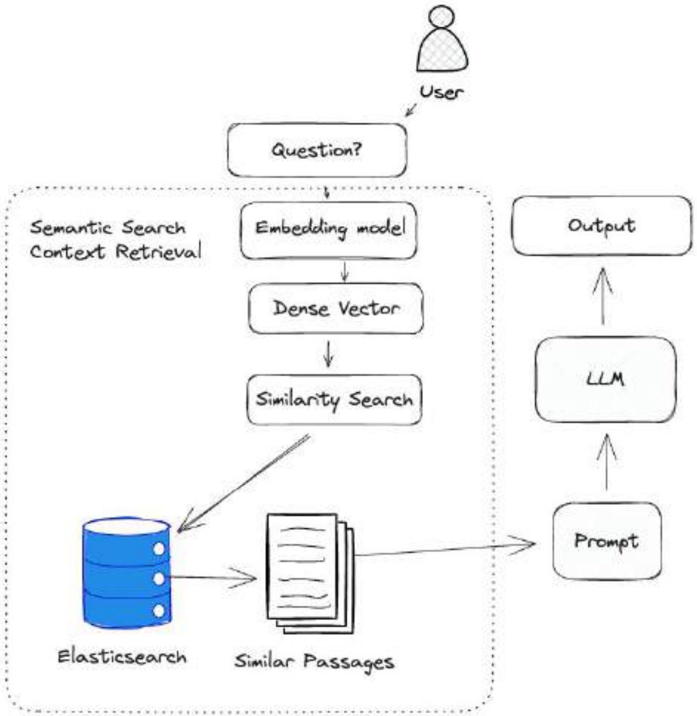
# From Vectors to Agents: Managing RAG in an Agentic World

Rajiv Shah
Chief Evangelist, Contextual AI
rajiv.shah@contextual.ai

https://github.com/rajshah4/LLM-Evaluation


ACME
GPT

# Building RAG is Easy



# Building RAG is Easy

```
docs = TextLoader("docs/sample.txt").load()
chunks = RecursiveCharacterTextSplitter(chunk_size=800).split_documents(docs)
vdb = FAISS.from_documents(chunks, OpenAIEmbeddings())
retriever = vdb.as_retriever(search_kwargs={"k": 4})
llm = ChatOpenAI(model="gpt-4o-mini", temperature=0)

prompt = ChatPromptTemplate.from_messages([
    ("system", "Answer only using the context below."),
    ("human", "Q: {question}\n\nContext:\n{context}\n\nA:")
])

rag_chain = (
    {"context": retriever | (lambda d: "\n\n".join(x.page_content for x in
d)), "question": RunnablePassthrough()}
    | prompt | llm | StrOutputParser()
)

print(rag_chain.invoke("What warranty terms are mentioned?"))
```

@rajistics

# RAG Reality Check

**95%**

of Gen AI projects fail to reach Production

### Accuracy
**<70%**
fails beyond simple extraction

### Latency
**>45s**
queries are too slow

### Scaling
**>1,000**
fails with more documents

### Cost
**100x**
complex queries more token use

### Compliance
**0%**
access control over documents

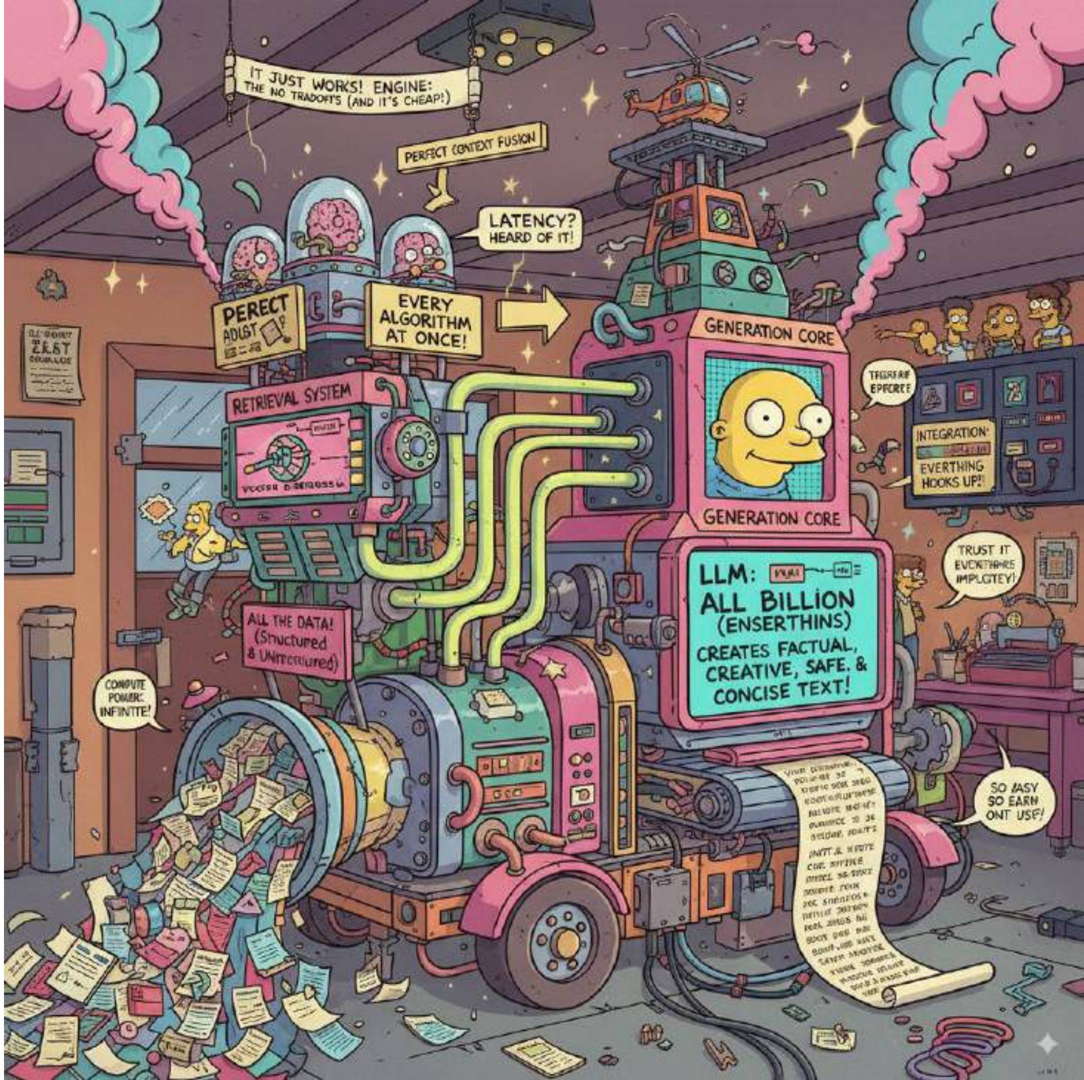
https://www.zeta-alpha.com/post/why-genai-pilots-fail-common-challenges-with-enterprise-rag

@rajistics

# Maybe try a different RAG?

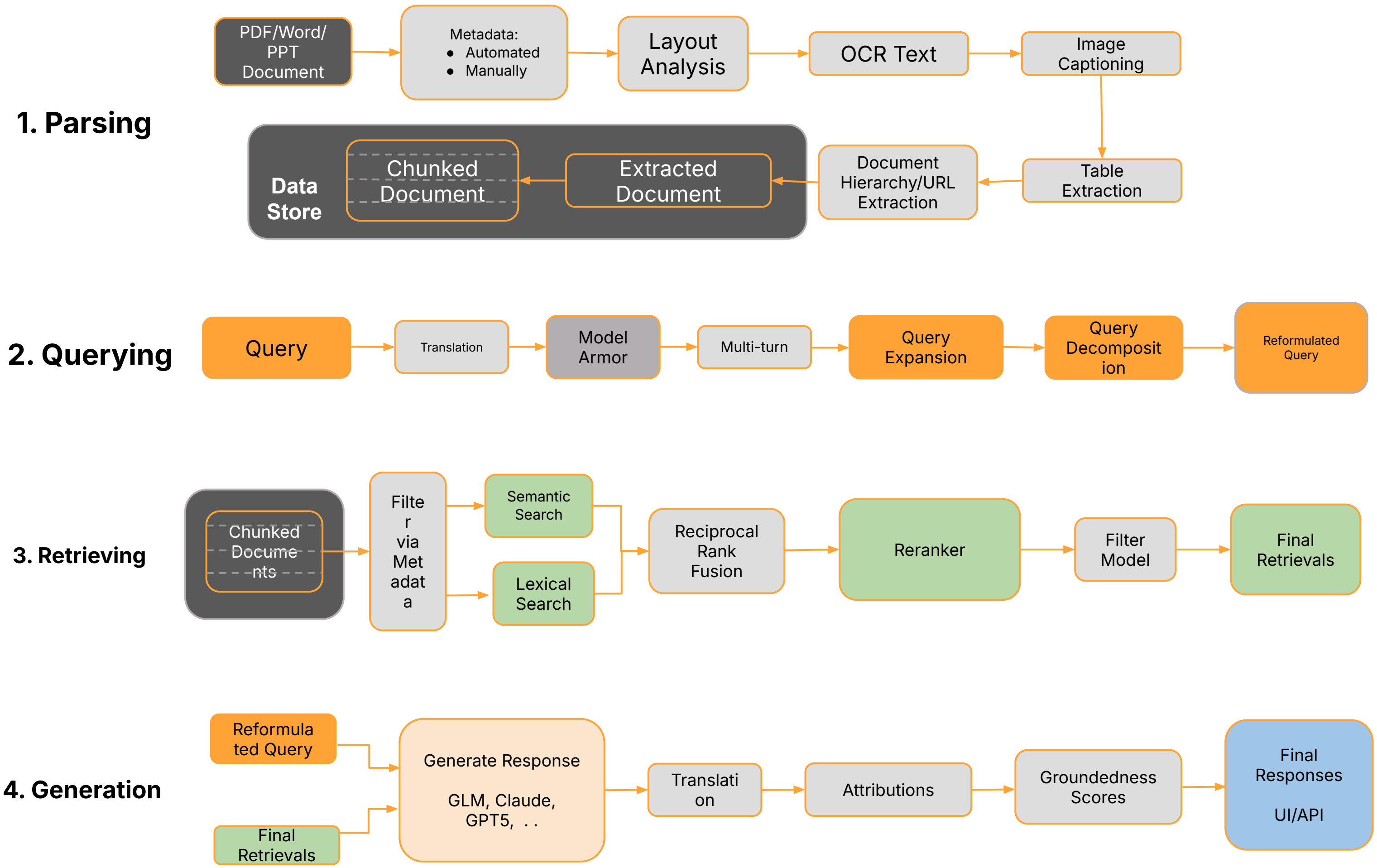
Basic RAG, Reliable RAG, HyDE (Hypothetical Document Embedding), HyPE (Hypothetical Prompt Embedding), Contextual Chunk Headers, Semantic Chunking, Contextual Compression, Document Augmentation, Fusion Retrieval, Reranking, Multi-faceted Filtering, Hierarchical Indices, Ensemble Retrieval, Dartboard Retrieval, Multi-modal RAG with Captioning, Retrieval with Feedback Loop, Adaptive Retrieval, Iterative Retrieval, DeepEval, GroUSE, Explainable Retrieval, Graph RAG with LangChain, Microsoft GraphRAG, RAPTOR, Self-RAG, Corrective RAG (CRAG), Sophisticated Controllable Agent, Vision-RAG, Cache-Augmented Generation (CAG), Agentic RAG, Retrieval-Augmented Fine-Tuning (RAFT), Self-Reflective RAG, RAG Fusion, Temporal Augmented Retrieval (TAR), Plan-then-RAG (PlanRAG), GraphRAG, FLARE, Contextual Retrieval, GNN-RAG

# Ultimate RAG Solution



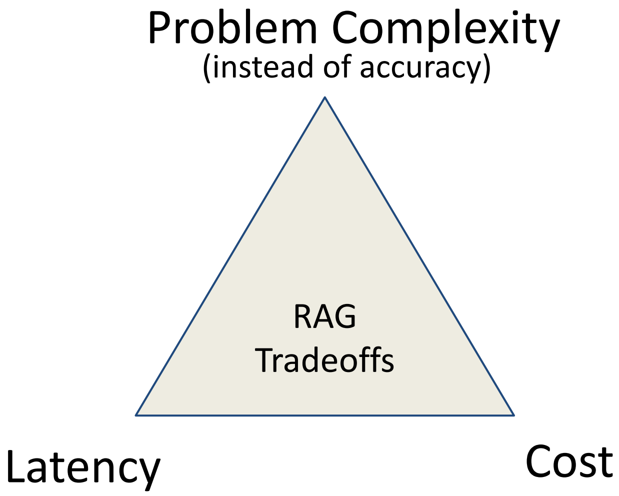

@rajistics

# RAG as a system




@rajistics

# Designing a RAG Solution




Practical:
Cost of a mistake

# RAG Considerations

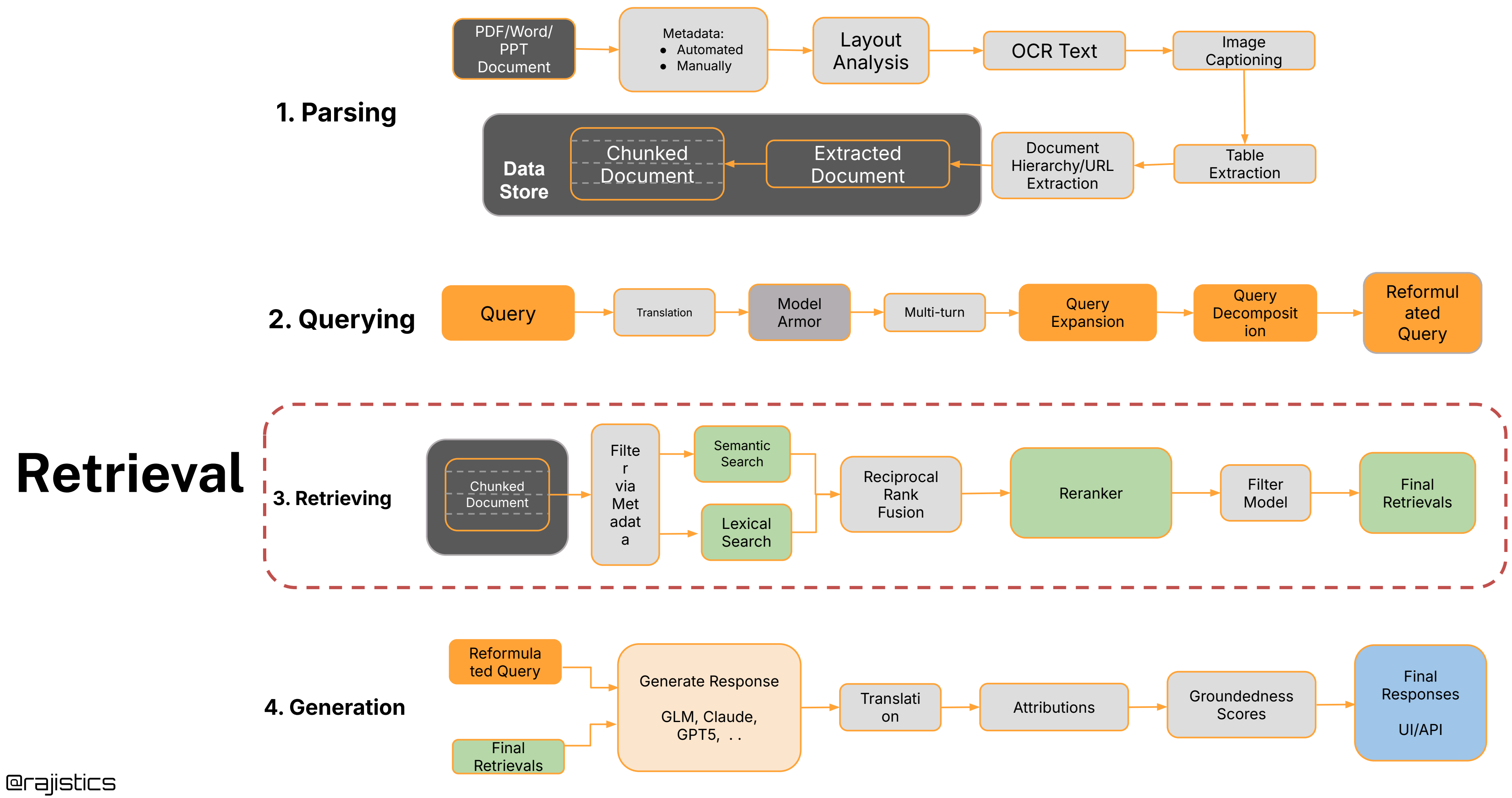
- Extraction
- Latency
- Amount of Queries
- Multilingual
- Domain difficulty
- Data Quality

| Generation → | 1. Simple Fact | 2. Summarization | 3. Multi-Source Synthesis | 4. Deep Reasoning/Analysis |
|---|---|---|---|---|
| **Retrieval 1: Single-hop** | Basic factual Q&A | Short doc summary | Summarize from 2–3 texts | Single-hop but deep reasoning |
| **Retrieval 2: Multi-hop** | Factual, but requires combining 2 steps to retrieve | Summaries that rely on multi-step retrieval | Synthesize multi-doc, multi-hop context | Multi-hop with multi-step logic in generation |
| **Retrieval 3: Cross-domain** | Straight pass-through, but from different data sources | Summaries that span multiple domains (e.g., news + scientific articles) | Cross-domain synthesis (e.g., financial + technical) | Complex reasoning across domain boundaries |
| **Retrieval 4: Ambiguous / advanced** | Passing through uncertain context or ambiguous queries | Summaries that handle contradictory / ambiguous sources | Complex bridging across ambiguous queries + multi-sources | Highest difficulty: multi-hop + cross-domain + advanced reasoning |

