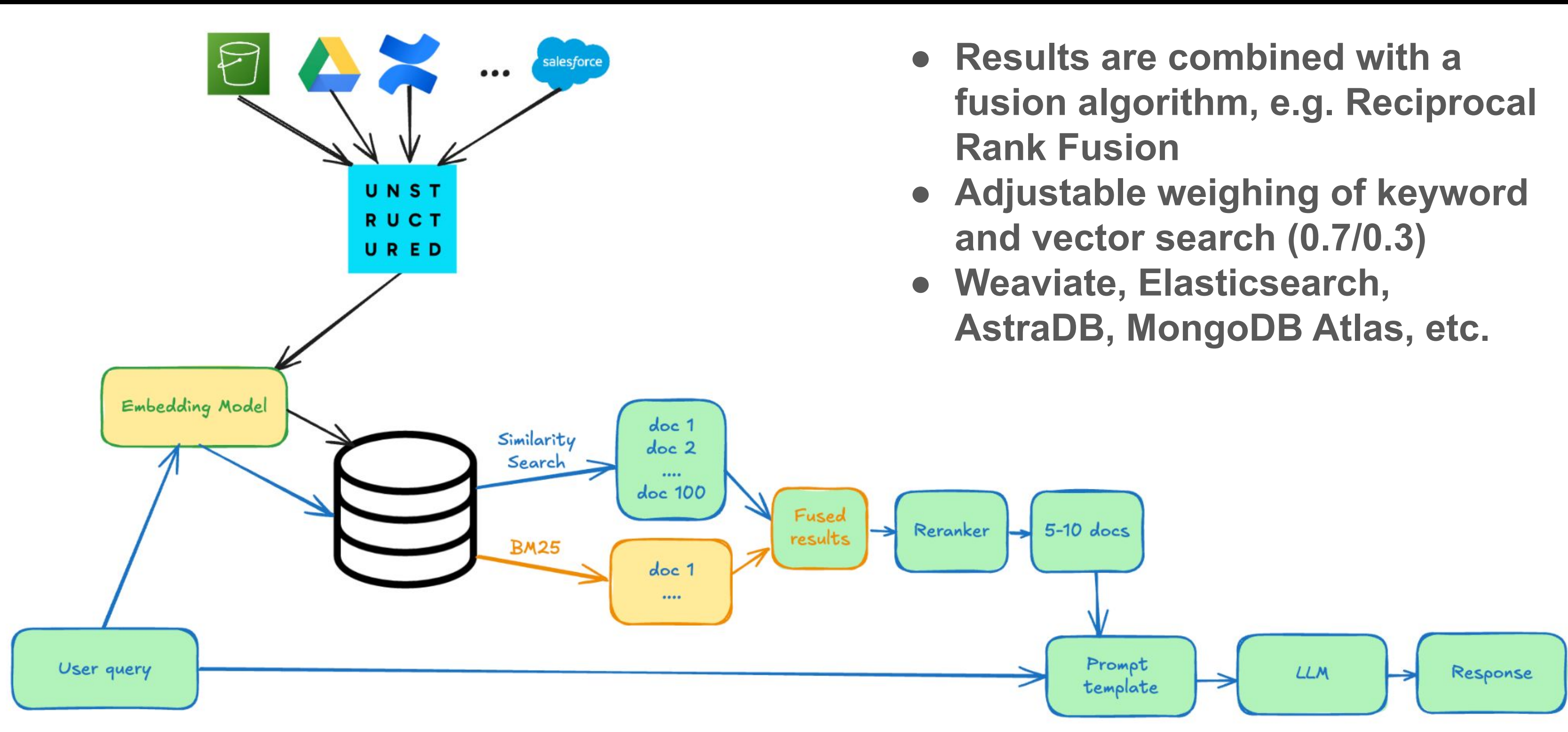
**Especially powerful in use cases with lots of jargon, acronyms, internal terms, etc.**

