Women of Search Presents: AI-Driven Information Retrieval

Audrey Lorberfeld

With contributions from Atita Arora, Erika Cardenas, and Meghan Boyd

24th April 2024

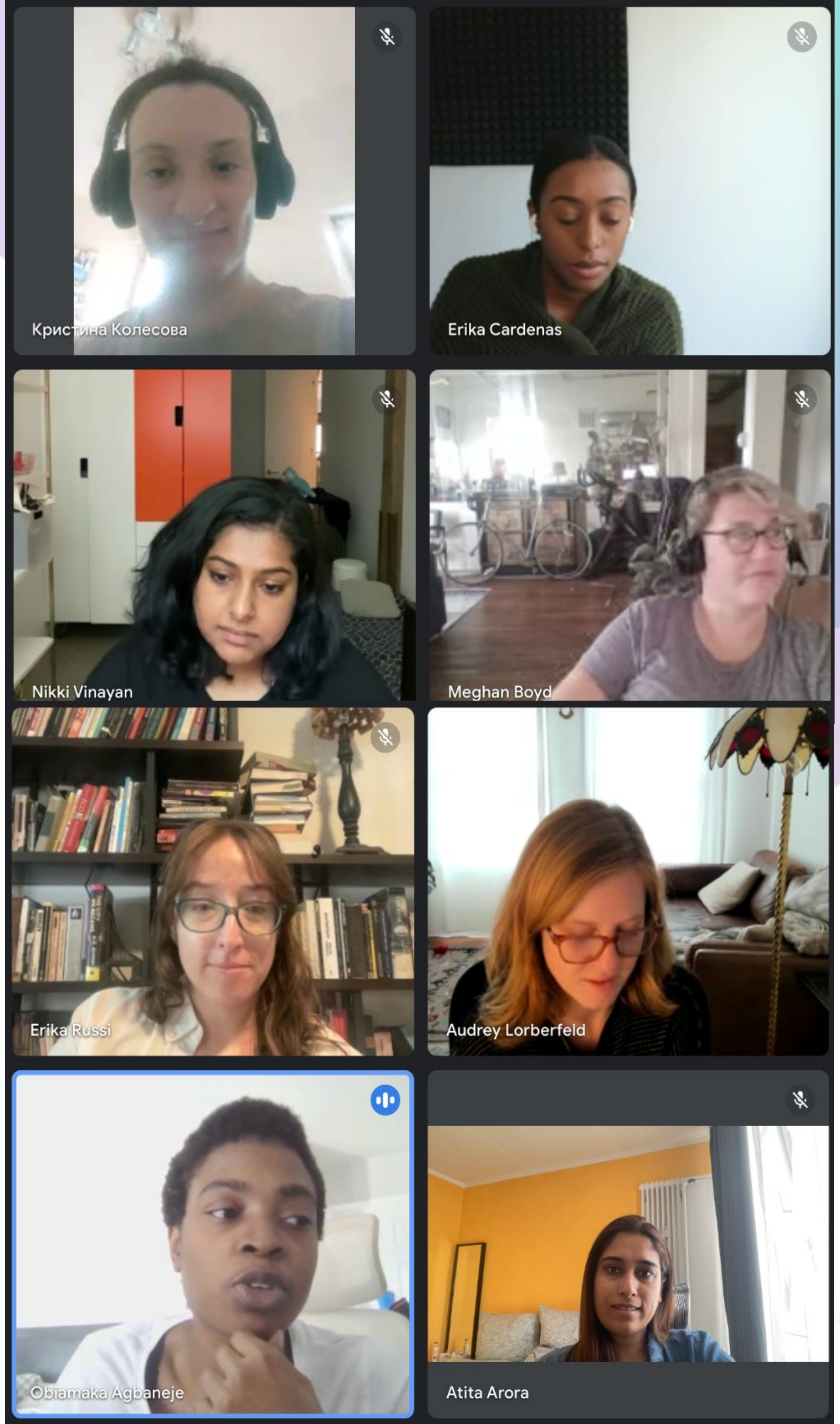
Women of Search

- Founded 3 years ago, in May 2021
- **226** members strong 💪
- A vibrant community dedicated to **empowering & celebrating women in search & related tech fields**
- Provide a platform for networking, mentorship, and knowledge-sharing

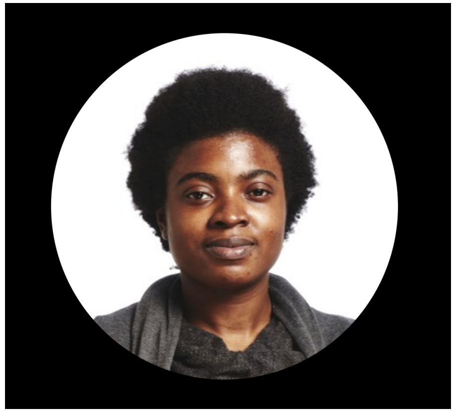
Group Updates

Updates

- **'Happy Hour'** continues: the 1st Wednesday of each month at 9 am PT.
- **Typescript Working Group**, founded and led by Moon Limb (S. Korea)
- **Research Paper Reading Club**, founded and led by Obiamaka Agbaneje (Canada)