@rajistics

# Consider Query Complexity

| | |
|---|---|
| Simple Keyword | 1. What is Tesla's total revenue in 2023? |
| Semantic variation | 2. How much bank did Tesla make last year from its operations? |
| Multi-hop | 3. Compare Tesla's revenue growth in 2023 with Rivian's net loss in the same year. |
| Cross-document | 4. Summarize how EV companies described supply chain issues in their 2023 filings. |
| Out of corpus | 5. In Rivian's 10-K, they mention compliance with the Clean Air Act. What specific obligations does this impose on them? |
| Agentic scenario | 6. If I were evaluating Rivian's environmental liabilities, how do the obligations under the Clean Air Act and California's Zero Emission Vehicle mandate intersect with the risks they disclosed in their last two annual reports?" |


1. Parsing
PDF/Word/ PPT Document
Metadata:
• Automated
• Manually
Layout Analysis
OCR Text
Image Captioning
Table Extraction
Document Hierarchy/URL Extraction
Extracted Document
Chunked Document
Data Store
2. Querying
Query
Translation
Model Armor
Multi-turn
Query Expansion
Query Decomposition
Reformulated Query
Retrieval
3. Retrieving
Chunked Document
Filter via Metadata
Semantic Search
Lexical Search
Reciprocal Rank Fusion
Reranker
Filter Model
Final Retrievals
4. Generation
Reformulated Query
Final Retrievals
Generate Response
GLM, Claude, GPT5, . .
Translation
Attributions
Groundedness Scores
Final Responses
UI/API
@rajistics

# Retrieval Approaches

**BM25**
Keyword-based retrieval

**Language Models**
Semantic meaning with embeddings

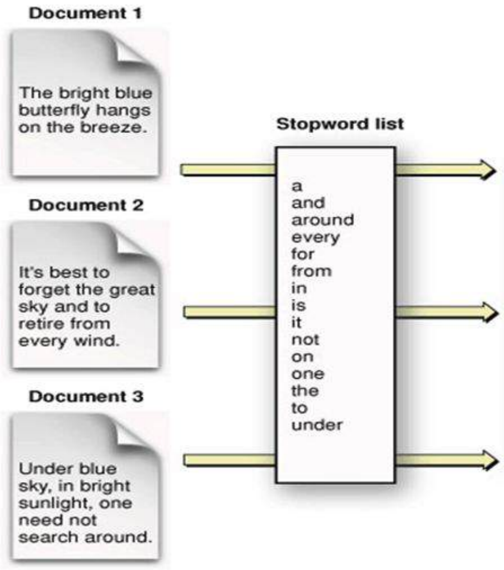
**Agentic Search**
Dynamic using LLM Reasoning

# Building RAG is Easy

```
docs = TextLoader("docs/sample.txt").load()
chunks = RecursiveCharacterTextSplitter(chunk_size=800).split_documents(docs)
vdb = FAISS.from_documents(chunks, OpenAIEmbeddings())
retriever = vdb.as_retriever(search_kwargs={"k": 4})
llm = ChatOpenAI(model="gpt-4o-mini", temperature=0)

prompt = ChatPromptTemplate.from_messages([
    ("system", "Answer only using the context below."),
    ("human", "Q: {question}\n\nContext:\n{context}\n\nA:")
])

rag_chain = (
    {"context": retriever | (lambda d: "\n\n".join(x.page_content for x in
d)), "question": RunnablePassthrough()}
    | prompt | llm | StrOutputParser()
)

print(rag_chain.invoke("What warranty terms are mentioned?"))
```

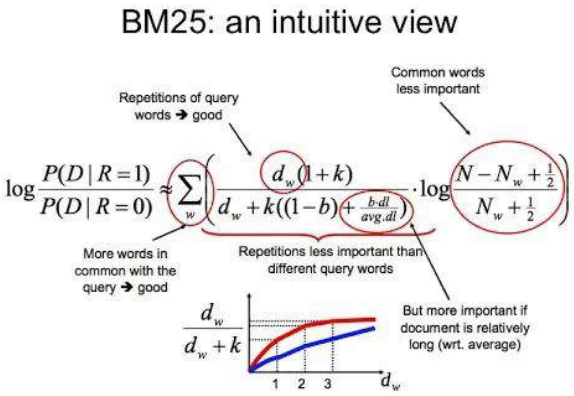
@rajistics

# BM25




**Inverted index**

| ID | Term | Document |
|---|---|---|
| 1 | best | 2 |
| 2 | blue | 1, 3 |
| 3 | bright | 1, 3 |
| 4 | butterfly | 1 |
| 5 | breeze | 1 |
| 6 | forget | 2 |
| 7 | great | 2 |
| 8 | hangs | 1 |
| 9 | need | 3 |
| 10 | retire | 2 |
| 11 | search | 3 |
| 12 | sky | 2, 3 |
| 13 | wind | 2 |




Probabilistic lexical ranking function

# BM25 Performance

- Keyword precision
- Efficient at scale
- Battle-tested

| N_docs | Linear (s) | Inverted Index | BM25 (s) |
|---|---|---|---|
| 1000 | 3.468 | 0.005 | 0.028 |
| 3000 | 10.188 | 0.014 | 0.097 |
| 6000 | 20.608 | 0.025 | 0.24 |
| 9000 | 30.092 | 0.061 | 0.36 |

## BM25 Failure Cases

Lexical, probabilistic matching can mis-rank when meaning diverges from exact word overlap.

### Synonym Gap (Vocabulary Mismatch)

Query: "physician salary cap policy"

Doc A (relevant): "Doctor compensation limits for hospitals..."

Doc B (distractor): "Company salary cap policy for managers..." ❌

BM25 Overlaps on “salary”, “cap”, “policy” → often ranks Doc B above Doc A.

Why: No shared token for physician↔doctor.

### Aliases & Abbreviations

Query: "International Business Machines layoffs 2024"

Doc A (relevant): "IBM announced workforce reductions in 2024..."

Doc B (distractor): "International business trends show slower layoffs..." ❌

BM25 Matches literal words → Doc B can outrank Doc A.

Why: Misses IBM ≡ International Business Machines.

Takeaway:

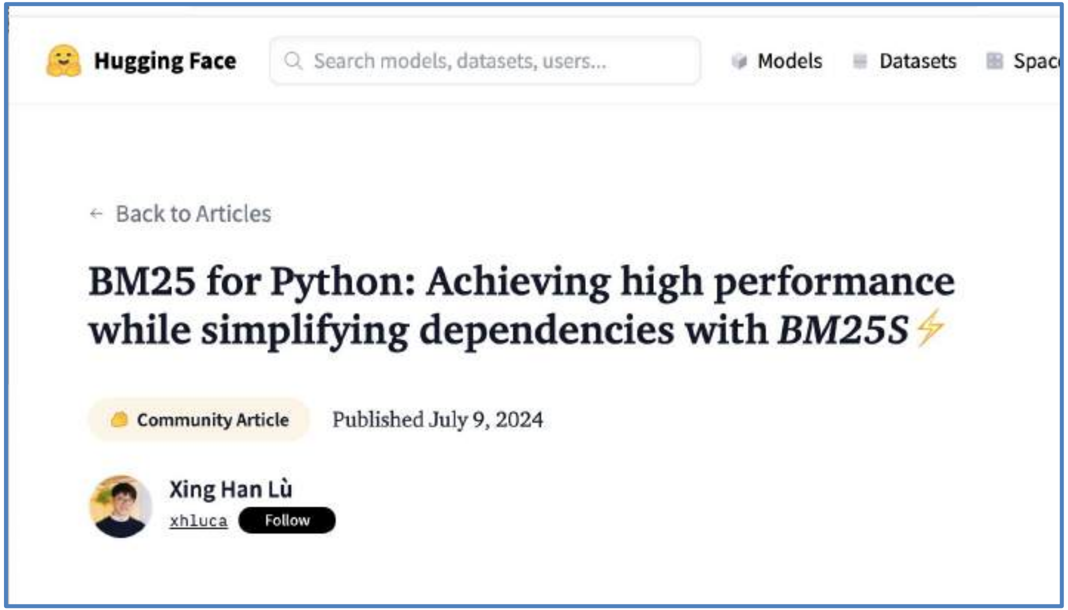
BM25 is a strong baseline

If you have keyword-heavy queries and need sub-second response →

BM25 might be sufficient

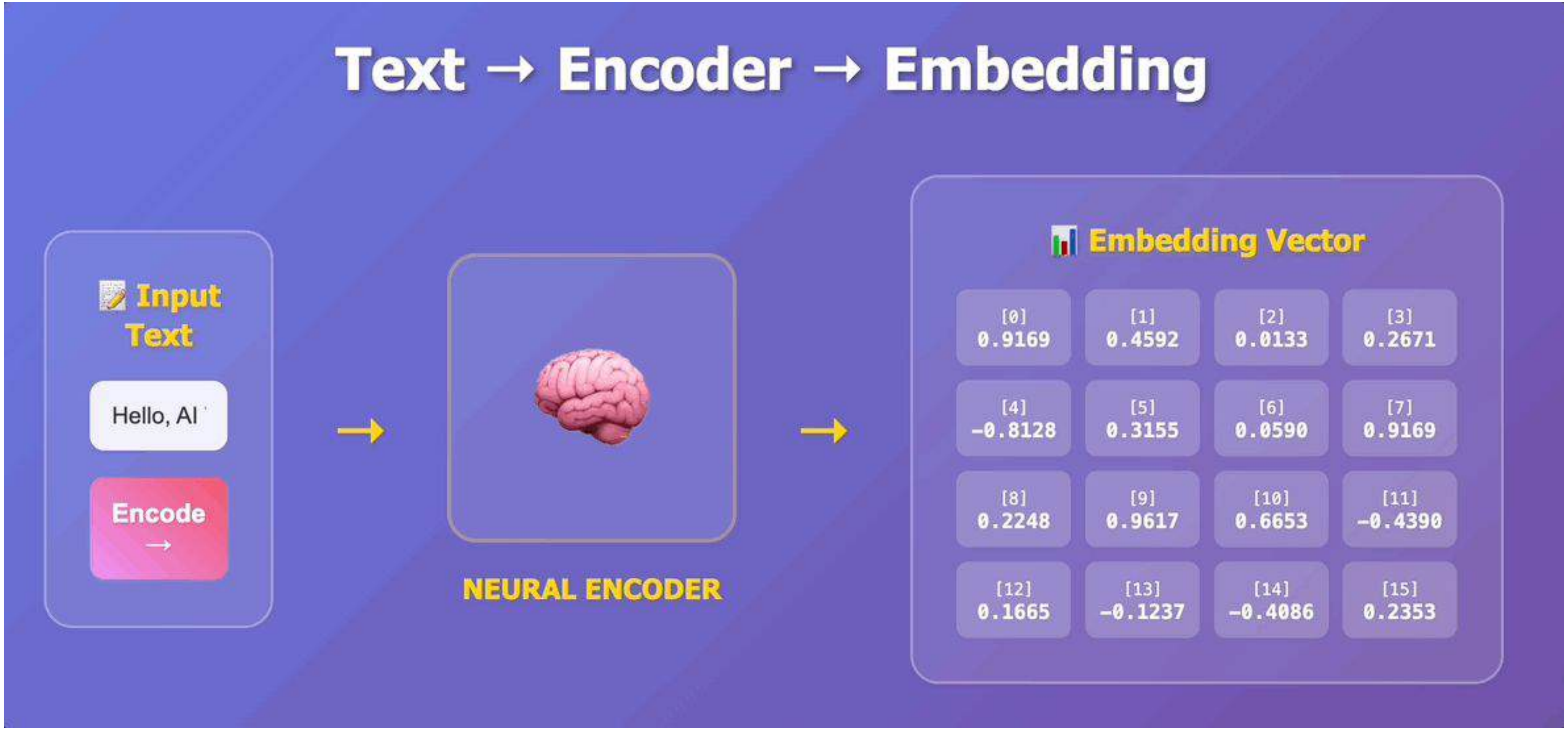
# Hands on: BM25s

- Fast lexical search implementing BM25 in Python using Numpy, Numba and Scipy

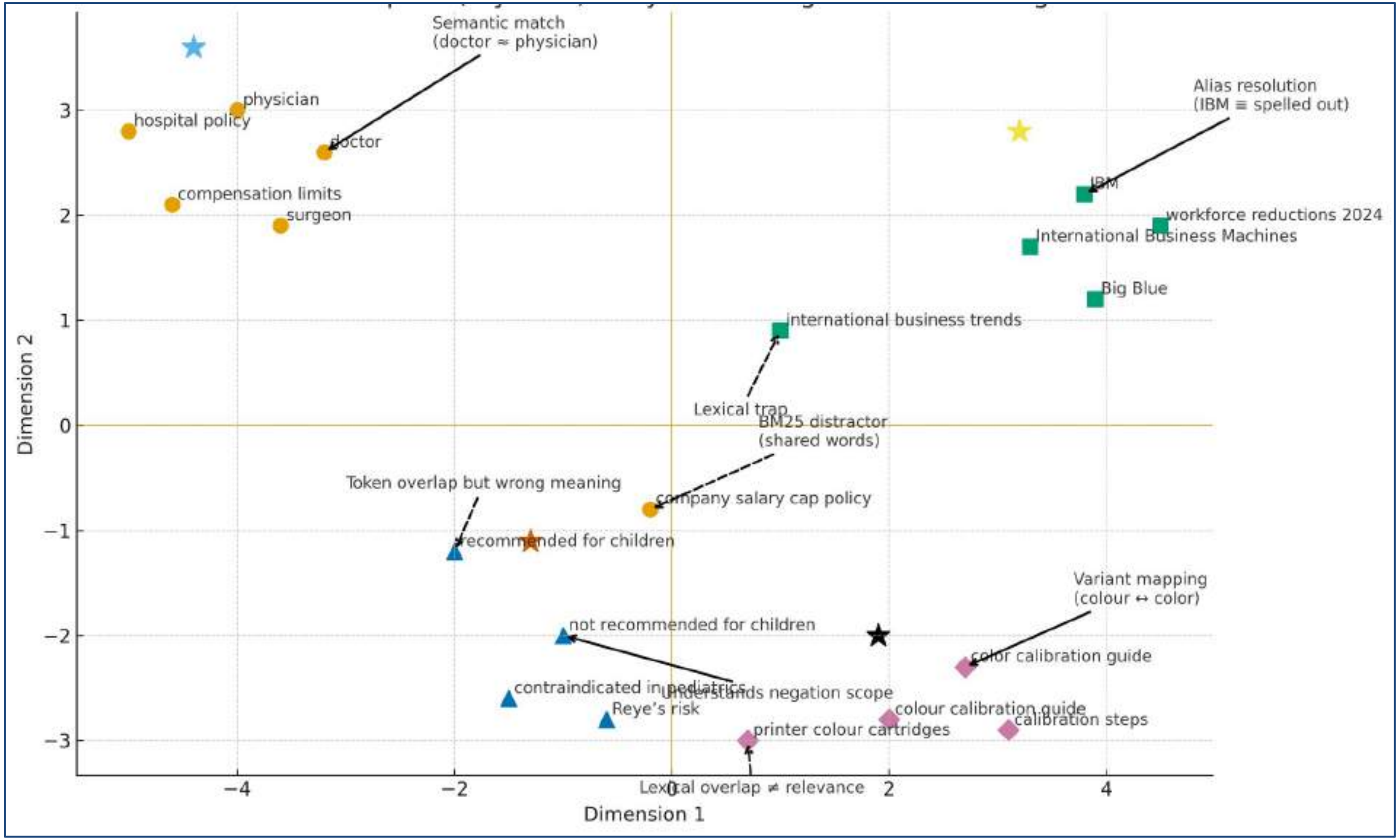



https://github.com/xhluca/bm25s

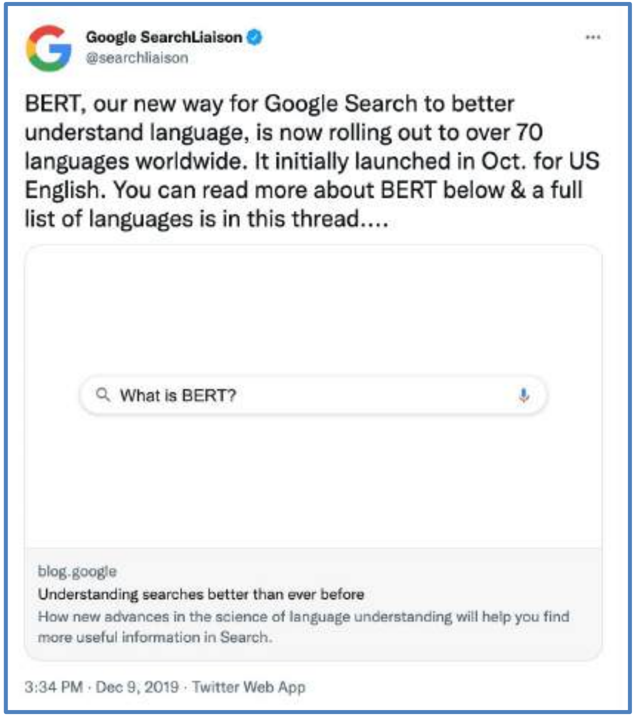
# Enter Language Models

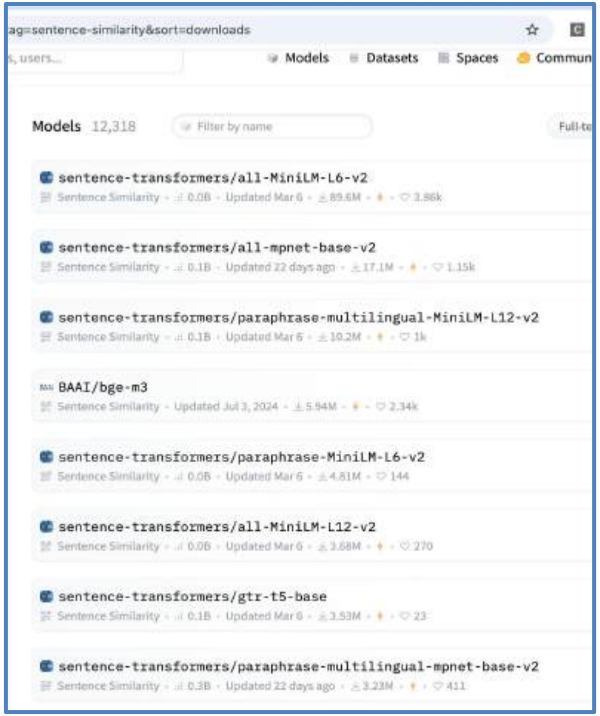


# Embeddings Visualized

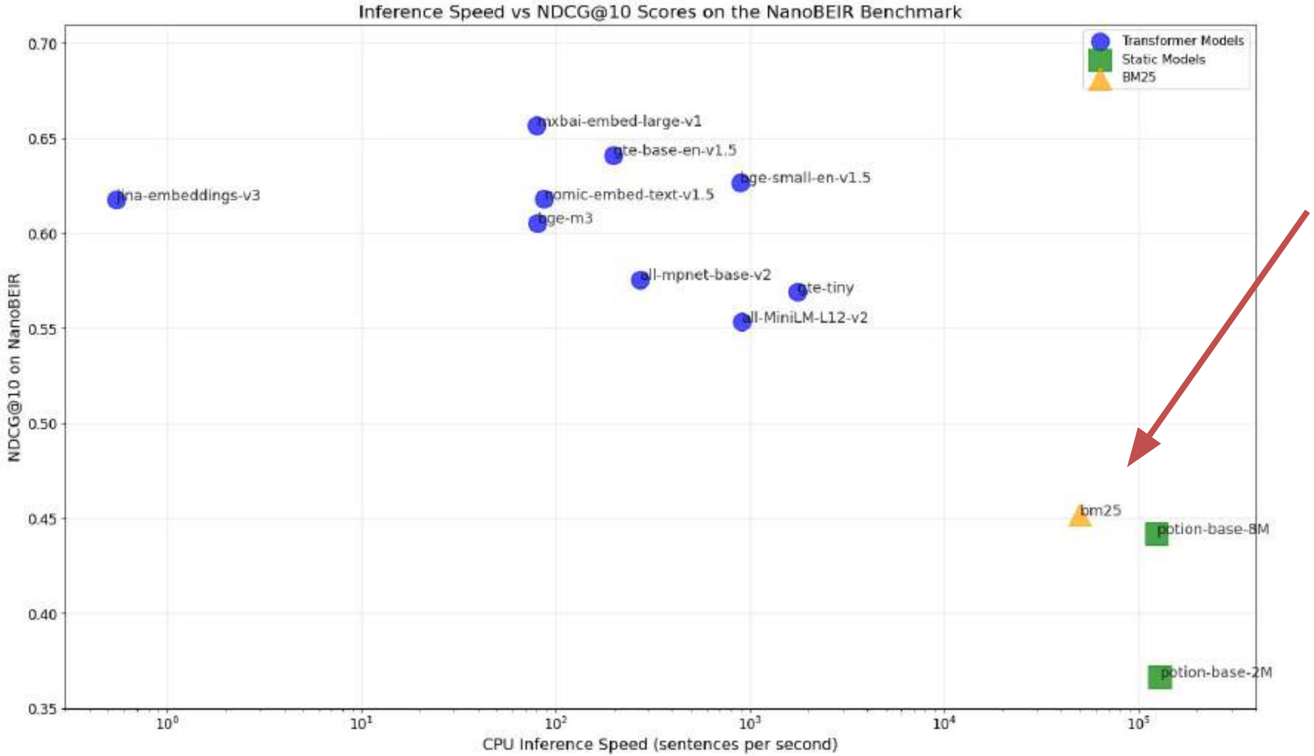


# Semantic search is widely used



# Which language model?




@rajistics

https://huggingface.co/blog/static-embeddings

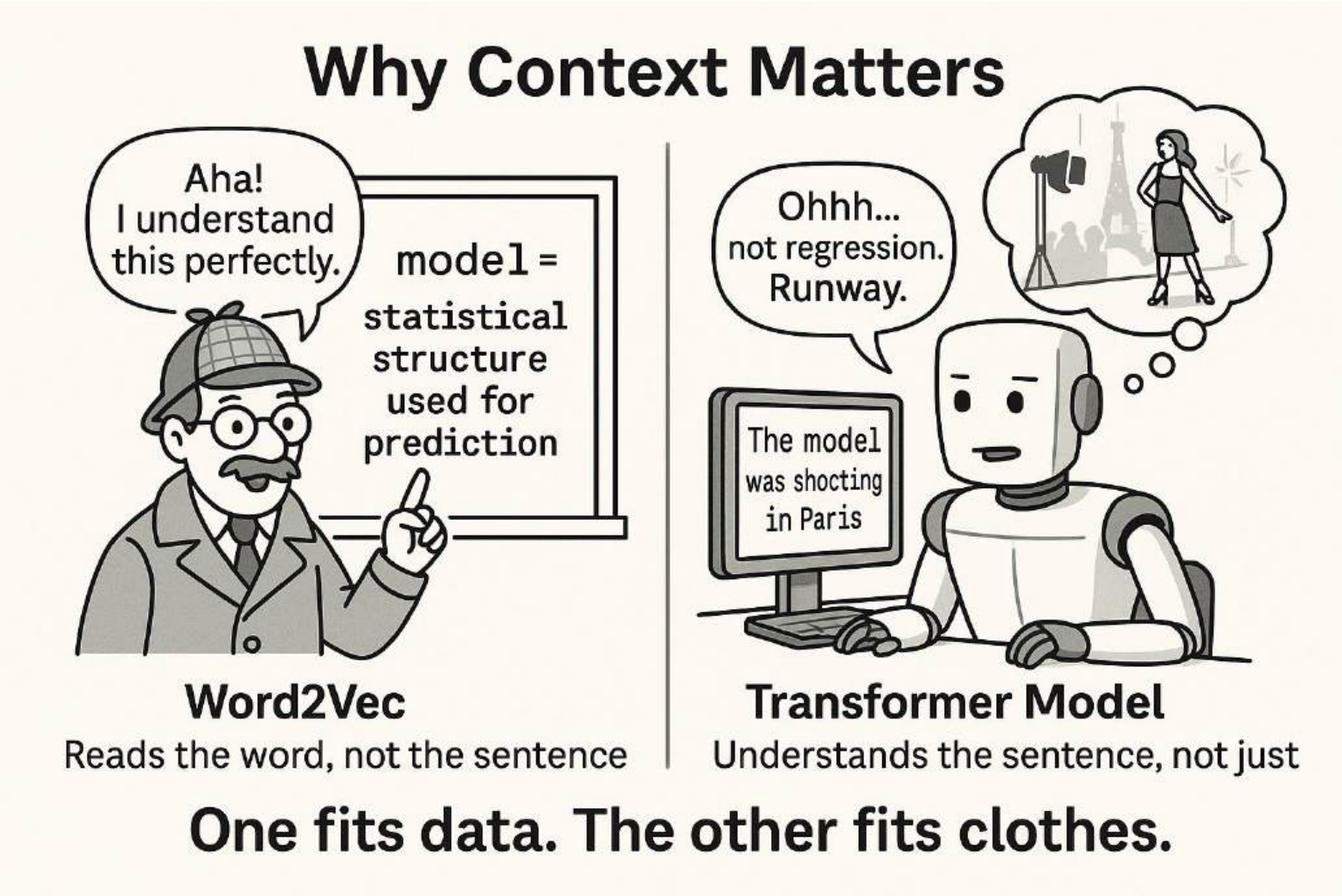
# Static Embeddings


Older versions: FastText, Word2Vec, Glove

- Uncontextualized
  - less accuracy
- Fast
- Lightweight CPU

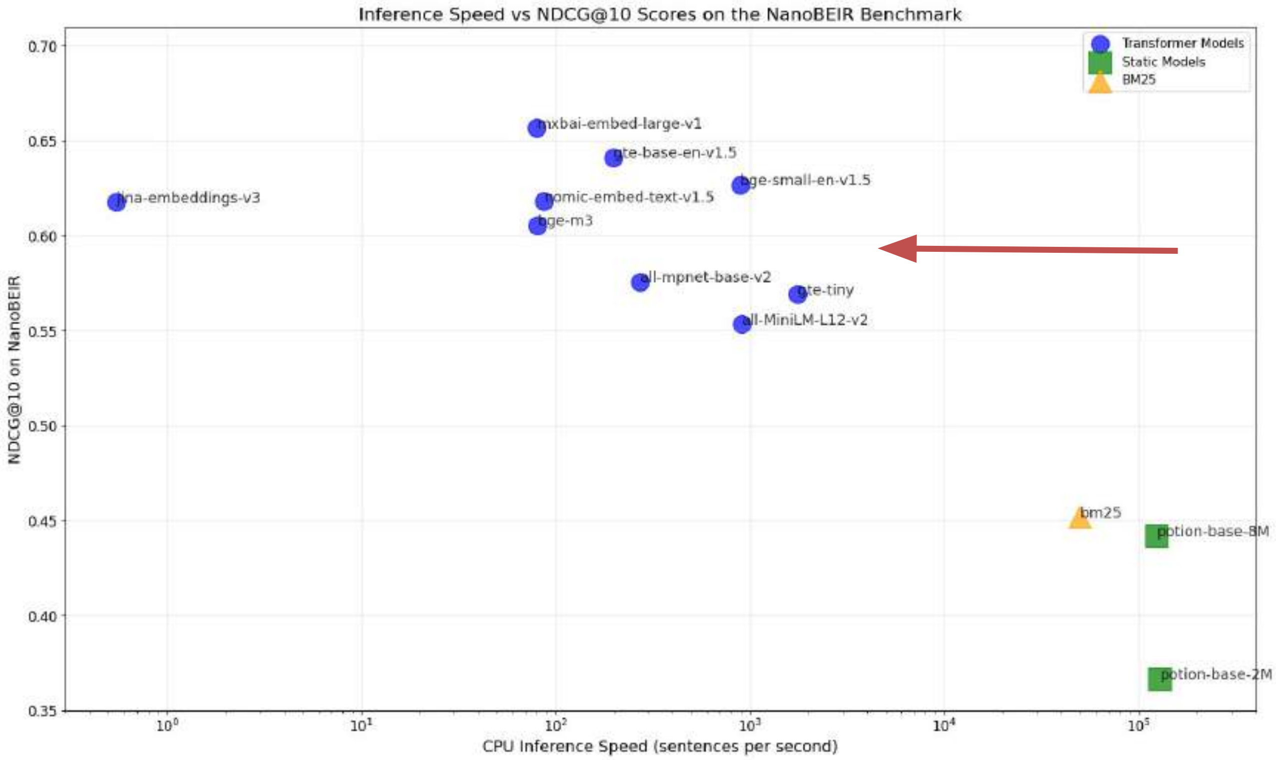


https://github.com/MinishLab/model2vec
https://projector.tensorflow.org/


Why Context Matters
Aha!
I understand
this perfectly.
model =
statistical
structure
used for
prediction
Ohhh...
not regression.
Runway.
The model
was shocting
in Paris
Word2Vec
Reads the word, not the sentence
Transformer Model
Understands the sentence, not just
One fits data. The other fits clothes.

@rajistics

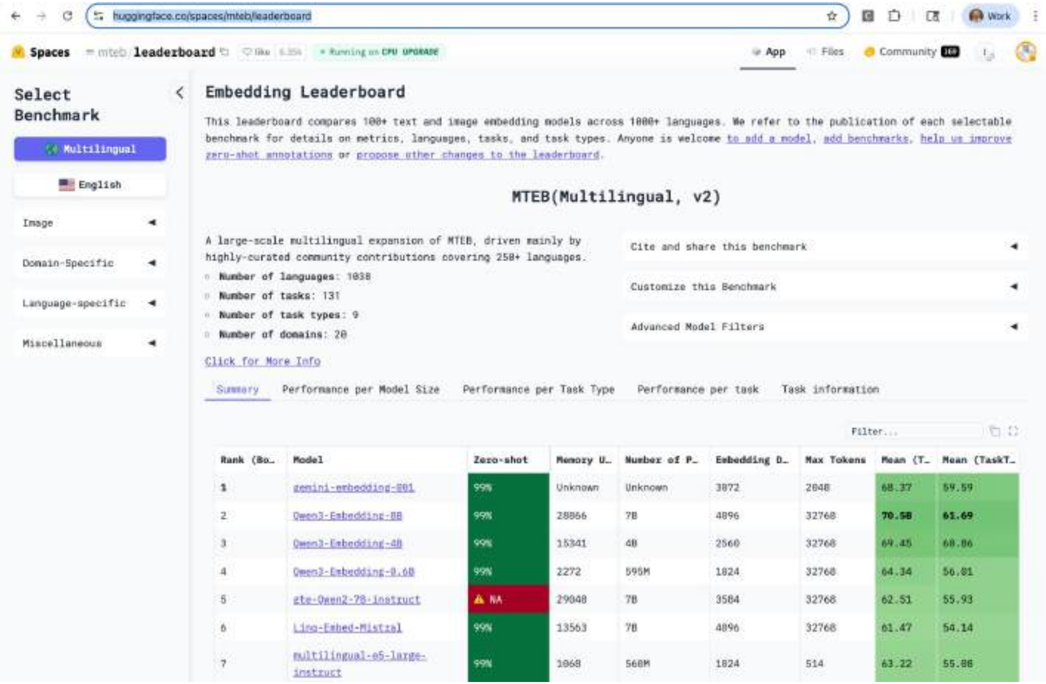
# Many more models!