**Simple, fast, easily scales**

**Doesn't capture semantic meaning**

**Can miss relevant docs due to vocabulary mismatch**

# Hybrid Search: similarity search + BM25



# Metadata filters

**Query: “What was Apple’s revenue in 2023?”**

**Retrieved chunks:**

**#1: “Net revenues were \$46.25 billion for 2023, 2% lower than 2022.”**

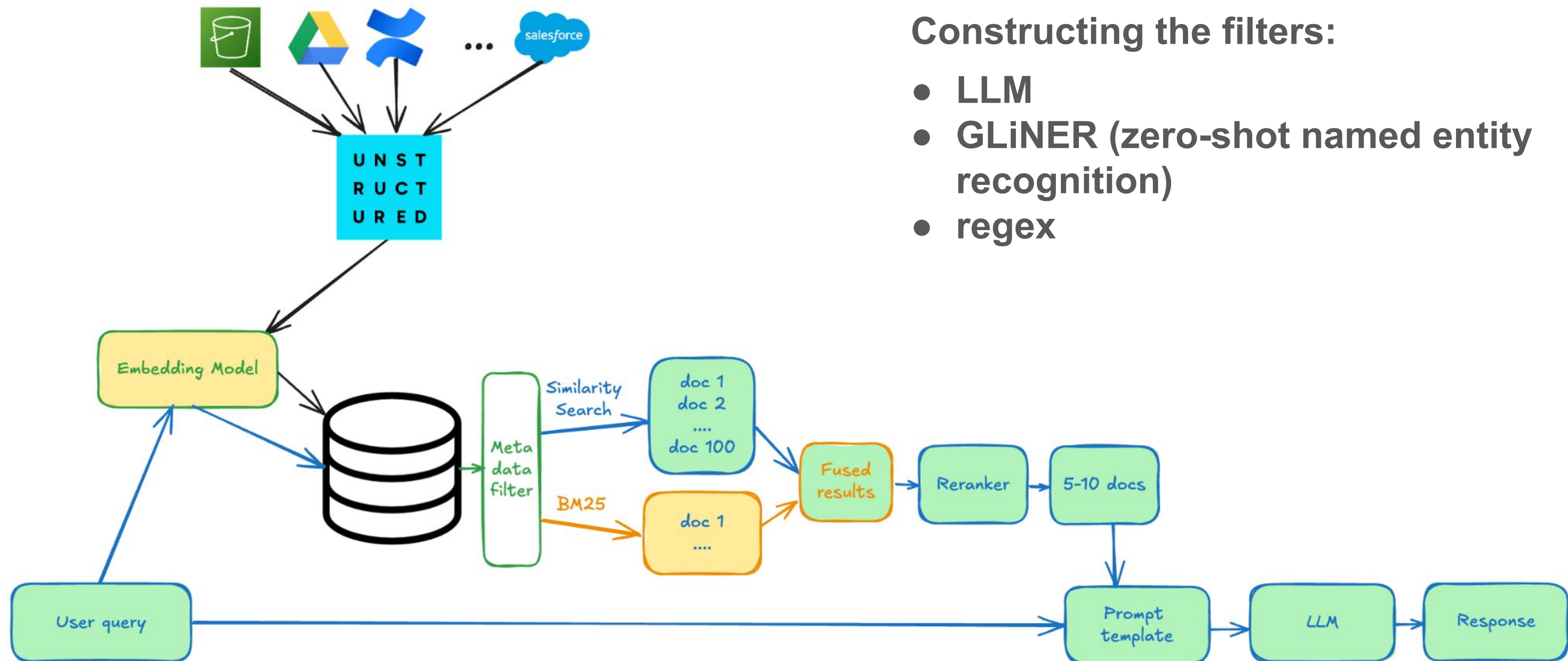
**#2: “During fiscal 2023, we generated total revenues of \$611.3 billion, which was comprised primarily of net sales of \$605.9 billion”**

# Metadata filters

- Real data doesn't exist in vacuum, you can get metadata - where the document is coming from, when it was updated, who authored it. Custom metadata is not difficult to add (e.g. company name, fiscal year)
- Use metadata to narrow down the search & improve retrieval precision
- Use metadata to get recent results