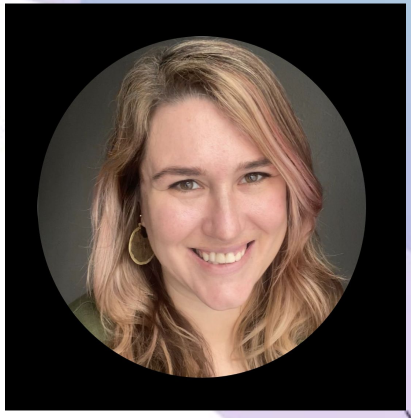
Member shout-outs



Obiamaka Agbaneje

Atita Arora

Erika Cardenas



Meghan Boyd

Moon Limb

Ashia Zawaduk

Elzbieta Jakubowska

Thank you, OSC

- 'Relevance & Matching Tech' Slack group:
  - Nearly 5k members
  - >129k messages sent in past 30 days
  - 88% of message-views in public channels in past 30 days
- Personally: **4 out of my 6 jobs** in tech have come directly from relationships I have made via this Slack community
- My first suggestion for people interested in getting involved with search is to join this Slack workspace

Let's get into it

Core Areas of Information Retrieval

- Relevance
- Retrieval Models
- Indexing
- Query Processing
- Ranking
- Evaluation
- User Interaction

Core Areas of Information Retrieval

- Relevance
- Retrieval Models
- Indexing
- Query Processing
- Ranking
- Evaluation
- User Interaction

→

AI has been used in these areas before...

- **2013:** Word2Vec released; FB incorporates graph search
- **2015:** Google incorporates RankBrain; MSFT Bing incorporates neural nets
- **2018:** BERT released
- **2022:** ChatGPT released

So what's special about *now*?

Now, the pace of innovation has **exploded**.

With the advent of LLMs, AI has become an integral part of life for *everyday* people, not just academics and industry experts.

Nearly everyone in the world has now experienced the value-add of these technologies first-hand.

Collective **expectation** when information-seeking is to receive **immediate**, **relevant**, **factual** information in response to complex, ambiguous, and natural language questions.

Search engines using AI now

- Andi
- Metaphor
- Brave
- YOU
- Phind
- Perplexity
- Kagi
- Komo
- ...Then there are search engine-adjacent tools like Gemini, Glean, Bing Copilot, Waldo, ChatGPT, etc.
- **More everyday!**

Generative AI is
cool, bro!!!!

Notoriously difficult IR techniques are now relatively easy

- Query intent identification
- Token/phrase disambiguation
- Query rewriting
- Multimodal search
- Metadata generation:
  - Document summarization
  - Keyword generation, etc.
- Advanced query stream analysis and segmentation

We are now living in an age in which **virtually no IR techniques are off the table**