@rajistics

# MTEB/RTEB

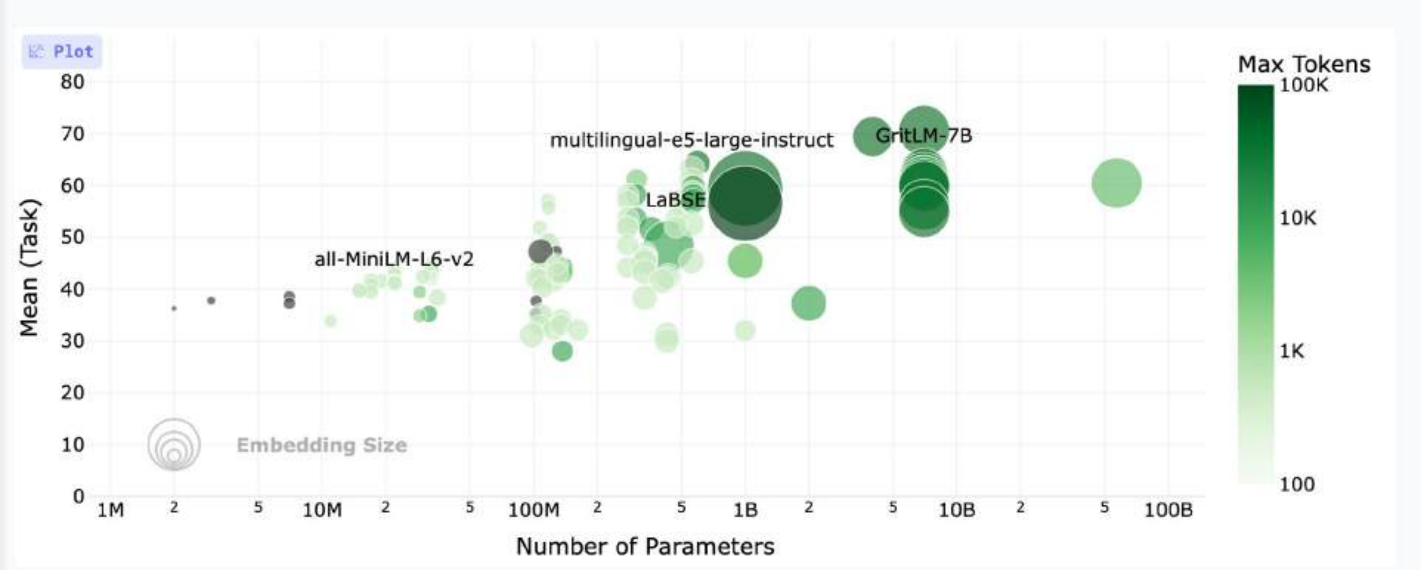
- 300 Models
- 100+ Tasks
- 1000+ Languages




@rajistics

https://huggingface.co/spaces/mteb/leaderboard
https://huggingface.co/blog/rteb
Hands On BEIR: https://colab.research.google.com/drive/1HfutiEhHMJLXiWGT8pcipxT5L2TpYEdt?authuser=1

# Selecting a embedding model




- Accuracy
- Latency
- Compute (CPU/GPU)

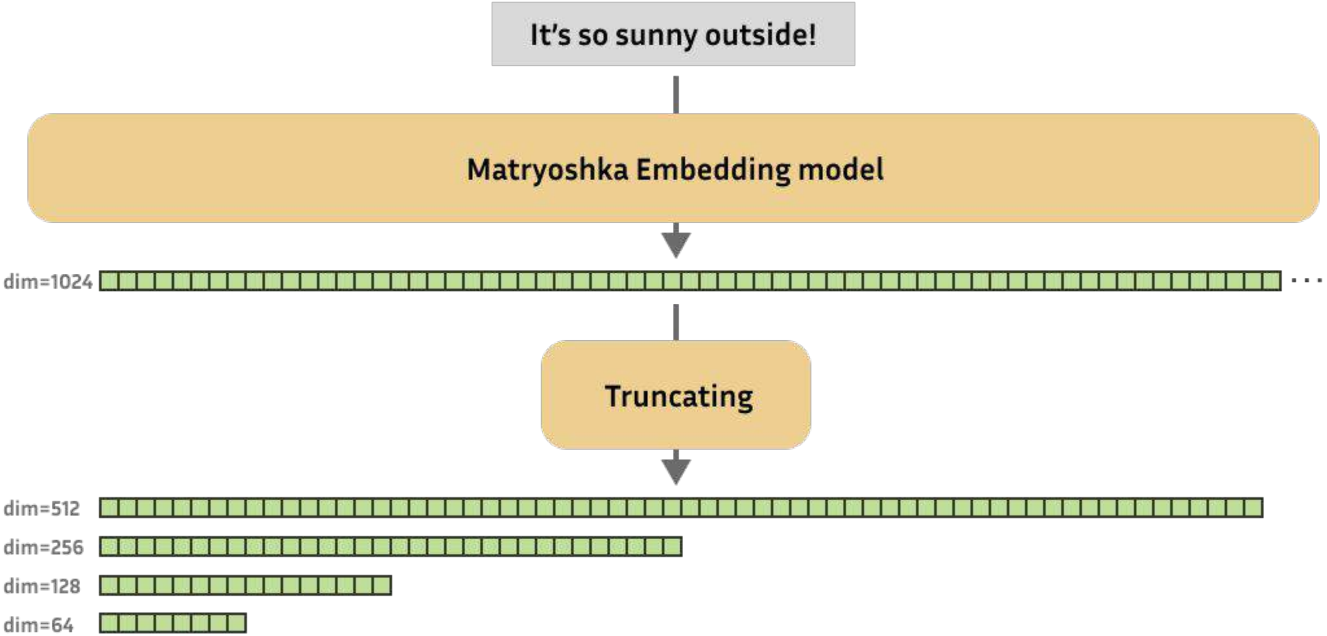
# Selecting a embedding model

Other considerations:

- Model Size
- Architecture (CPU/GPU/Quantization)
- Embedding Dimension (128 to 8960)
- Training Data (Multilingual, Domain)
  - Fine Tuning

https://www.linkedin.com/posts/rahul-jha-bbab34135_sklearn-vs-faiss-on-10-million-vectors-i-activity-7366464933313597441-iq-o?utm_source=share&utm_medium=member_desktop&rcm=ACoAAADryTMBzpgBkxPBepvoHPa6tLooVFSd8sQ

# Matryoshka Embedding Models

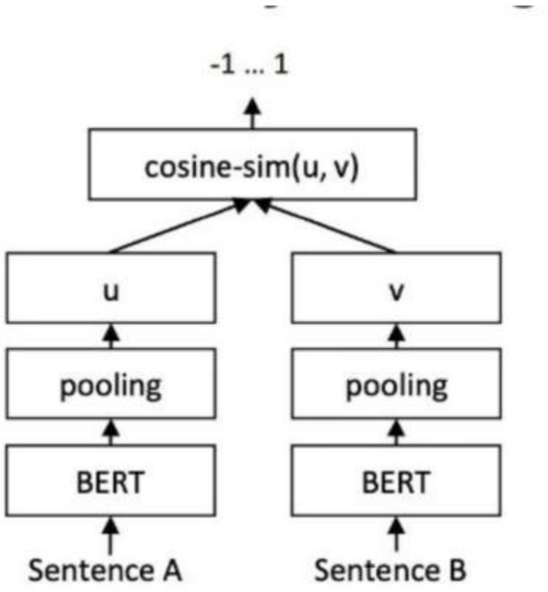



https://huggingface.co/blog/matryoshka

@rajistics

# Sentence Transformer

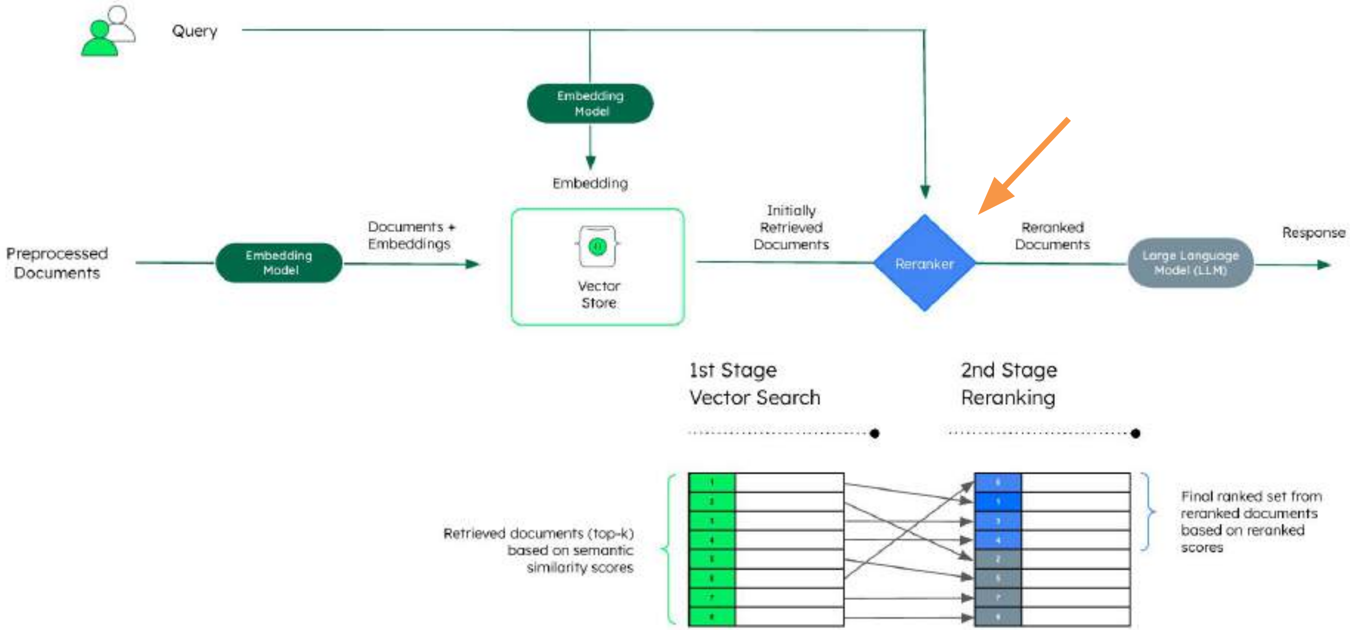
- Designed for Sentence-Level Meaning
- Semantic Search Ready
- Better Performance on Retrieval
- Efficiency




Elmo -> BERT -> DistilBERT
Sentence Transformers

https://sbert.net/

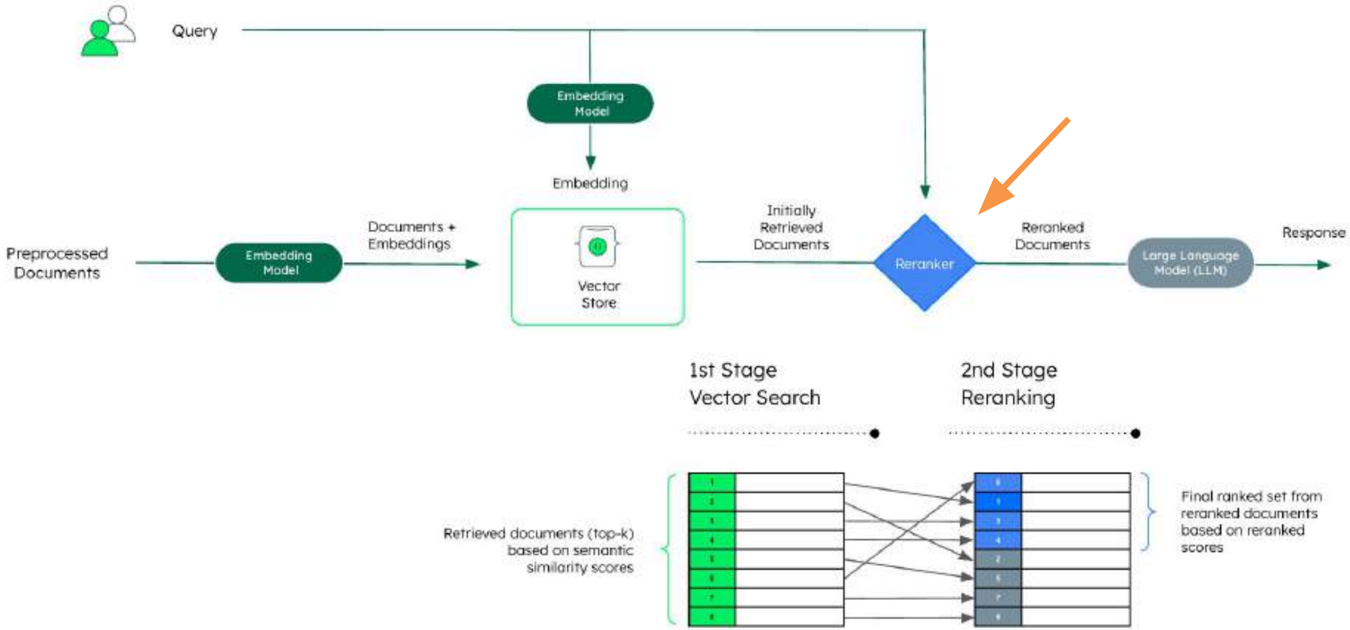
# Cross Encoder / Reranker




https://www.mongodb.com/resources/basics/artificial-intelligence/reranking-models

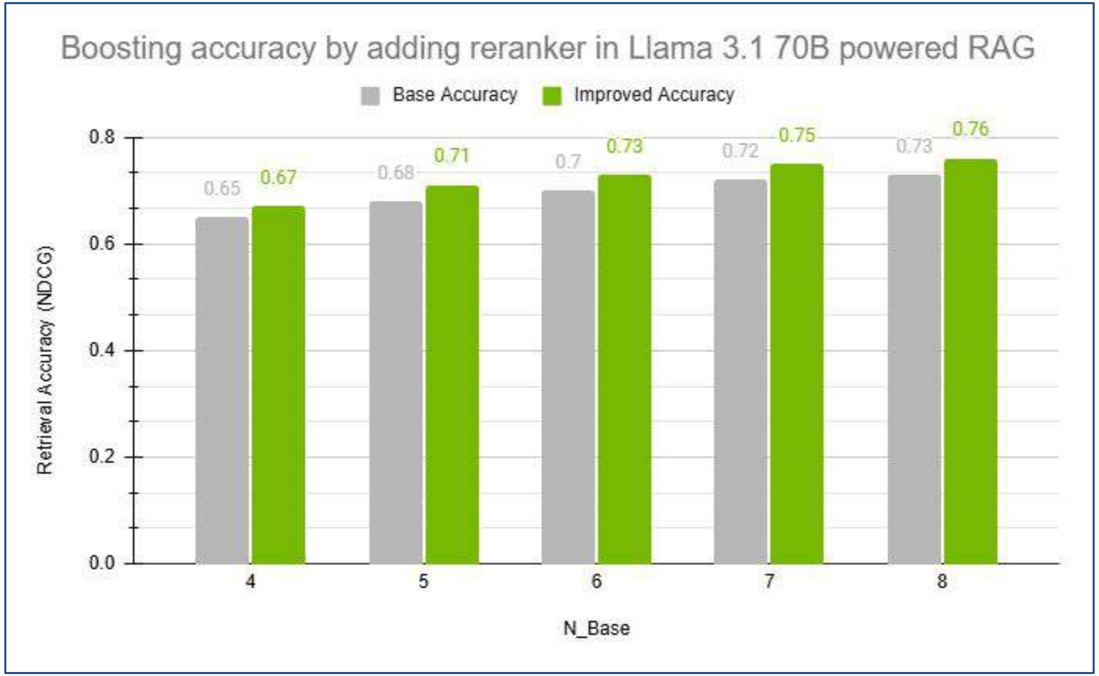
@rajistics

# Cross Encoder / Reranker



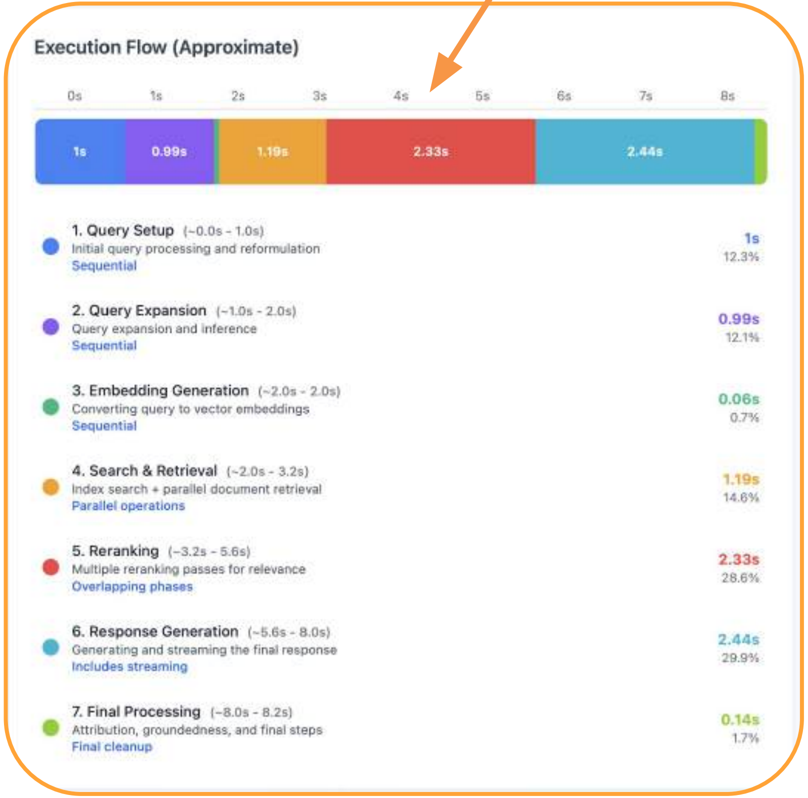

https://www.mongodb.com/resources/basics/artificial-intelligence/reranking-models

# Cross Encoder / Reranker




https://developer.nvidia.com/blog/how-using-a-reranking-microservice-can-improve-accuracy-and-costs-of-information-retrieval/

# Cross Encoder / Reranker



# Hands On: Retriever & Reranker

## Retrieve & Re-Rank Demo over Simple Wikipedia

This examples demonstrates the Retrieve & Re-Rank Setup and allows to search over Simple Wikipedia.