# Metadata filters



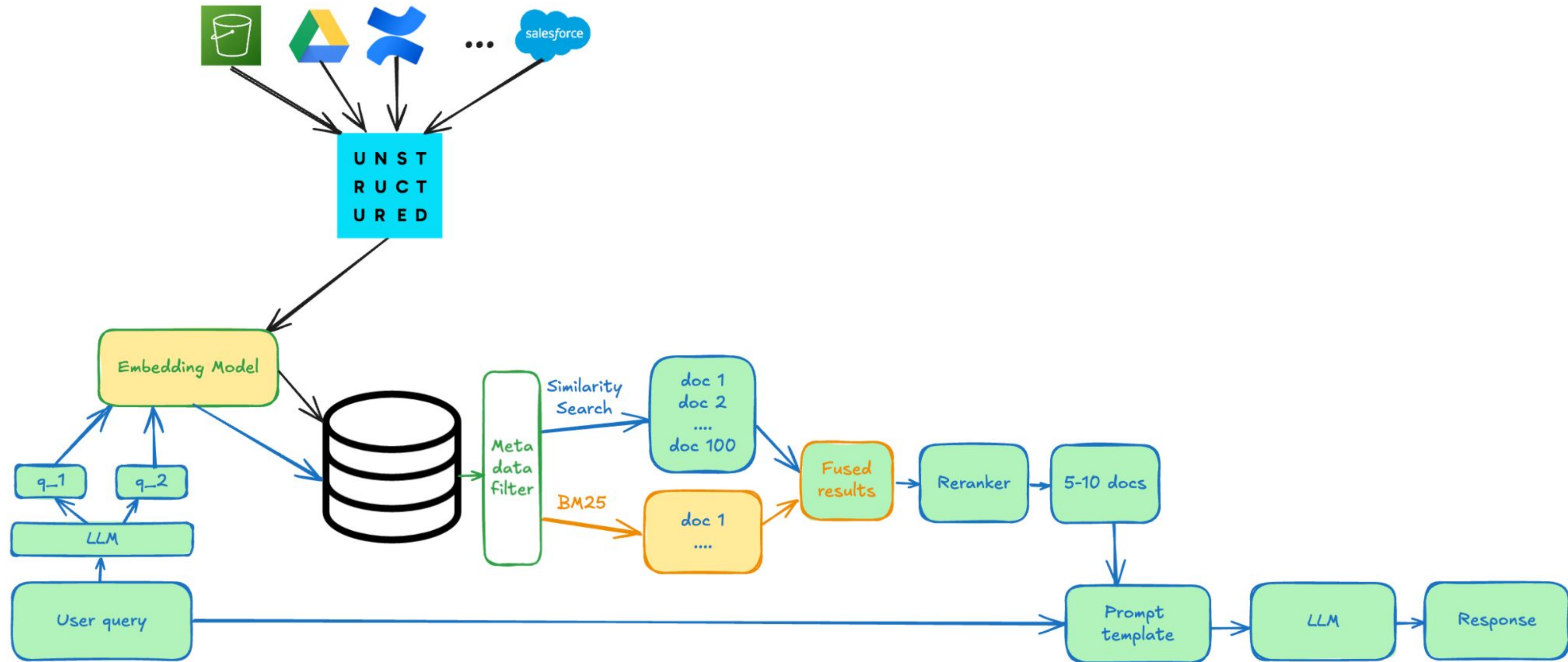
# Query decomposition

● Query: “How do employee incentive plans differ between Chevron and Walmart?”