Popular AI-Driven Information Retrieval Techniques

Popular AI-driven IR techniques

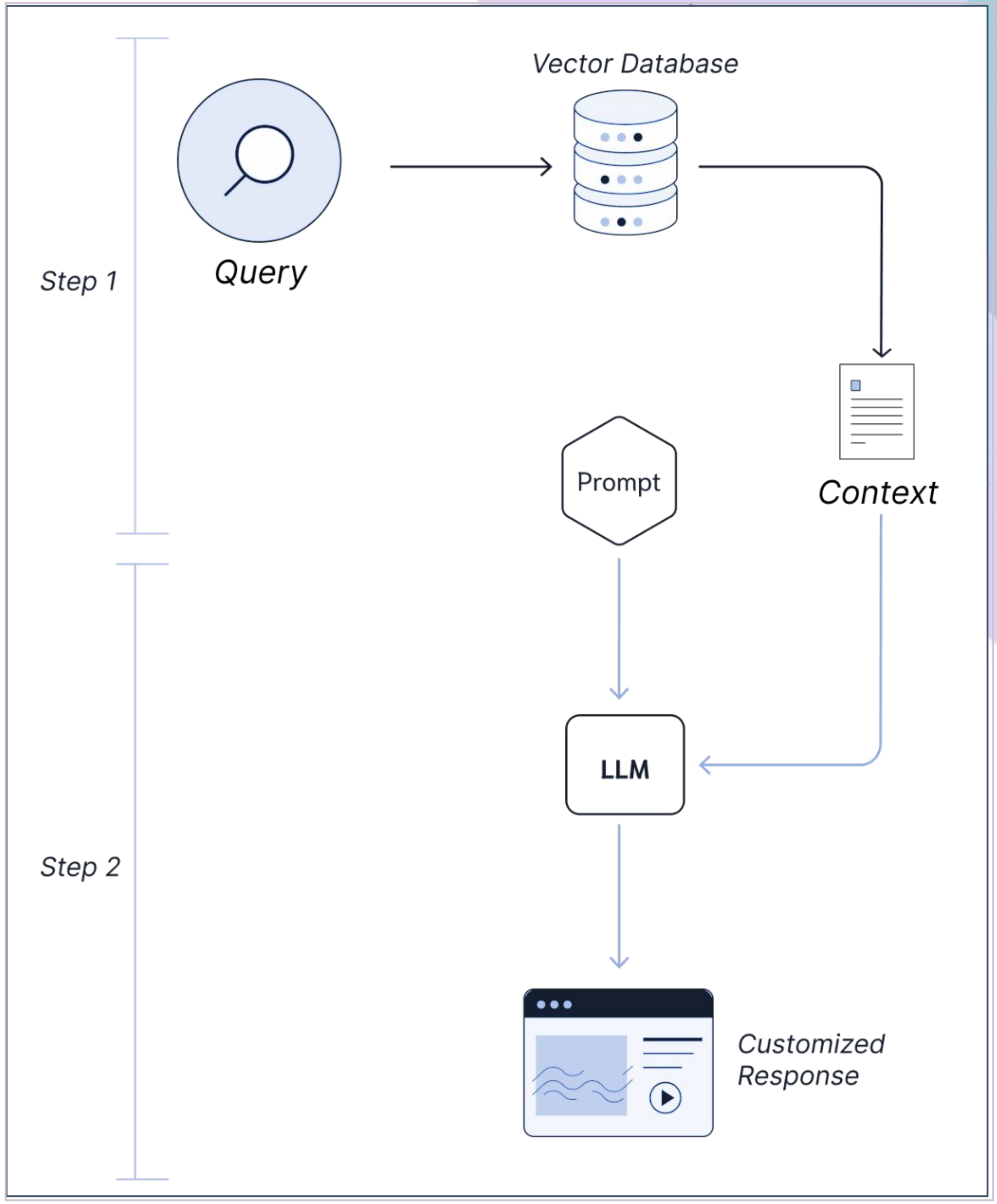
- **RAG: R**etrieval **A**ugmented **G**eneration
  - Makes getting started with AI-driven IR incredibly easy
  - Tons of frameworks out there to help you get started
- **Reranking**
  - Great for domain-specific corpori
  - Generally better than semantic search or lexical search by themselves
  - Easy to integrate into an existing lexical search stack
- **Fine-tuning**
  - Can essentially customize an already-great retrieval model for your domain

RAG

- **TLDR:** Give an LLM access to information it would otherwise not know about
- **Advantages**
  - Hardly any expert-level IR or ML knowledge necessary
  - Easy to spin up e2e RAG apps, POCs
  - Data can be in any format (not just vectors)
  - Drastically lower probability of hallucinations
- **Notes**
  - Since end product is *generated*, does not produce traditional IR outputs, e.g. ranked search results
  - Beholden to LLM context window limitations
- **Ideal use cases**
  - Chatbots
  - Q&A systems
  - Code generation
  - Personalization
- **Popular tools**
  - LlamaIndex
  - LangChain
  - Haystack

RAG: How it works

- Intercept user query en route to LLM
- Send user query to DB
- Retrieve top `k` results
- Add those results to prompt that you send to LLM
- Bam! LLM now knows new information and uses it to answer the user query



Advanced RAG

- RAG Fusion
  - Use LLM to generate versions of user query & execute in parallel, synthesize and rerank results
- Self-Reflective RAG
  - LLM trained to retrieve (or not) on demand, and understand when retrieved context is irrelevant
- Corrective RAG
  - RAG w/a lightweight evaluator in the loop to throw out suboptimal context
- Agentic RAG
  - Category of RAG techniques that use agents (reasoning engines (LLMs) that execute instructions) to orchestrate retrieval
- Small-to-Big Retrieval
  - Use small chunks to grab larger, more semantically coherent chunks to send to LLM
- RAG with HyDE
  - (Hypothetical Document Embeddings) Use LLM to generate hypothetical answer to user query; answer is used to retrieve similar (real) answers as context
- Chain of Abstraction
  - Fine-tune an LLM to generate abstract reasoning chains w/placehodlers for real-world values; using external tools & sources to fill placeholders
- More fine-grained pre- and post-processing of context, e.g.:
  - Context reranking
  - Vector quantization

Advanced RAG, 1: RAG Fusion

- **TLDR**: Use LLM to generate multiple, distinct query rewrites, execute those queries in parallel, synthesize and rerank results
- **Advantages**
  - "*bridge the gap between what users explicitly ask and what they intend to ask*"
- **Notes**
  - Increased latency
- **Ideal use cases**
  - Ambiguous queries, specialized vocabularies
- **Popular tools**
  - OG opensource repo from Adrian H. Raudaschl (Python)
  - LLamaIndex: RAGFusionPipelinePack
  - LangChain: rag-fusion