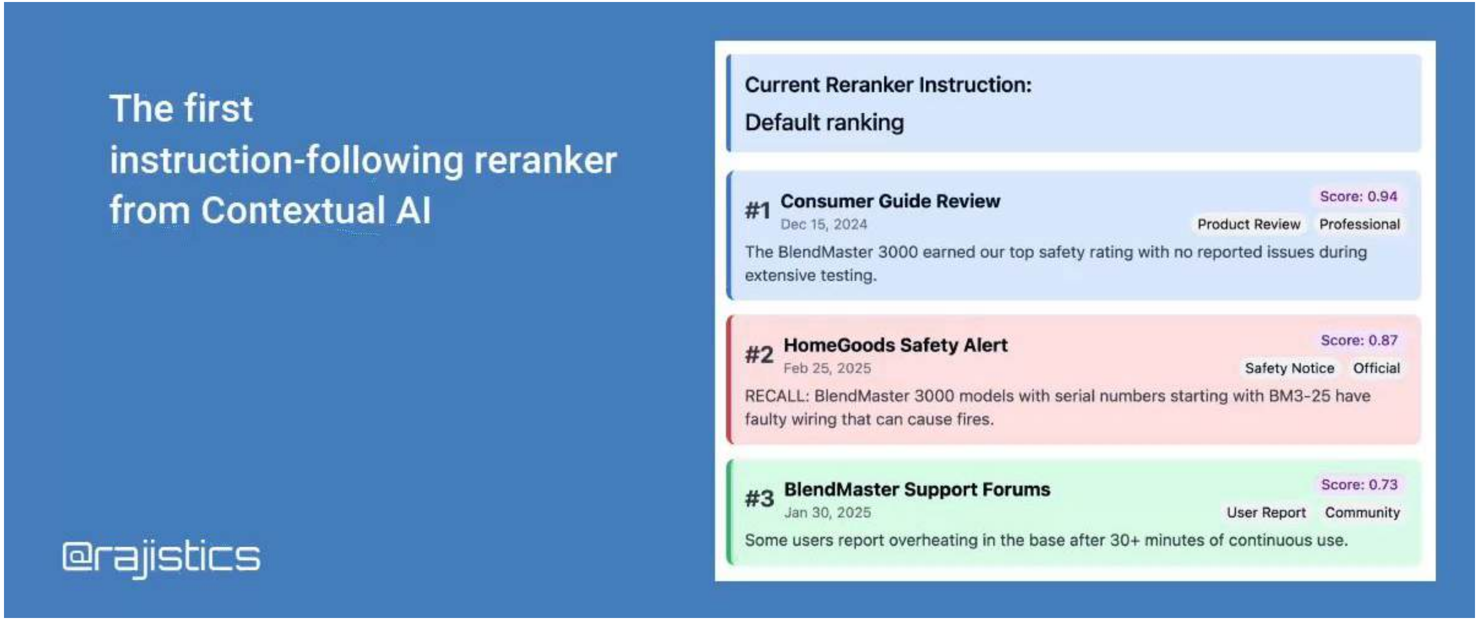
You can input a query or a question. The script then uses semantic search to find relevant passages in Simple English W
smaller and fits better in RAM).

For semantic search, we use `SentenceTransformer('multi-qa-MiniLM-L6-cos-v1')` and retrieve 32 potentia
answer the input query.

Next, we use a more powerful CrossEncoder (`cross_encoder = CrossEncoder('cross-encoder/ms-marco-M`
that scores the query and all retrieved passages for their relevancy. The cross-encoder further boost the performance, e
search over a corpus for which the bi-encoder was not trained for.

https://colab.research.google.com/drive/1lRr0J5fumRBP-RmTm5kD9lMd9nuOlhmI?authuser=1#scrollTo=HETfTO4P-otj

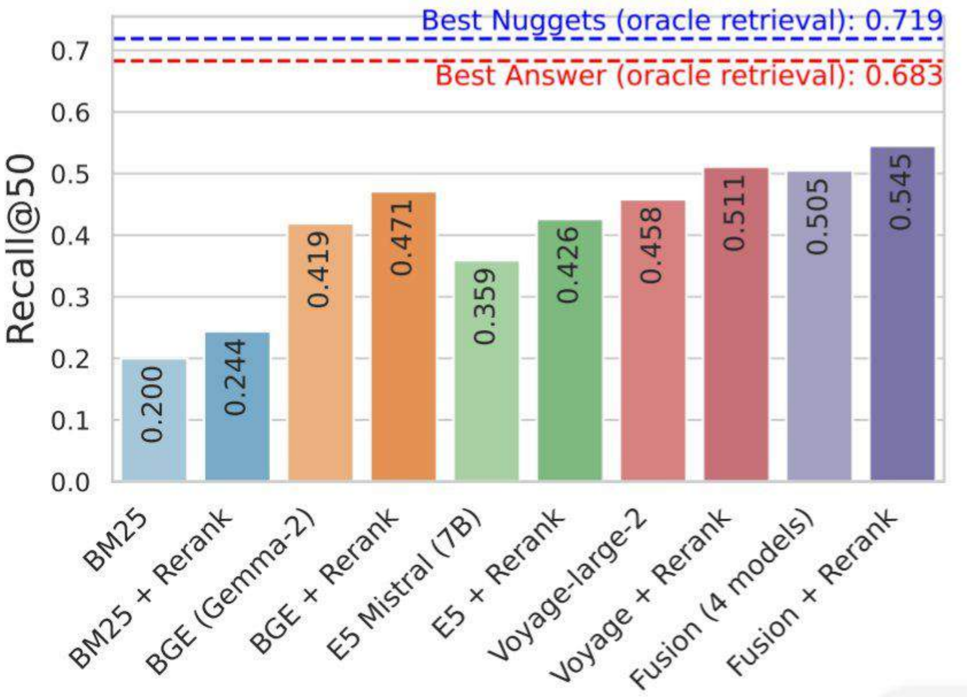
# Instruction Following Reranker



# Combine Multiple Retrievers

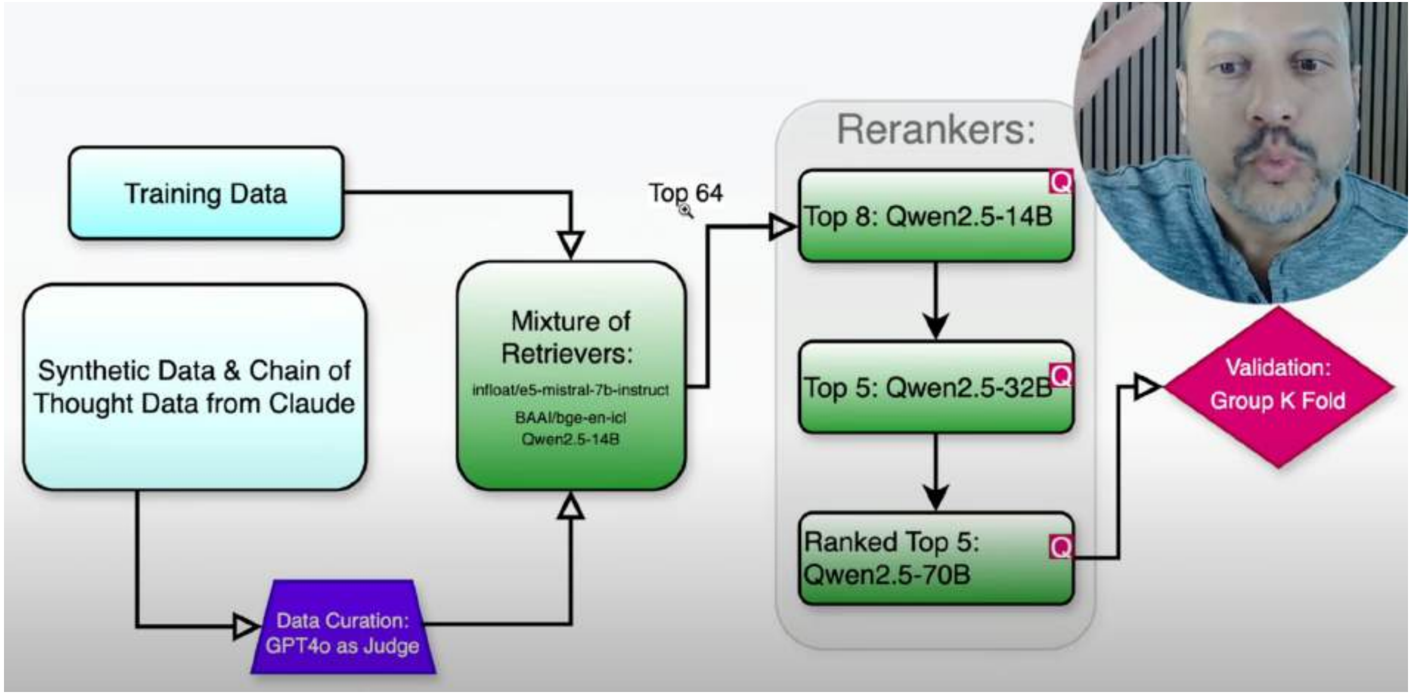
Technical Documents:

- BM25
- BGE (Gemma-2)
- E5 Mistral (7B)
- Voyager-large-2



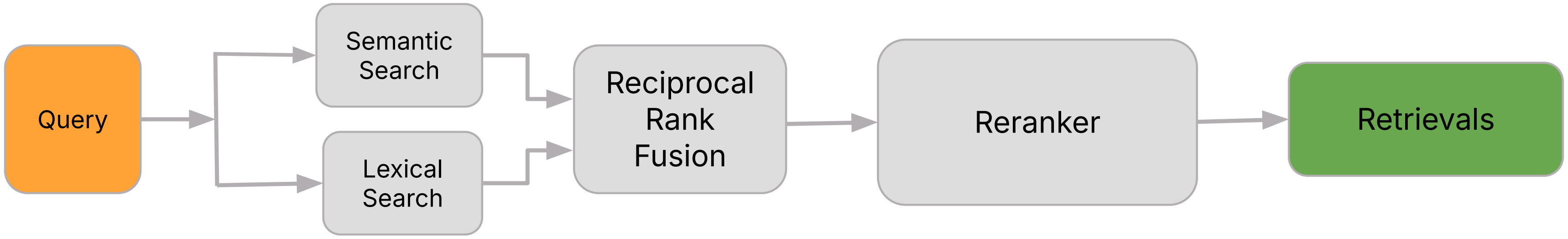

FreshStack: https://arxiv.org/pdf/2504.13128

# Cascading Rerankers in Kaggle




https://www.youtube.com/watch?v=Bnn2m4S22T4

# Best practices




@rajistics

# Families of Embedding Models

A quick taxonomy to orient choices (speed, accuracy, and how they're used).

Faster — More accurate (at match quality)

Model2Vec (Static) · DistilBERT (First-gen) · Sentence-Tfrs (Bi-encoder) · OpenAI Embeddings · BGE / Instructor (LLM-based) · Cross-Encoders (Rerankers)

### Model2Vec (Static Embeddings)

- Token-level vectors; fixed vocabulary
- Very fast; tiny footprint
- Weak on synonyms/negation
- Okay for simple similarity

### DistilBERT (First generation)

- CLS/mean pooling for sentences
- General-purpose; light footprint
- Better semantics than static vectors
- Baseline for small/medium use

### Sentence Transformers (Sentence-based)

- Siamese training for pair similarity
- Great recall for search & clustering
- Index-friendly; fast at query time
- Solid default for retrieval

### Cross-Encoders — Reranker

- Token-level interaction per pair
- Highest accuracy for top-k rerank
- Slow O(k) — not full-corpus search
- Use after retrieval for precision

### OpenAI Embeddings — Managed API

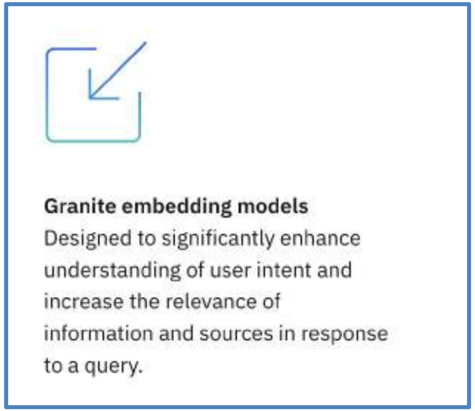
- Hosted, versioned; easy scaling
- Multilingual; frequent updates
- Pay-per-use; vendor lock-in risk
- Great when not self-hosting

### BGE / Instructor (LLM-based)

- Large LLM encoders; instruction aware
- Strong R@k across domains
- Heavier to serve; quantize if needed
- Pairs well with a reranker

# Lots of New Models


John Hopkins University




IBM


Google

https://arxiv.org/pdf/2509.06888
https://developers.googleblog.com/en/introducing-embeddinggemma/
https://huggingface.co/collections/ibm-granite/granite-embedding-models-6750b30c802c1926a35550bb

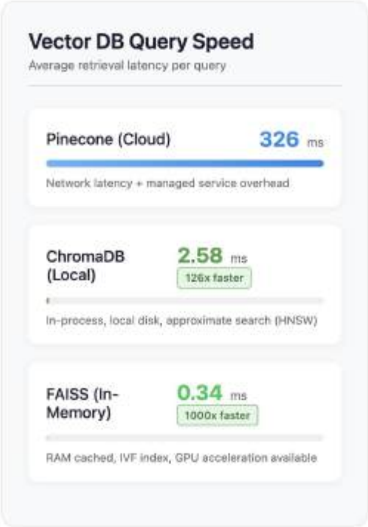
# Other retrieval methods:

- SPLADE for sparse
- ColBERT Late Interaction
- GraphRAG
- Many RAG flavors

# Operational Concerns:

## Computing Embeddings


## Storing Embeddings




@rajistics

https://www.linkedin.com/posts/rahul-jha-bbab34135_sklearn-vs-faiss-on-10-million-vectors-i-activity-7366464933313597441-iq-o
https://pub.towardsai.net/vector-databases-performance-comparison-chromadb-vs-pinecone-vs-faiss-real-benchmarks-that-will-3eb83027c584

# Vector Database Options

## 🗄️ Vector Database Layered Storage Architecture

Storage tier optimization and technical solution configuration based on latency requirements

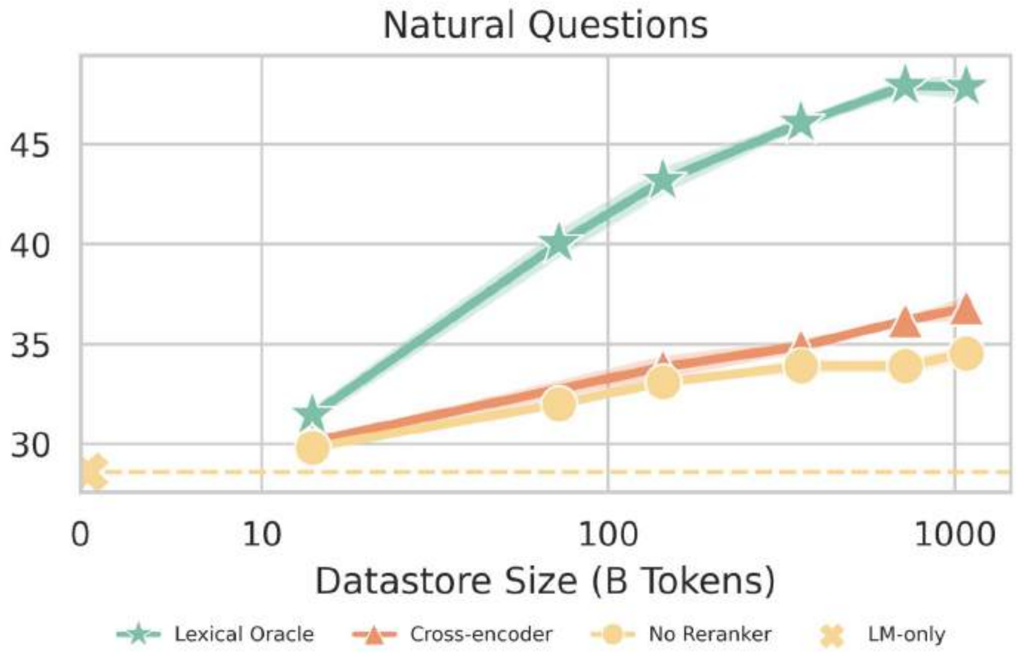
| Storage Tier | Latency Requirements | Core Application Scenarios | Technical Solutions |
|---|---|---|---|
| Hot Data Layer | < 50ms | 🔍 Real-time search<br>🎯 Intelligent recommendations<br>📢 Targeted advertising | Traditional specialized vector databases<br>(e.g., Milvus hot instances) |
| Warm Data Layer | 50-500ms | 💬 Standard RAG dialogue<br>👥 Multi-tenant shared services | S3Vector<br>Milvus three-tier storage instances |
| Cold Data Layer | > 500ms | 📦 Historical data archiving<br>📊 Offline data analysis | S3+Spark/Daft<br>Milvus vector data lake |