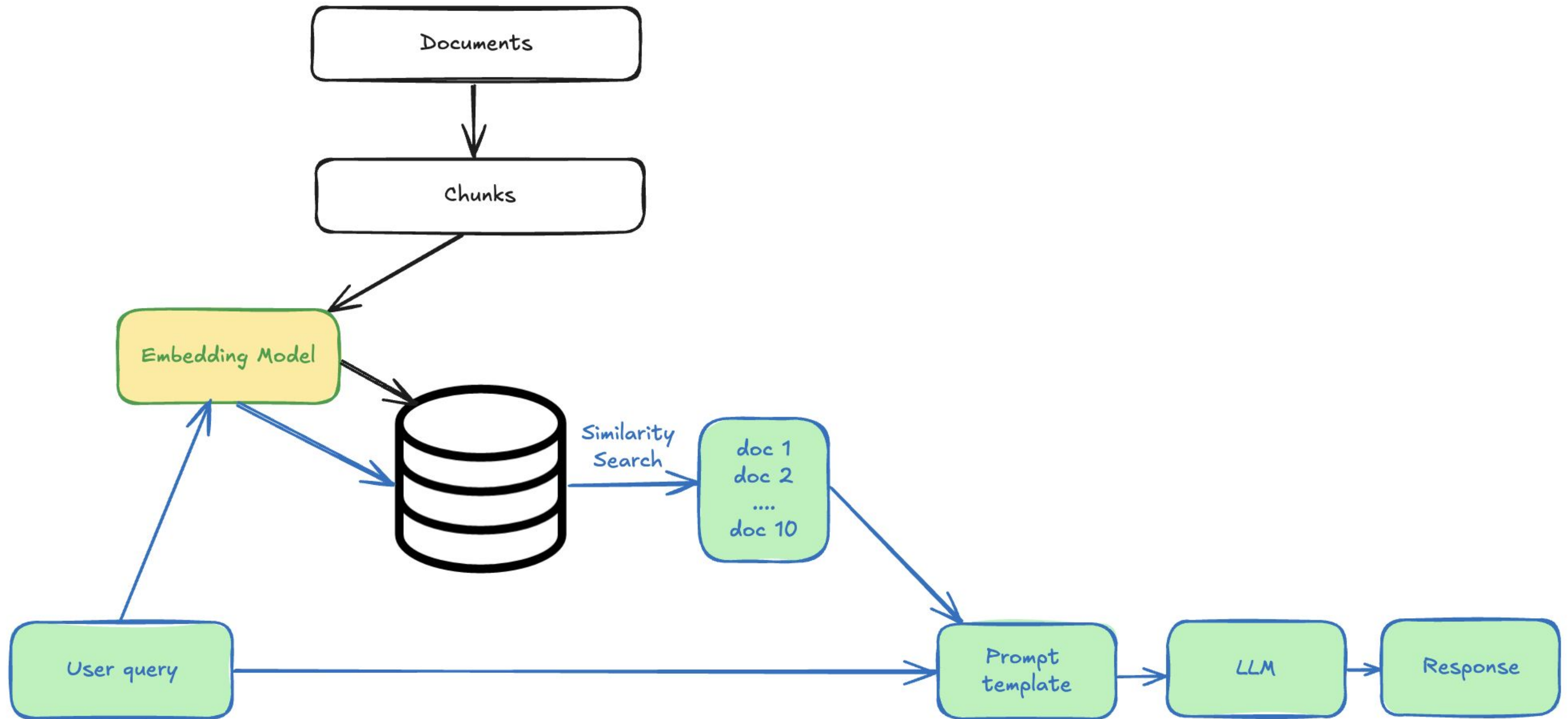
● Instead:

- Query\_1: “Chevron employee incentive plans”
- Query\_2: “Walmart employee incentive plans”
- Combine the results