Advanced RAG, 1: RAG Fusion

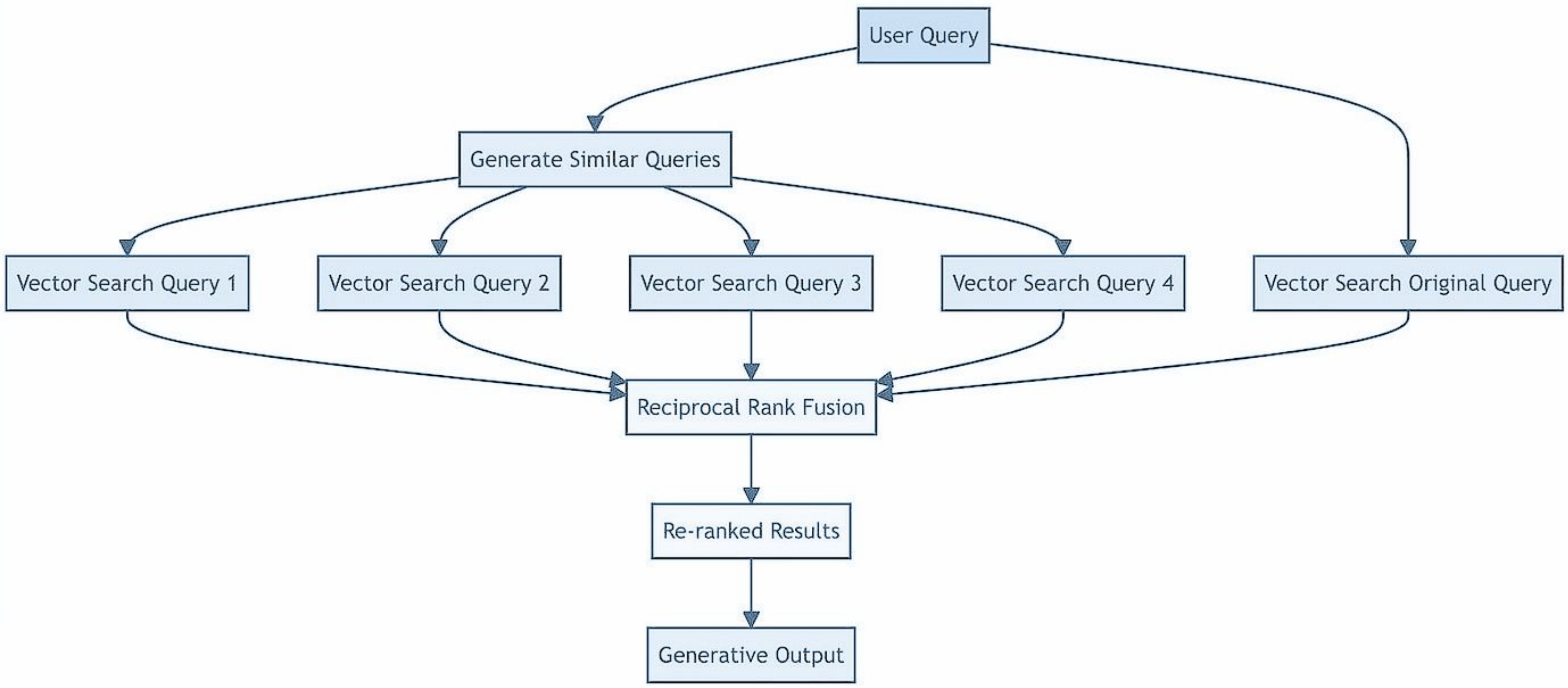


Image by Adrian H. Raudaschl

Advanced RAG, 1: RAG Fusion

Reciprocal Rank Fusion Algorithm (courtesy of Elasticsearch)

```
score = 0.0
for q in queries:
    if d in result(q):
        score += 1.0 / ( k + rank( result(q), d ) )
return score
```

for each query,
if a relevant document is in its result set,
increase score by 1/ranking_constant + the document’s position