💡 **Note:** This architecture demonstrates the latency requirements, applicable scenarios, and corresponding technical solutions for different storage tiers in vector database layered storage. The hot→warm→cold tier design optimizes overall storage costs and performance.

https://zilliz.com/blog/will-amazon-s3-vectors-kill-vector-databases-or-save-them

# Operational Concerns:

As datastores get bigger, you need to work on improving retrieval performance



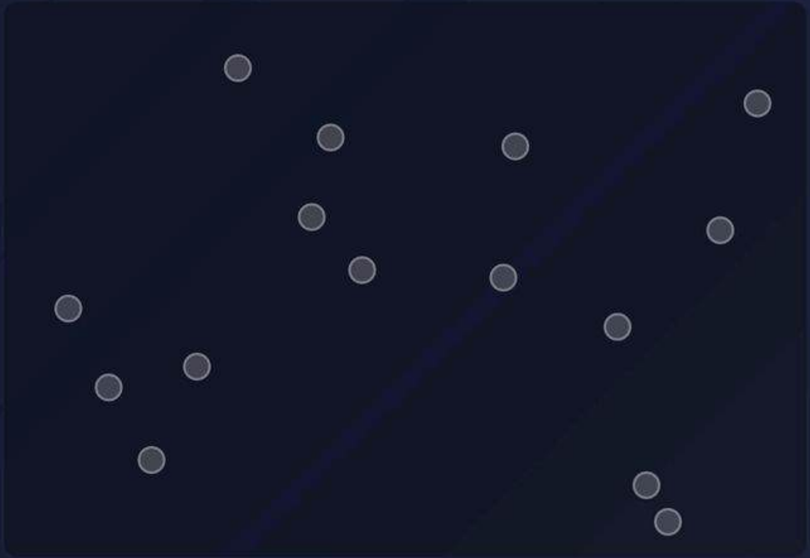

Trillion Token: https://arxiv.org/pdf/2407.12854

@rajistics

# Search Strategy Comparison

Watch how different approaches explore the solution space

## Traditional RAG

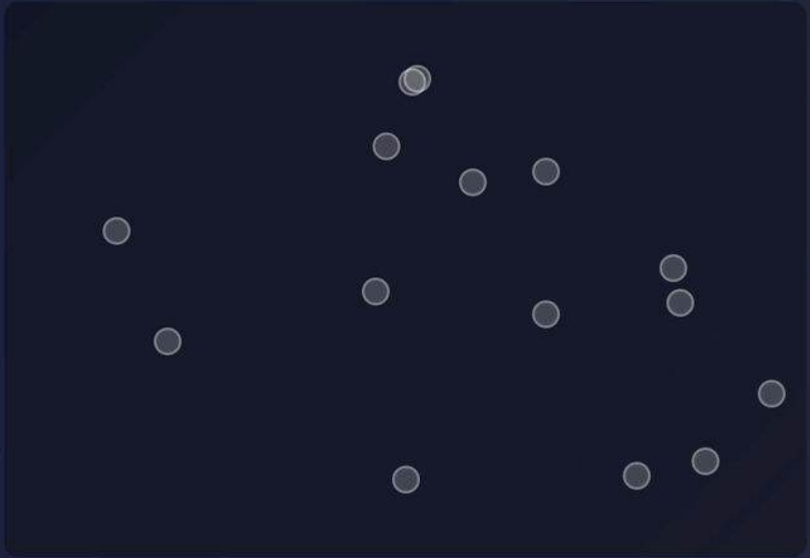


Ready to search...

**0**
Queries Made

**0ms**
Time Taken

## Agentic RAG



Ready to explore...

**0**
Queries Made

**0ms**
Time Taken

# Tools use / Reasoning

Use reasoning models to keep using queries until satisfied

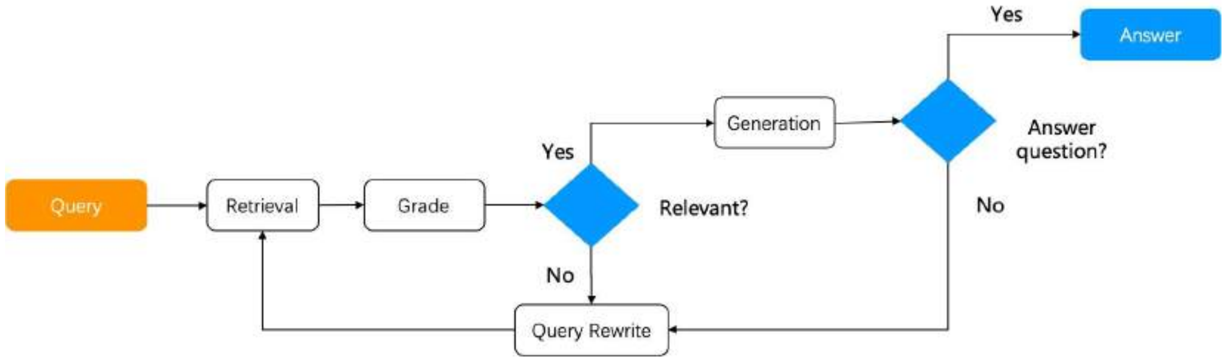
### Scientific research

scientific_research.py

```
from agno.agent import Agent
from agno.models.openai import OpenAIChat

task = (
    "Read the following abstract of a scientific paper and provide a critical evalu
    "results, conclusions, and any potential biases or flaws:\n\n"
    "Abstract: This study examines the effect of a new teaching method on student p
    "A sample of 30 students was selected from a single school and taught using the
    "The results showed a 15% increase in test scores compared to the previous seme
    "The study concludes that the new teaching method is effective in improving mat
)
reasoning_agent = Agent(
    model=OpenAIChat(id="gpt-5-mini-2024-08-06"), reasoning=True, markdown=True
)
reasoning_agent.print_response(task, stream=True, show_full_reasoning=True)
```

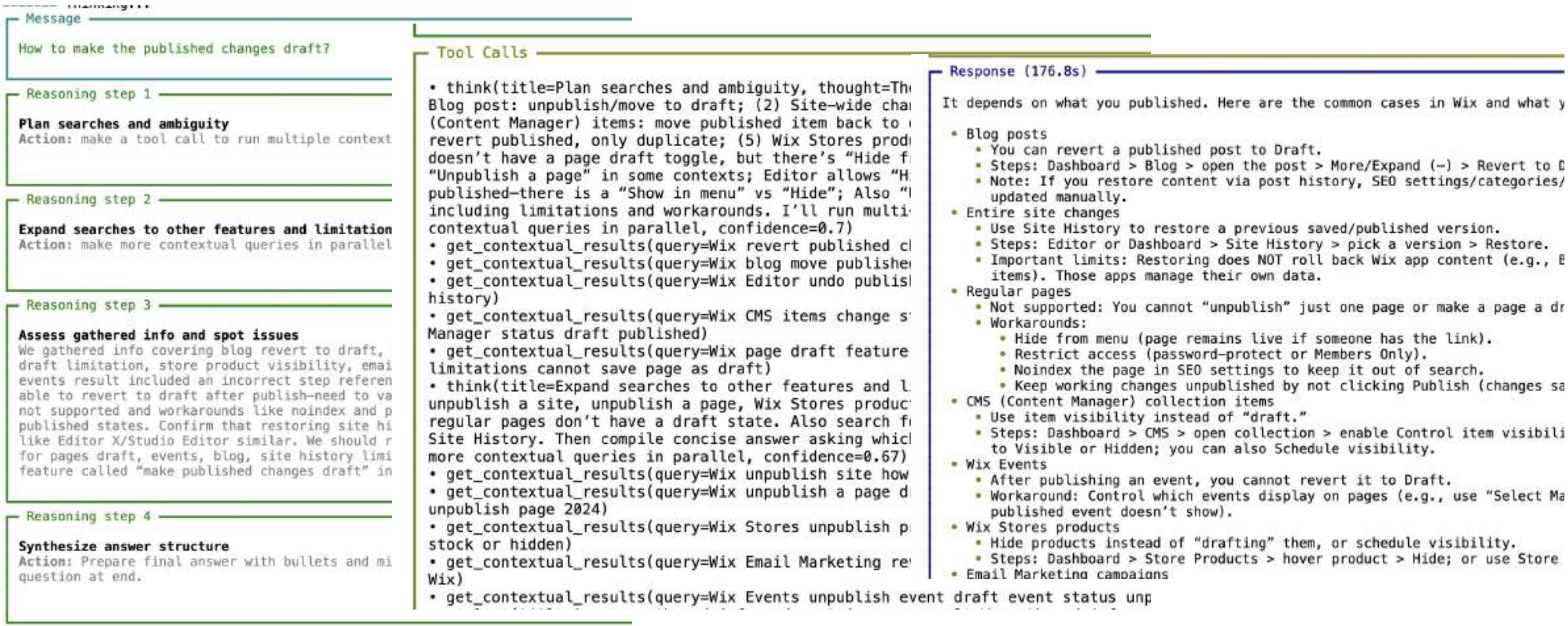
https://docs.agno.com/concepts/reasoning/reasoning-agents

# Agentic RAG



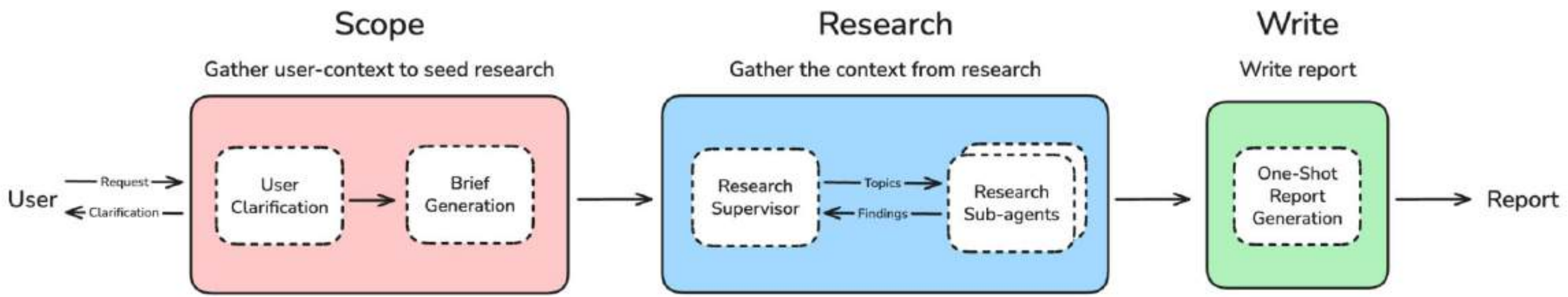

https://ragflow.io/blog/agentic-rag-definition-and-low-code-implementation

# Tools use / Reasoning




@rajistics

# 🔬 Open Deep Research




https://github.com/langchain-ai/open_deep_research

# DeepResearch Bench

100 PhD-level research tasks

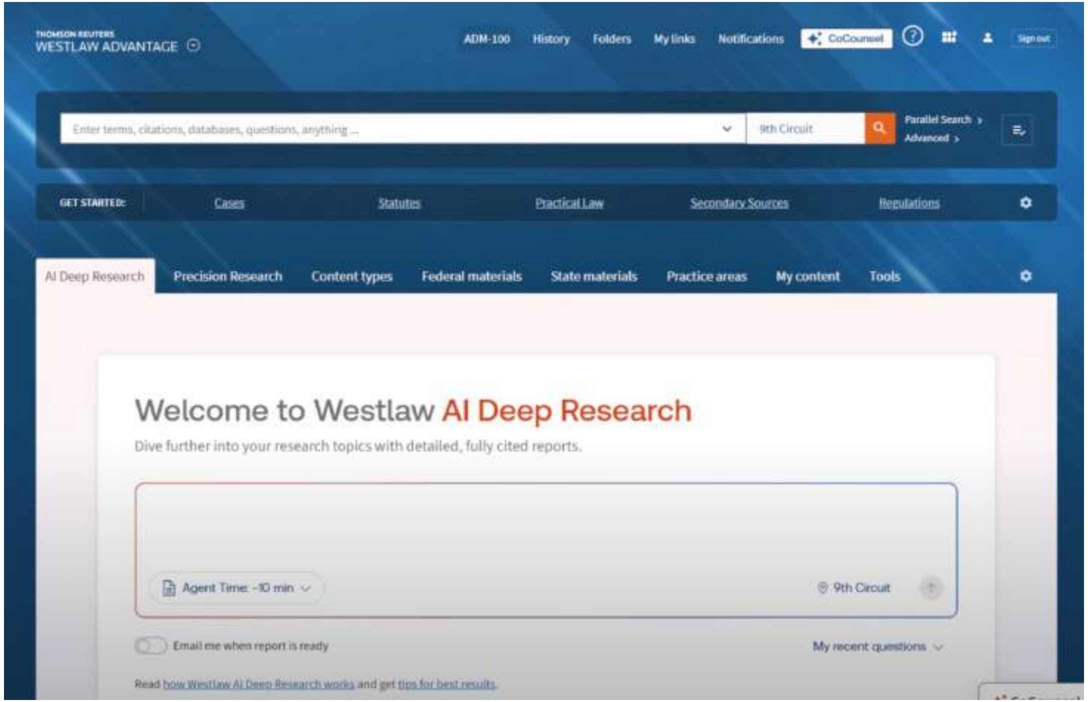
| Rank | model | overall | comp. | insig |
|---|---|---|---|---|
| 1 | 🚀 gemini-2.5-pro-deepresearch | 49.71 | 49.51 | 49.45 |
| 2 | 🚀 openai-deepresearch | 46.45 | 46.46 | 43.73 |
| 3 | 🚀 claude-research | 45 | 45.34 | 42.79 |
| 4 | 🚀 kimi-researcher | 44.64 | 44.96 | 41.97 |
| 5 | 🚀 doubao-deepresearch | 44.34 | 44.84 | 40.56 |
| 6 | 🚀 langchain-open-deep-research | 43.44 | 42.97 | 39.17 |
| 7 | nvidia-aiq-research-assistant | 40.52 | 37.98 | 38.39 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Deep Research Bench Submission | c0a160b | openai:gpt-4.1-nano | openai:gpt-4.1 | openai:gpt-4.1 | $87.83 | 207,005,549 |

https://huggingface.co/spaces/Ayanami0730/DeepResearch-Leaderboard

@rajistics

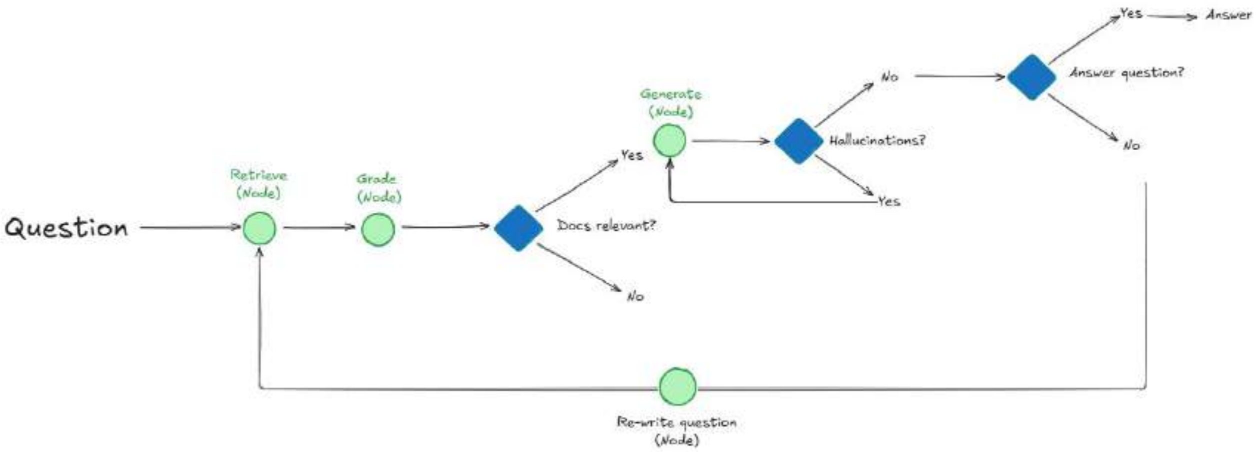
# Westlaw AI Deep Research




https://www.youtube.com/watch?v=tvpH36uT7hw

@rajistics

# Agentic RAG



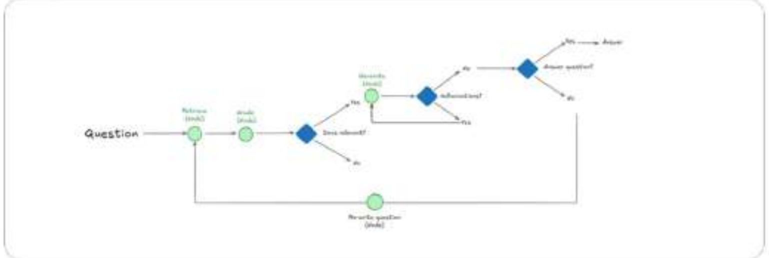

Self RAG: https://arxiv.org/pdf/2310.11511

# Agentic RAG

r/LangChain • 2 days ago
Best-Information2493

## I Taught My Retrieval-Augmented Generation System to Think 'Do I Actually Need This?' Before Retrieving

Tutorial




Traditional RAG retrieves blindly and hopes for the best. Self-Reflection RAG actually evaluates if its retrieved docs are useful and grades its own responses.