# Query decomposition

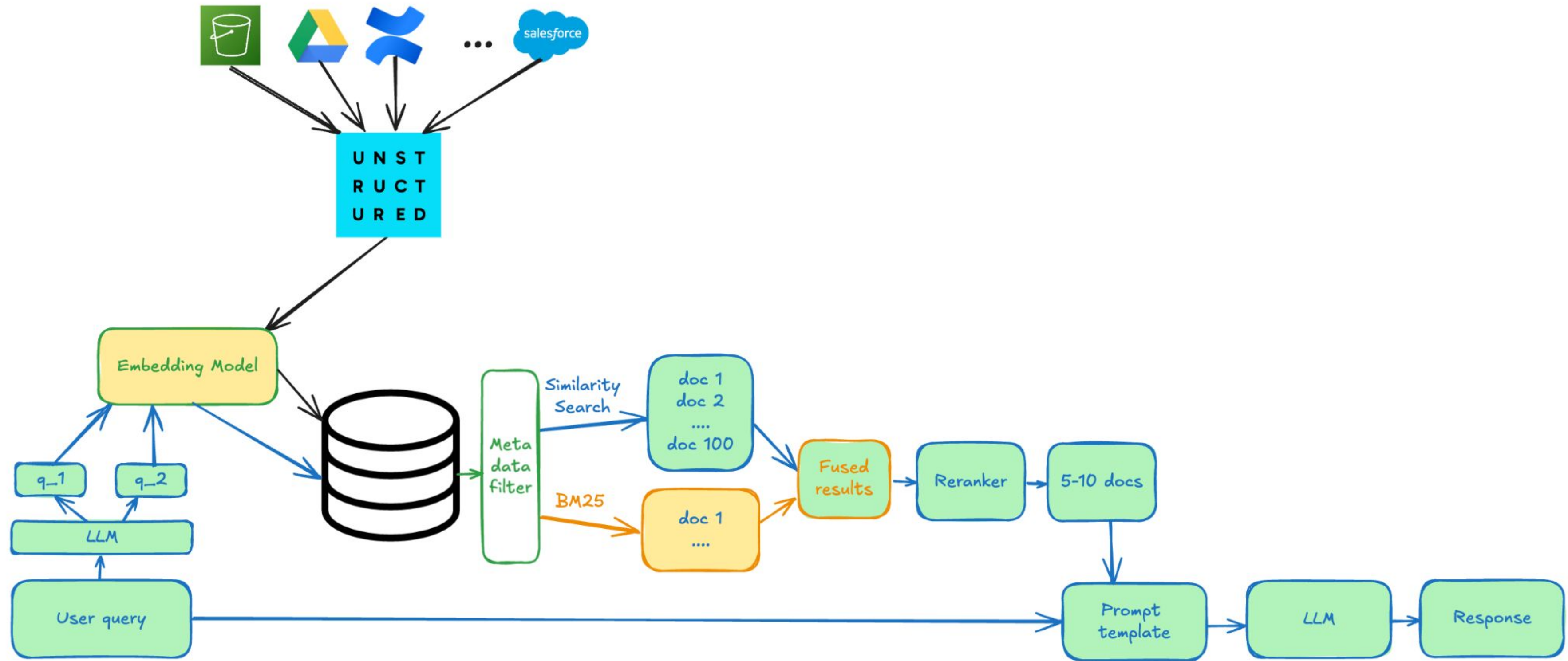


# Where we started





# Where we are



# Trends: Agentic RAG & GraphRAG

**Agentic RAG = retriever is a tool available to your LLM agent. LLM can:**

- rephrase, decompose queries
  - construct metadata filters
  - review results & retrieve more context
  - use additional tools (calculator, web search, run code, etc.)
  - and more
- **Pros:** Flexible, powerful, often higher quality of responses
- **Cons:** Requires a powerful LLM for best results, increases cost & latency.


**Mistakes compound - requires validation and error correction mechanisms.**

# Trends: Agentic RAG & GraphRAG

**GraphRAG = retrieval is enhanced using a knowledge graph. Retriever can**

- **Get broader context based on structured relationships which may be missed by both similarity search and BM25**

 **Pros: better contextual understanding, improved logical reasoning**

 **Cons: requires special data preprocessing to build knowledge graphs, requires special databases (e.g. neo4j)**

# Evals are a must: can't improve what you don't measure

- Vibe check is not enough
- Always look at your data
- Evaluate individual components and system as a whole
- Measure recall for retrieval
- Use LLM-as-a-Judge to evaluate generated answers

# RAG vs Long-Context Window Models

- **Handles large volumes of data:**  
*2M token window can only fit about 10-15 standard annual financial reports*
- **Has better answers precision for pointed questions:**  
*Long-context models (e.g. Gemini 2.0) struggle with needle-in-haystack problem*
- **Cost-effective and scales:**  
*Lower token usage & compute cost, potentially lower latency (depends on your RAG)*
- **Can incorporate diverse data:**  
*Structured and unstructured data can be preprocessed into a vector store*
- **Offers explainability:**  
*RAG offers citations, links to sources, vital in industries like healthcare, legal, etc.*
- **Enables Access Control:**  
*RAG allows role-based data access*
- **Supports Advanced AI:**  
*RAG is a crucial component for Agentic systems, report generators, not just chatbots*



# Resources:

- Chunking for RAG: [unstructured.io/blog/chunking-for-rag-best-practices](https://unstructured.io/blog/chunking-for-rag-best-practices)
- Understanding embedding models:  
[unstructured.io/blog/understanding-embedding-models-make-an-informed-choice-for-your-rag](https://unstructured.io/blog/understanding-embedding-models-make-an-informed-choice-for-your-rag)
- Building an Advanced RAG System With Self-Querying Retrieval (metadata filters):  
[www.mongodb.com/developer/products/atlas/advanced-rag-self-querying-retrieval/](https://www.mongodb.com/developer/products/atlas/advanced-rag-self-querying-retrieval/)
- Creating a LLM-as-a-Judge by Hamel Husain: [hamel.dev/blog/posts/llm-judge/](https://hamel.dev/blog/posts/llm-judge/)
- Notebooks with RAG examples: [docs.unstructured.io/examplecode/notebooks](https://docs.unstructured.io/examplecode/notebooks)
- RAG vs. Long-Context:  
[unstructured.io/blog/rag-vs-long-context-models-do-we-still-need-rag](https://unstructured.io/blog/rag-vs-long-context-models-do-we-still-need-rag)
- Agents by Chip Huyen: [huyenchip.com/2025/01/07/agents.html](https://huyenchip.com/2025/01/07/agents.html)

# Q&A

**Burning questions? Hot takes? Not a question but a comment?**

**Send them my way:**

**Twitter/X: @mariaKhalusova**

**LinkedIn: Maria Khalusova**

**BlueSky: @mariak.bsky.social**

*(Carrier pigeons and smoke signals also accepted, but response is not guaranteed)*