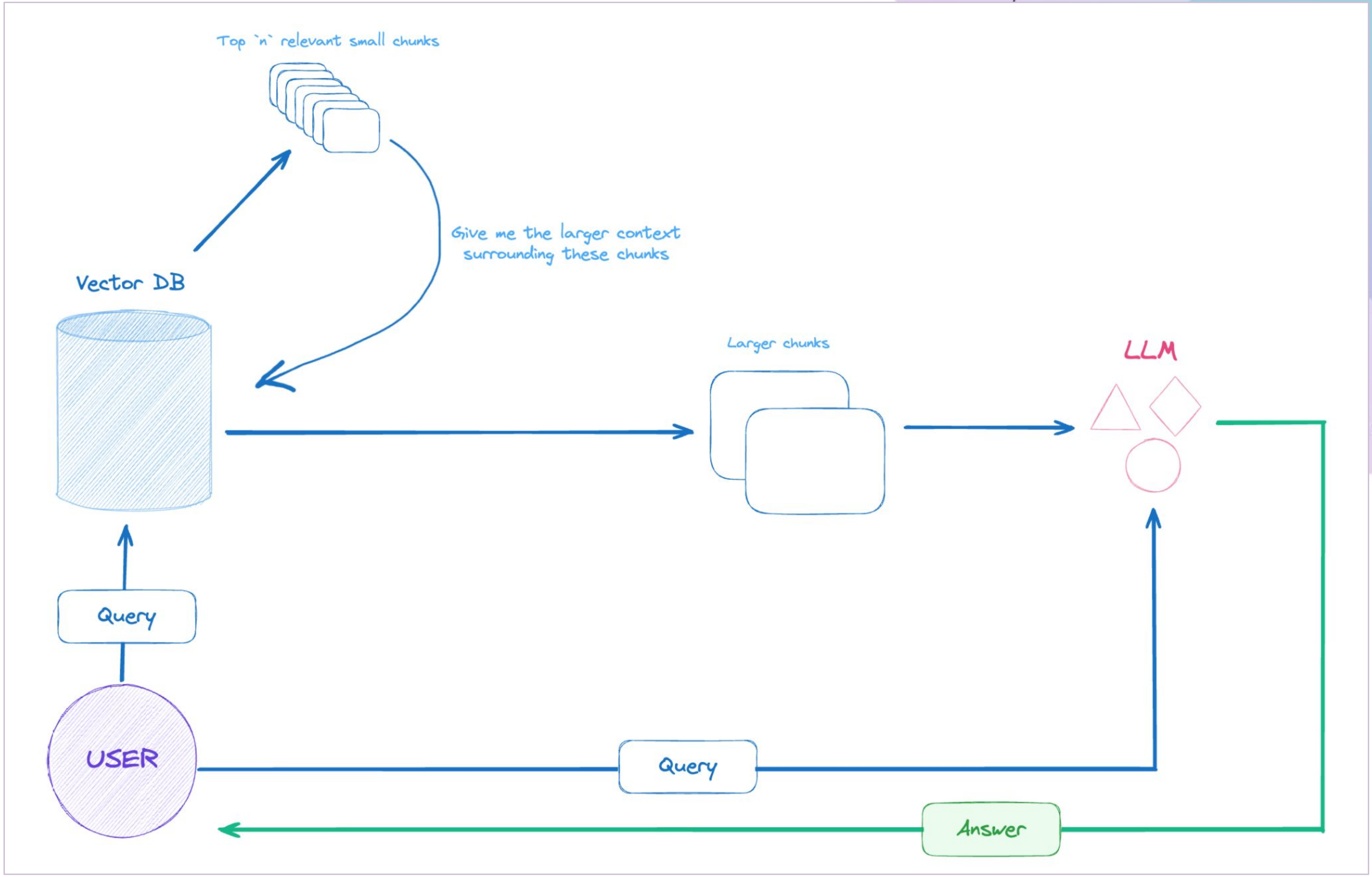
k is a ranking constant (A higher value indicates that lower ranked documents have more influence)
q is a query in the set of queries
d is a document in the result set of q
result(q) is the result set of q
rank(result(q), d) is d's rank within the result(q) starting from 1

Individual result sets can use diff ranking algos!

Advanced RAG, 2: Small-to-Big Retrieval

- **TLDR:** Retrieve small chunks from DB, use small chunks to retrieve larger chunks; send larger chunks to LLM as context
- **Advantages**
  - Optimize precision by fetching small chunks
  - Optimize contextual information passed to LLM via large chunks
- **Notes**
  - Requires more hands-on preprocessing of data
- **Ideal use cases**
  - Everything!
- **Popular tools**
  - LlamaIndex
    - Sentence Window Node Parser: Retrieve at the sentence level, expand chunk window.
    - Recursive Retriever: Retrieve smaller chunk, reference the parent chunk.
  - LangChain:
    - Parent Document Retriever: Fetch the small chunks, look up the parent ids for those chunks and return the larger documents.