### What makes it special:

- Self-grading on retrieved documents Adaptive retrieval
- decides when to retrieve vs. use internal knowledge
- Quality control reflects on its own generations
- Practical implementation with Langchain + GROQ LLM

https://www.reddit.com/r/LangChain/comments/1njmb1r/i_taught_my_retrievalaugmented_generation_system/

# Agentic RAG

**Best-Information2493** OP · 1d ago

You're absolutely right about the inefficiency!

**Non-relevant docs:** The system usually tries to rewrite and retrieve again, but it should have better fallbacks like using pure LLM knowledge or graceful exit.

**Resource waste:** Going through the full pipeline just to restart is brutal. Better approaches would be:

- Early stopping at each step,

- Circuit breakers to prevent endless loops,

- Caching intermediate results

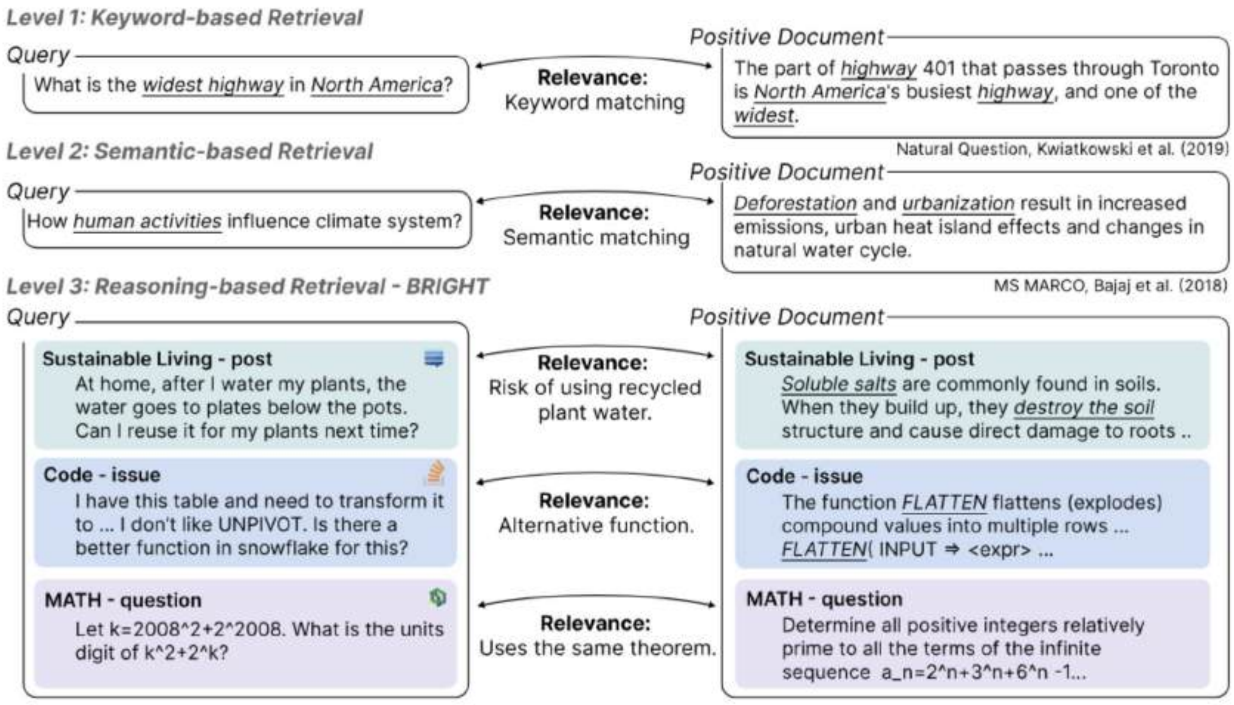
The paper prioritizes accuracy over efficiency real production systems definitely need smarter resource management.

2 Reply Award Share ···

https://www.reddit.com/r/LangChain/comments/1njmb1r/i_taught_my_retrievalaugmented_generation_system/

# Research: BRIGHT

Analyzing retrieval reasoning

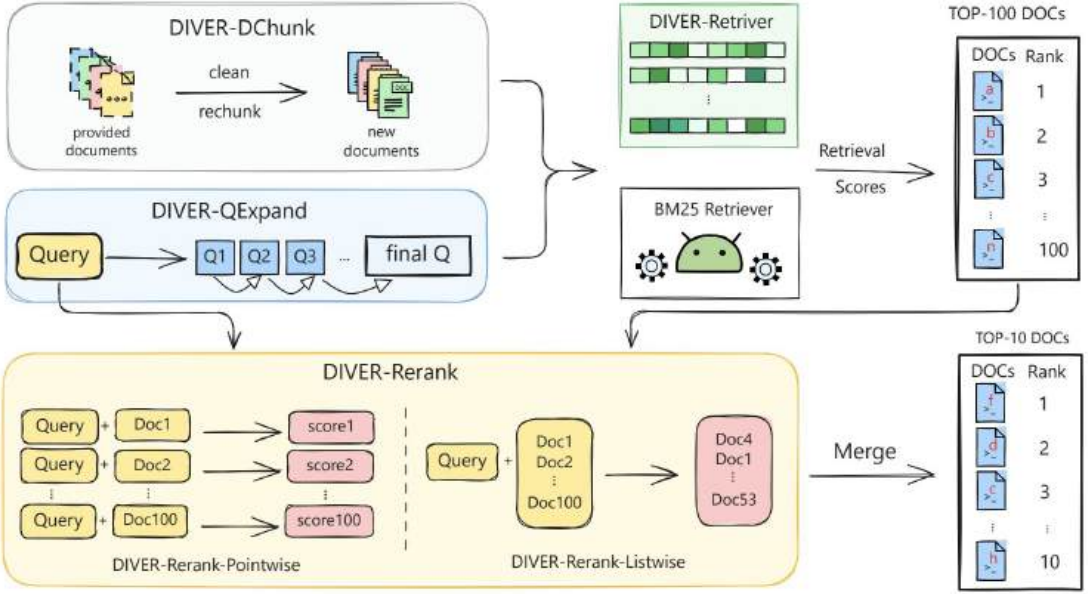



BRIGHT: https://arxiv.org/pdf/2407.12883

# BRIGHT #1: DIVER

Reasoning-intensive

Information Retrieval




https://arxiv.org/pdf/2508.07995

@rajistics

# BRIGHT #1: DIVER

Reasoning-intensive

Information Retrieval

Table 1: Prompts used in DIVER-QExpand for query expansion. Braces {} denote placeholders.

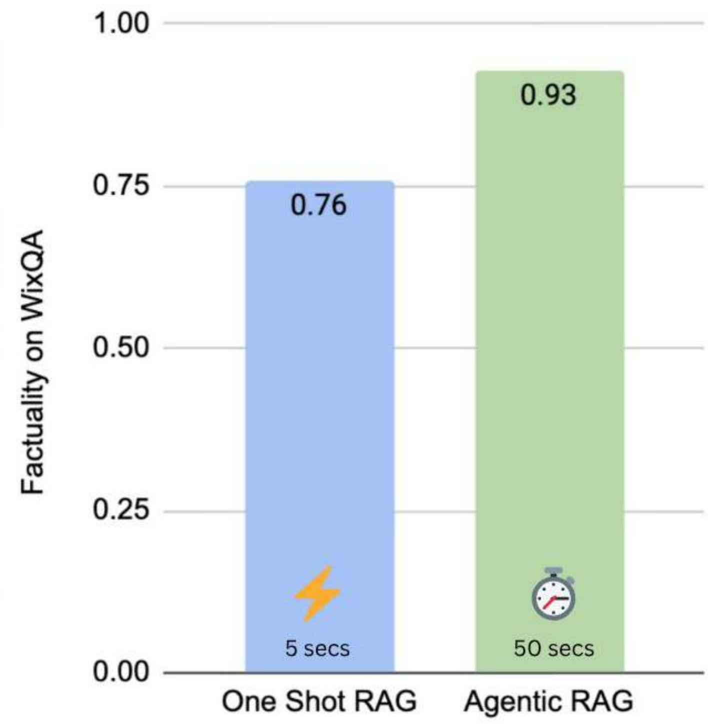
| Prompt Stage | LLM Instruction |
|---|---|
| **First Round** | Given a query and the provided passages (most of which may be incorrect or irrelevant), identify helpful information from the passages and use it to write a correct answering passage. Use your own knowledge, not just the example passages!<br>Query: {`query`}<br>Possible helpful passages: {`top-k retrieved documents`} |
| **Subsequent Rounds** | Given a query, the provided passages (most of which may be incorrect or irrelevant), and the previous round's answer, identify helpful information from the passages and refine the prior answer. Ensure the output directly addresses the original query. Use your own knowledge, not just the example passages!<br>Query: {`query`}<br>Possible helpful passages: {`top-k retrieved documents`}<br>Prior generated answer: {`last-round expansion`} |

https://arxiv.org/pdf/2508.07995

# Agentic RAG on WixQA

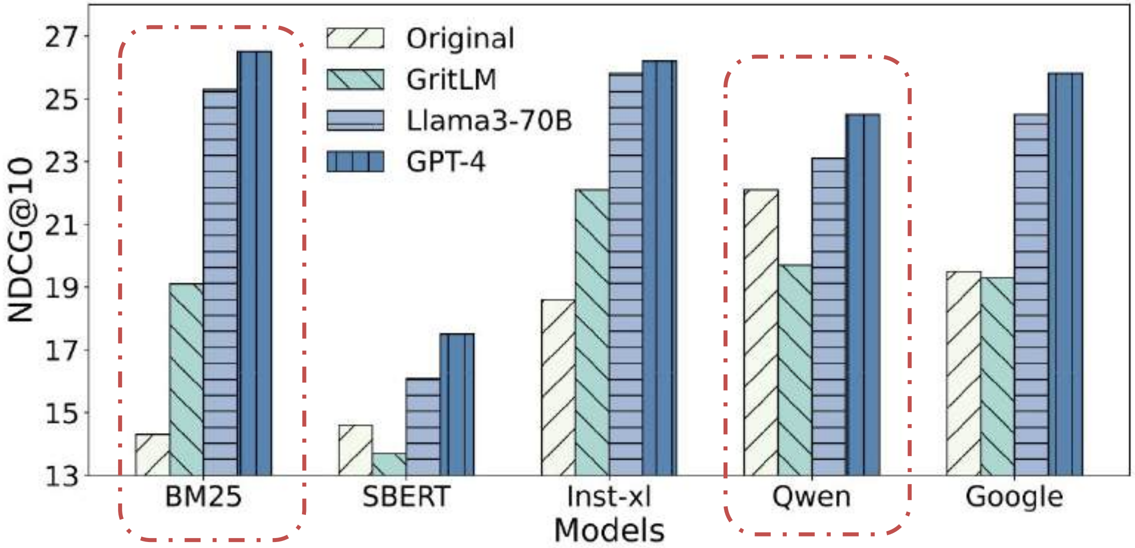
Pick:
- Accuracy
- Latency

(6s versus 50s)



# Rethink your Assumptions

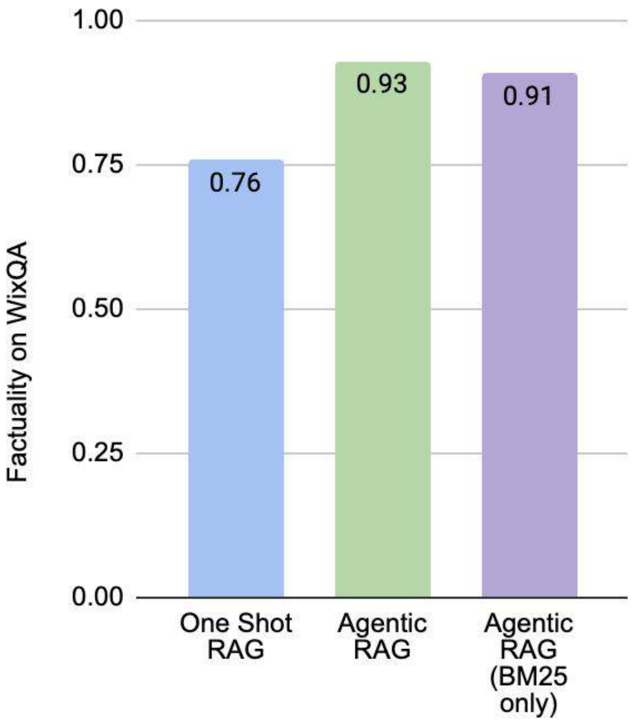
Querying with LLM using BM25

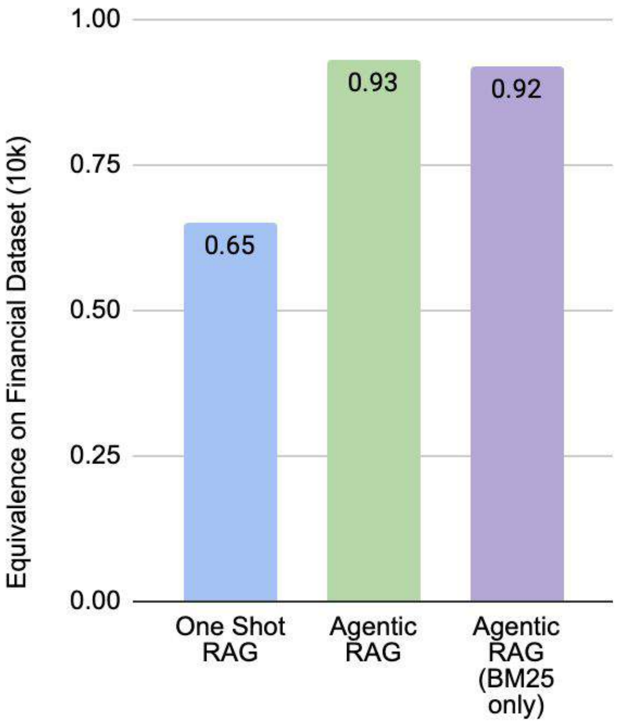



BRIGHT: https://arxiv.org/pdf/2407.12883

@rajistics

# Agentic RAG with BM25

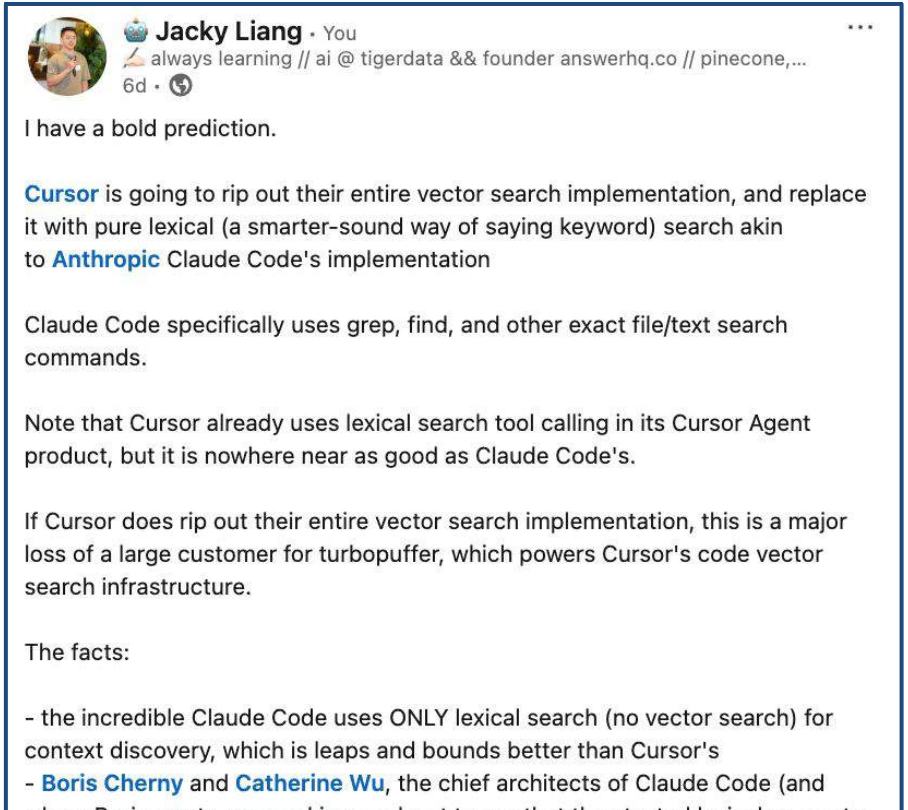



@rajistics

# Agentic RAG for Code Search

- **Claude Code (Lexical / Iterative Search)**
- Keeps searching (like grep) until it finds or rules out a function/dependency



Jacky Liang · You
always learning // ai @ tigerdata && founder answerhq.co // pinecone,...
6d ·

I have a bold prediction.