Advanced RAG, 2: Small-to-Big Retrieval



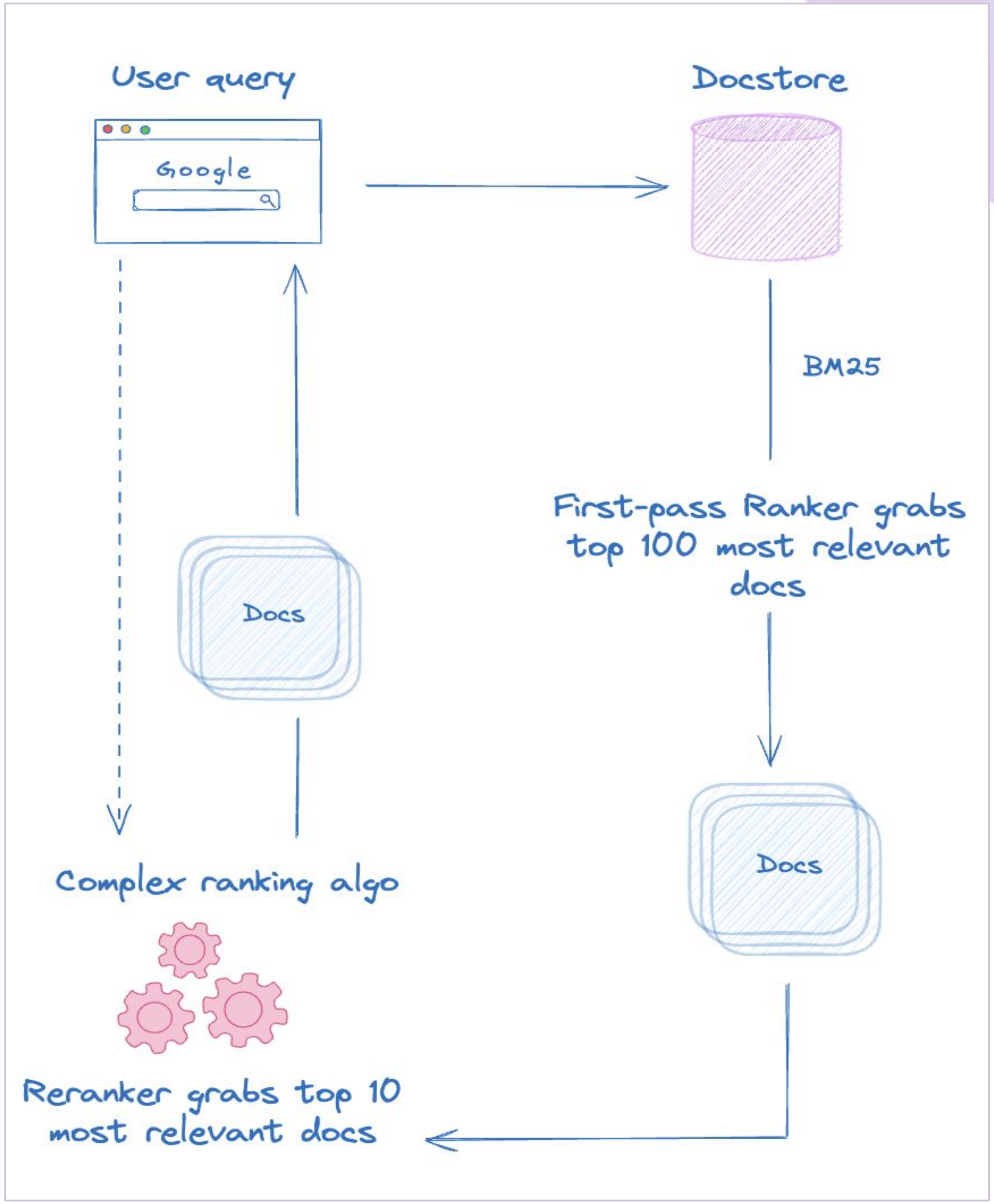
The future of RAG

- More agentic RAG techniques
    - More Agents is All You Need (Feb '24) — Kinda like RAG Fusion, except with multiple instances of an LLM (agents); agents combine/vote for best answer
- Mixed retrieval
    - Retrieve from vector DB, keyword search engine, SQL DB, etc. & combine
- Auto-retrieval
    - Use LLM to infer set of metadata filters & pass rewritten query to external DB
- Evaluation frameworks
- Exciting things are happening *right now*!
    - RAFT: Retrieval Augmented Fine-Tuning (March '24)
        - Finetune an LLM to disregard any retrieved documents that do not contribute to answering a given question, thereby eliminating distractions.
    - DSPy
        - Collection of "compilers" that build the perfect prompt for LLMs

Reranking

- **TLDR:** Reorder `k` documents from `n` originally retrieved documents
  - `n` documents retrieved by "first-pass" ranker, algorithm is optimized for recall
  - `k` reranked documents retrieved by more complex ranking algorithm
- **Advantages**
  - Targeted use of compute resources
  - Known-valuable technique to drastically increase relevance
- **Notes**
  - Increased latency
  - Implementing complex reranking algos requires strong technical knowledge
  - Reranking output is only as good as the first-pass ranker
- **Ideal use cases**
  - Search applications of any kind, as long as they have enough compute resources, where relevance is paramount
- **Popular tools**
  - Cohere (API)
  - RAGatouille (ColBERT)

Reranking


User query
Google
Docstore
BM25
First-pass Ranker grabs top 100 most relevant docs
Docs
Docs
Complex ranking algo
Reranker grabs top 10 most relevant docs

Types of rerankers

- **Cross-encoders (e.g. BGE)**
  - Related to bi-encoders (dense retrieval), except cross-encoders vectorize its 2 sentences simultaneously
  - Simultaneous embedding preserves relationships
  - Slow, but accurate; don't scale well
- **'Bag-of-embeddings' Rerankers (e.g. ColBERT)**
  - Uses transformer models (e.g. BERT), encode queries + docs into mult. embeddings
  - *Late* interaction architecture (vectors come from *last* output layer of model)
  - Doc-Query pair similarity measured by "Maximum Similarity" (MaxSim)
- **LLMs (e.g. RankZephyr)**
  - Use LLMs to identify similarity between query:document pairs

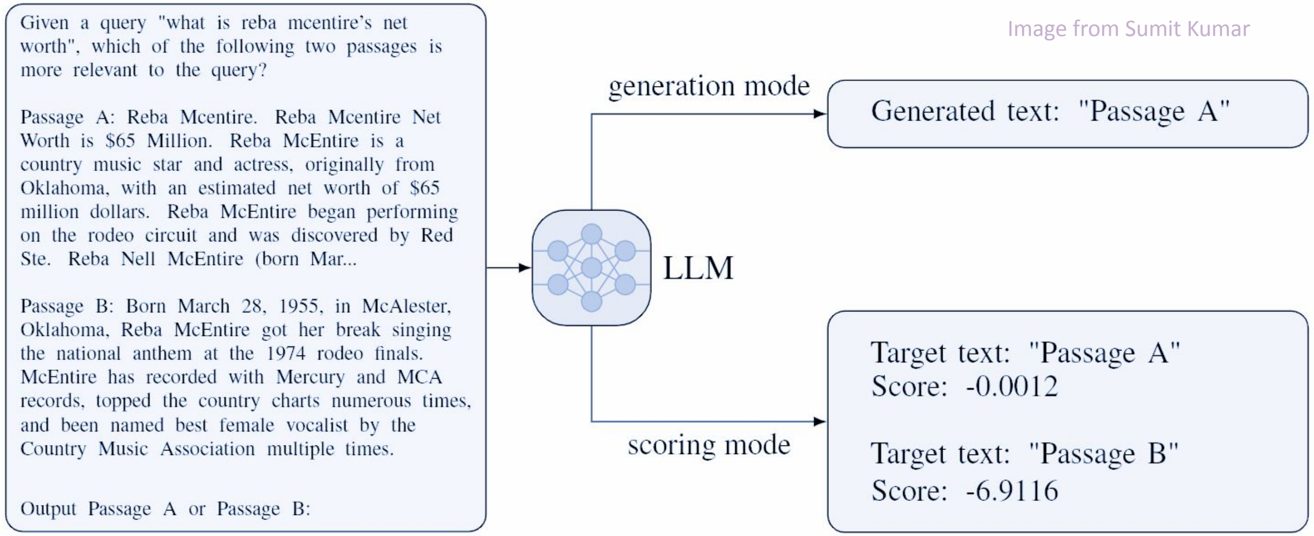


Image from Sumit Kumar

The future of reranking

- More dependence on LLMs for generating reranking schemas
  - **March '24: Instruction-based Unsupervised Passage Reranking (InstUPR) —** Use instruction-tuned LLM(s) to create reranking schemas
  - **March '24: Hierarchy-Aware Reranking —** Use taxonomic (hierarchical) graph of entities diversify and dedupe ranking results
  - **Feb '24: ListT5 —** List-wise reranking for zero-shot reranking that doesn't incur huge computational cost, mitigates 'lost in the middle' problems
  - **Jan '24: InRanker —** Use LLM(s) to generate synthetic, labeled training data to fine-tuning

Fine-Tuning Pretrained Models

- **TLDR:** Customize already-great model for a specific use
- **Advantages**
  - Specialized for a specific task/domain
  - Adapts easily to tone, style and vocabulary as desired
- **Notes**
  - Have to re-fine-tune as information changes
  - Less generalizable, can be overfit if training data is too small
  - Need someone w/advanced ML skills to implement
  - Black box interpretability
- **When to avoid**
  - New task is orthogonal to pretrained model
  - Business domain/data is constantly changing
- **Ideal use cases**
  - When data is restricted and/or sensitive, e.g. medical field
  - When precision is very important
  - When specific brand tone, style matter
- **Popular tools**
  - HuggingFace
  - OctoML
  - PromptLayer
  - Together.AI
  - OpenAI

RAG *and* Fine-tuning

Why not both??