Cursor is going to rip out their entire vector search implementation, and replace it with pure lexical (a smarter-sound way of saying keyword) search akin to Anthropic Claude Code's implementation

Claude Code specifically uses grep, find, and other exact file/text search commands.

Note that Cursor already uses lexical search tool calling in its Cursor Agent product, but it is nowhere near as good as Claude Code's.

If Cursor does rip out their entire vector search implementation, this is a major loss of a large customer for turbopuffer, which powers Cursor's code vector search infrastructure.

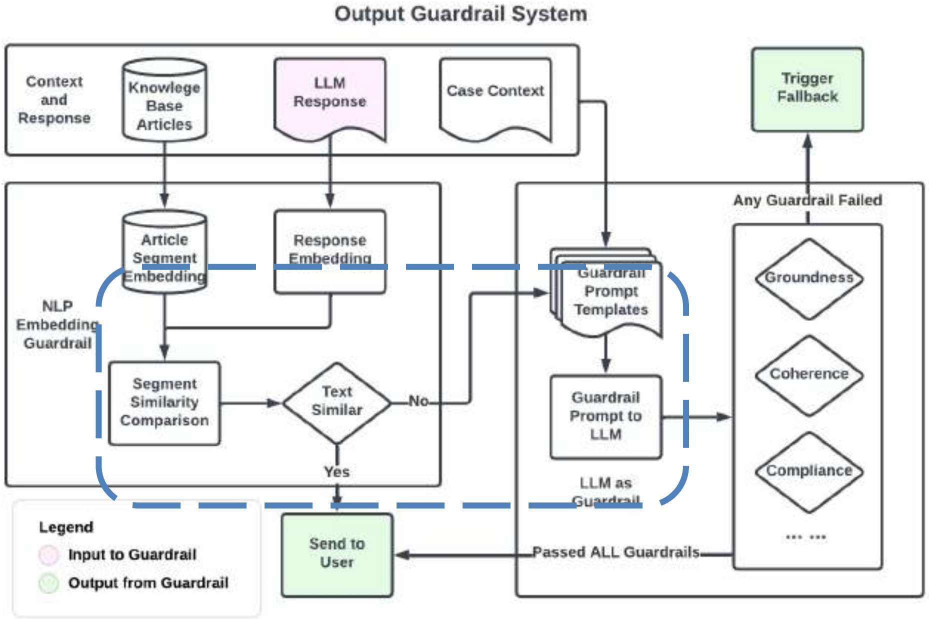
The facts:

- the incredible Claude Code uses ONLY lexical search (no vector search) for context discovery, which is leaps and bounds better than Cursor's
- Boris Cherny and Catherine Wu, the chief architects of Claude Code (and

https://x.com/pashmerepat/status/1926717705660375463
https://www.tigerdata.com/blog/why-cursor-is-about-to-ditch-vector-search-and-you-should-too#reading-the-cursor-tea-leaves

# Combine Retrieval Approaches

Response Guardrail:
2 Tier System of Text + LLM




https://careersatdoordash.com/blog/large-language-modules-based-dasher-support-automation/
UAR: https://arxiv.org/html/2406.12534v1

@rajistics

# Hands on: Agentic RAG

## Agentic RAG with Hugging Face smolagents vs Vanilla RAG

Author: @MariaKhalusova

Last updated: Jan 9th, 2025

### What you'll learn:

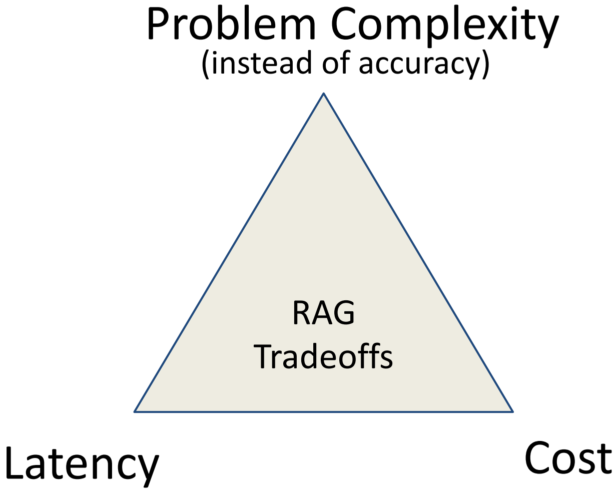
1. Parsing PDF documents from S3 into DataStax AstraDB with Unstructured Platform
2. Building Vanilla RAG in pure Python without using specialized frameworks
3. Differences between Vanilla RAG and Agentic RAG
4. Creating Agentic RAG with Hugging Face `smolagents` library
5. Whether Agentic RAG can produce better answers (spoiler: it can!)

In Vanilla RAG, your system uses the user's question to perform a single retrieval step and get a batch of documents that are r
relevant to the query. These documents are then passed on to the LLM to generate an answer grounded in the context of thos

However, this approach has limitations. If the results of the retrieval are inadequate (either irrelevant or incomplete), this will h
negative impact on generation. There are many different methods one can employ to improve the retrieval quality, such as cho
embedding model, switching to a different retrieval method (e.g., BM25, or hybrid, metadata filtering, etc.), increasing the num
documents, and adding a reranker. However, there may still be situations where a single retrieval step, or retrieving based on t
"as is," may not produce optimal results.

Smolagents: https://colab.research.google.com/drive/1hG3dPgd8wjrO9wSD0K0Feo7EY1iXqrEN
Page Index: https://github.com/VectifyAI/PageIndex

# Solutions for a RAG Solution




- High cost of mistakes + budget → Rerankers
- Need <5s latency → BM25 + Static Embeddings
- Complex multi-hop queries → Agentic RAG

# Retriever Checklist

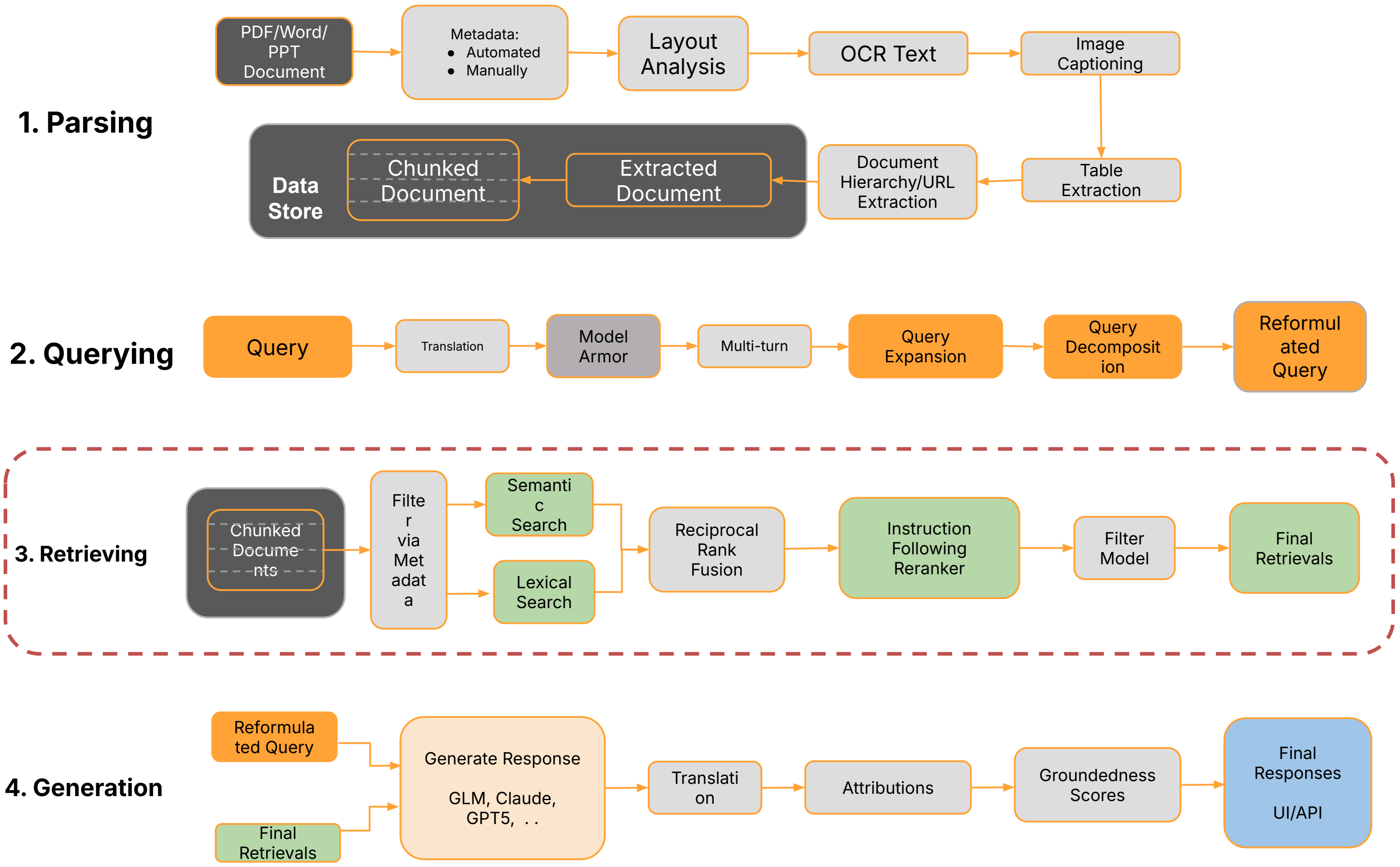
- Keyword / BM25
- Semantic Search / Embedding Model
- Agentic / Reasoning LLM

**BM25**
Keyword-based retrieval

**Language Models**
Semantic meaning with embeddings

**Agentic Search**
Dynamic
using LLM Reasoning

# Retrieval

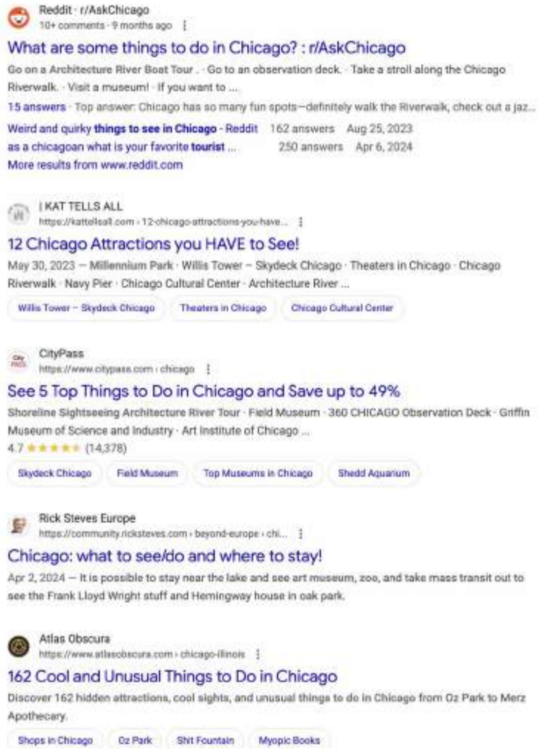



@rajistics

# RAG - Generation

Don't want a list of search results
So use a generation model

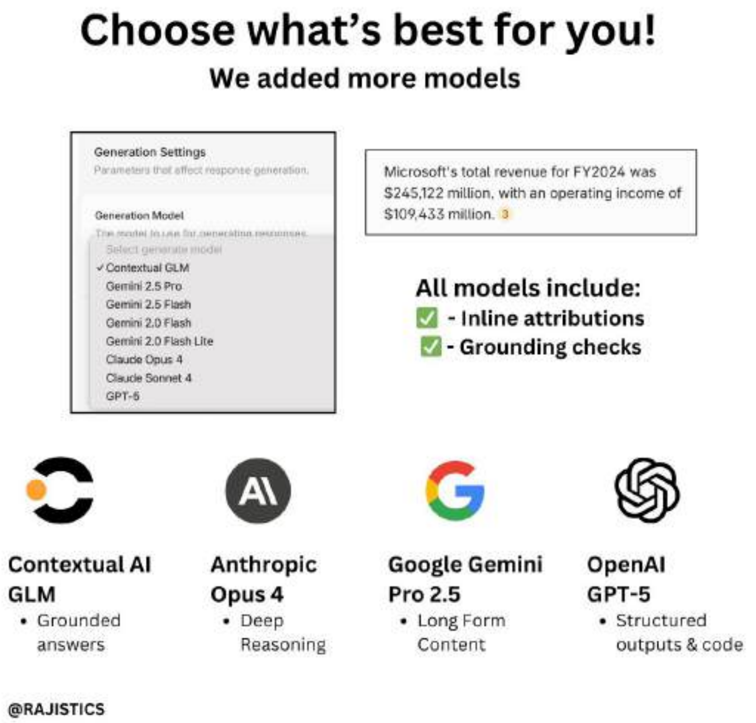
Greatest area of technical improvement for RAG
in the last few years



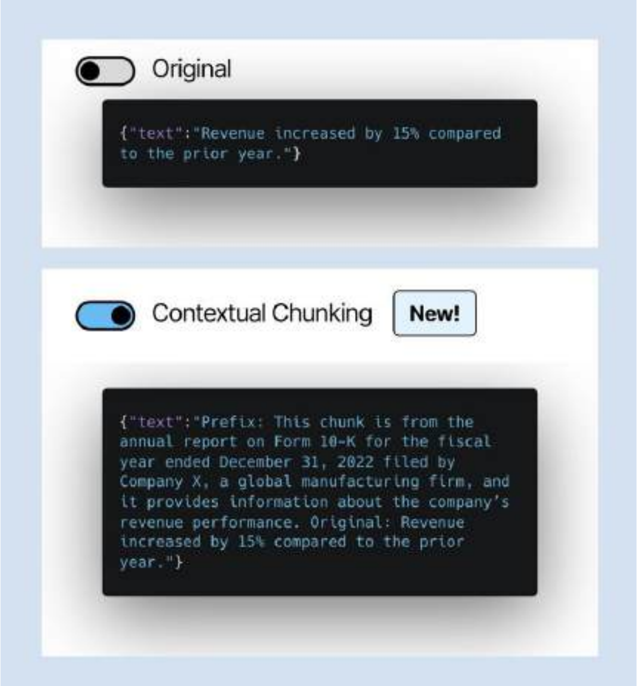

@rajistics

# RAG - Generation

Less interesting, because either choose
- Best generation model that fits your cost/latency budget
- Special needs
    - Low hallucination (Contextual GLM)
    - Domain Specific (Fine Tuned Healthcare LLM)
    - Language Specific
- Don't overindex on Context Window size -> Context Rot post



# Chunking approaches




https://www.linkedin.com/feed/update/urn:li:activity:7374098075033694208/

# From Vectors to Agents: Managing RAG in an Agentic World

Rajiv Shah
Chief Evangelist, Contextual AI
rajiv.shah@contextual.ai

https://github.com/rajshah4/LLM-Evaluation