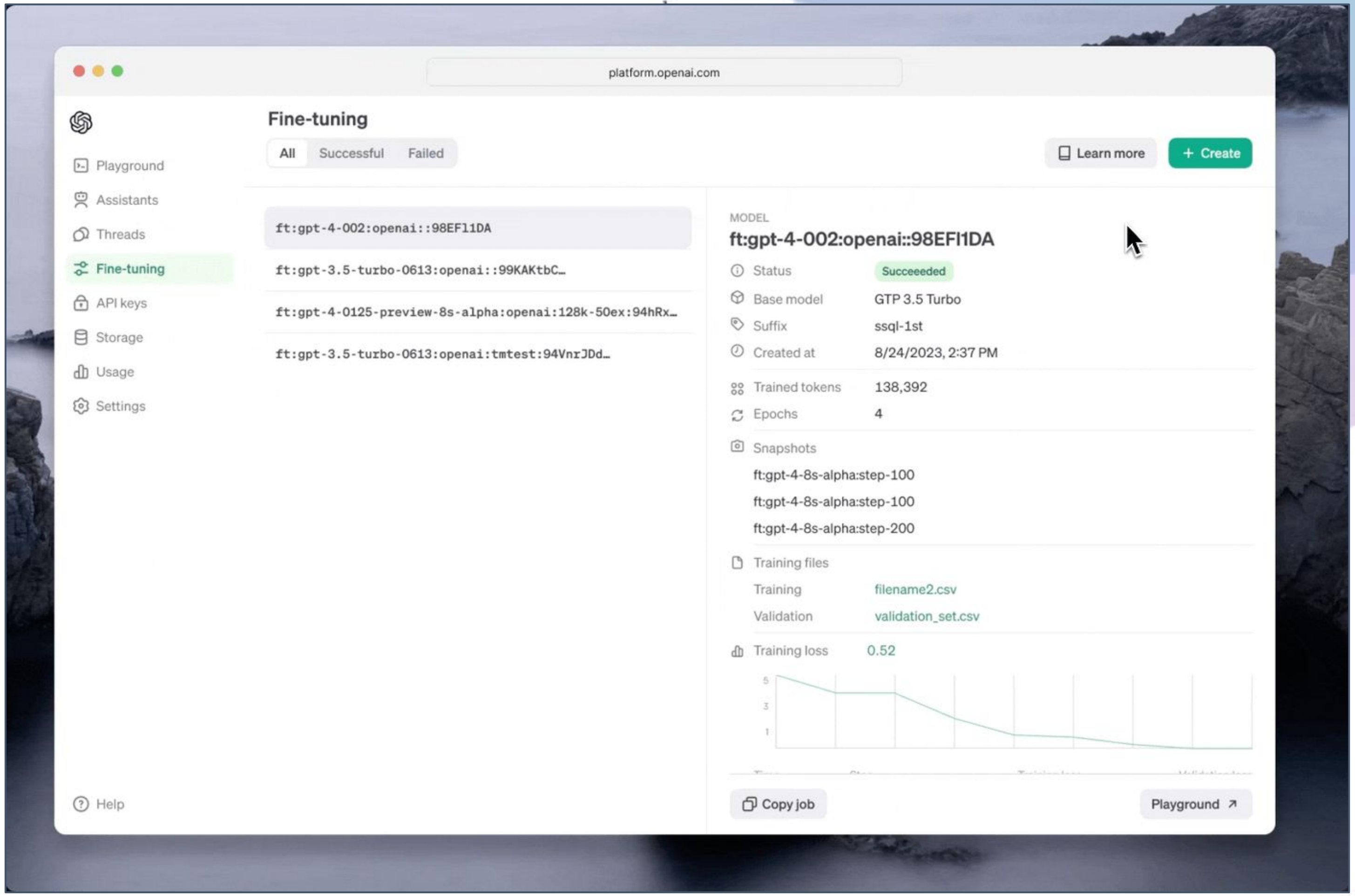
- RAG goal = accurate retrieval
- Fine-tuning goal = specific output
- Ideal use case:
  - Any app that relies on external, dynamic data, but that also requires unique customization or domain expertise
  - E.g.: chat assistant that needs to know most recent data, but also needs to serve it to end user in specific tone or format

Popular Methods of Fine-Tuning

- ⭐ **Supervised**
  - **TLDR:** Model understand what ‘correct’ answers look like
  - **Need:** pairs of high-quality, labeled data
- **Reinforcement-Learning with Human Feedback**
  - **TLDR:** Model decides what action to take based on feedback
  - **Need:** triples of state:action taken:feedback
- **Unsupervised**
  - **TLDR:** Model (e.g. LLM) uses data itself as training, e.g. “masked language modeling”
  - **Need:** Unlabeled data

The future of fine tuning

- **Parameter-efficient fine tuning** (PEFT, March '24)
  - Freeze the weights of OG model; add new weights; fine-tune new weights on a new training dataset.
  - E.g. Low-Rank Adaptation (LoRA)
- **Representation fine-tuning** (REFT, April '24)
  - Like PEFT, but edit weights instead of replace subset
- **Prompt-oriented unsupervised fine-tuning** (POUF, April '23)
  - Model figures out what you want it to do by you continuously giving it hints in the form of prompts



The future is mixed

Mixed methods

- RAG + Reranking
- RAG + Fine-tuning
- RAG + Reranking + Fine-tuning
- Agentic methods that do all of this and more!
- Hybrid search, RAG, reranking, etc etc etc :infinity:

The art of search engineering is now the ability to know which techniques work for which use cases, and how to combine these techniques in a way that meets production demands.

+ Evals! 😊

Thank you!