RAG: Retrieval Augmented Generation - Lecture 5

Dmitry Kan, PhD

Syllabus

Week 1: Introduction to Generative AI and Large Language Models (LLM)

- Introduction to Large Language Models (LLMs) and their architecture
- Overview of Generative AI and its applications in NLP
- Lab: Tokenizers

Week 2: Using LLMs and Prompting-based approaches

- Understanding prompt engineering and its importance in working with LLMs
- Exploring different prompting techniques for various NLP tasks
- Hands-on lab: Experimenting with different prompts and evaluating their effectiveness

Week 3: Evaluating LLMs

- Understanding the challenges and metrics involved in evaluating LLMs
- Exploring different evaluation frameworks and benchmarks
- Hands-on lab: Evaluating LLMs using different metrics and benchmarks

Week 4: Fine-tuning LLMs

- Understanding the concept of fine-tuning and its benefits
- Exploring different fine-tuning techniques and strategies
- Hands-on lab: Fine-tuning an LLM for a specific NLP task

Week 5: Retrieval Augmented Generation (RAG)

- Understanding the concept of RAG and its advantages
- Exploring different RAG architectures and techniques
- Hands-on lab: Implementing a RAG system for a specific NLP task

Week 6: Use cases and applications of LLMs

- Exploring various real-world applications of LLMs in NLP
- Discussing the potential impact of LLMs on different industries
- Hands-on lab: TBD

Week 7: Final report preparation

- Students work on their final reports, showcasing their understanding of the labs and the concepts learned.

Prereqs

- LLMs
- Embeddings and embedding models
- Vector Search

Outline

- LLMs strengths vs weaknesses
- RAG
- Embeddings and vector search
- Practical dive into RAG
- Lab assignments

LLMs: strengths (class)

- Summarization
- Tutor to help you understand tasks
- Generates fluent, accurate language
- Versatility: coding, math, data analysis
- Help with mundane work
- Paraphrasing, grammar help
- Might be less biased than a single person
- Can be customized
- You can use own words and talk to it
- LLMs abstracted keyword search

LLMs: strengths (Dima)

- Advanced language understanding
- Human-like text generation capability (vs 10 blue links in Google)
- Few-shot / 0-shot learning
- Versatile applications: sentiment analysis, content creation, customer support automation, coding assistance and more (Week 6)
- || architecture, scalable and relatively simple architecture

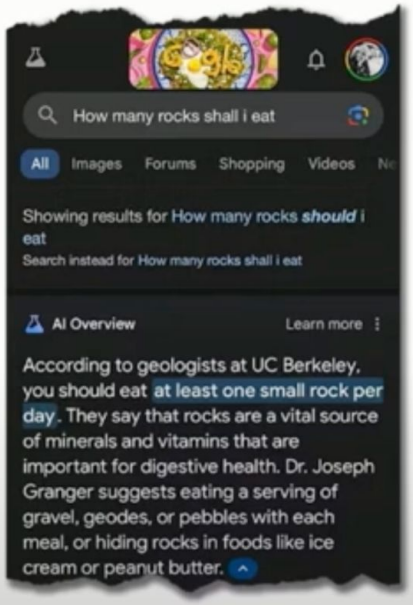
LLMs: weaknesses (class)

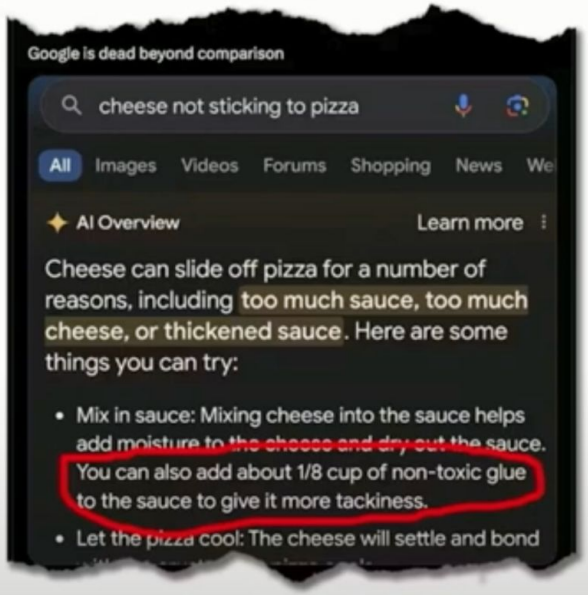
- Hallucinations: makes up articles
- Quality is dependent on training data
- Cost, energy (spend and inefficiency), environmental impact
- Overly confident
- Sweeps at the surface
- Does not have citations
- Can be used to create fake news, propaganda
- Might be too flattering
- Need to know prompting
- Biased, that comes from data
- Delete data => forget knowledge
- Data privacy

LLMs: weaknesses (Dima)

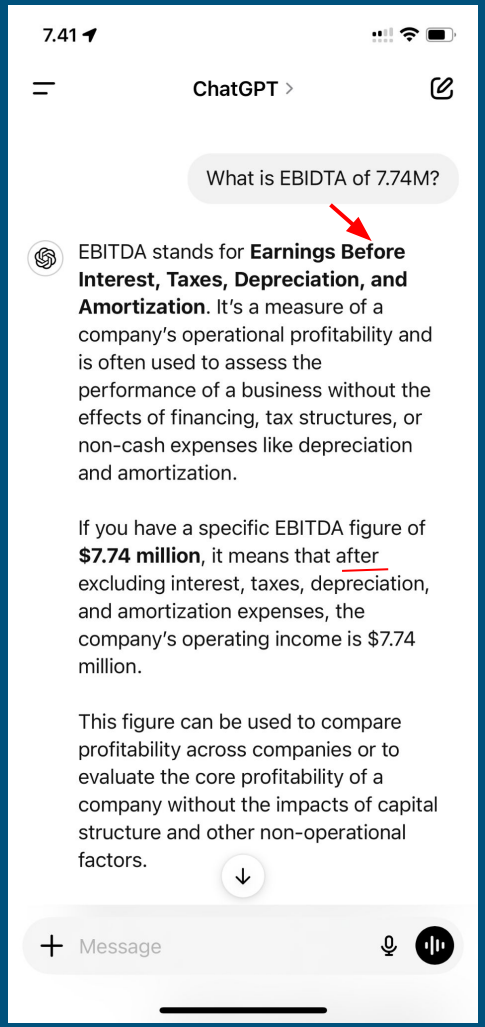
- **Not up to date** with the changing world / domain
- Do not provide **references** to the answer (like 10 blue links in Google)
- Can **hallucinate** (lack of specialized knowledge)
- **Dependence on context**: lack depth, accuracy or *logical consistency*
- **Need lots of GPU / compute** (GPU RAM, CPU latency)

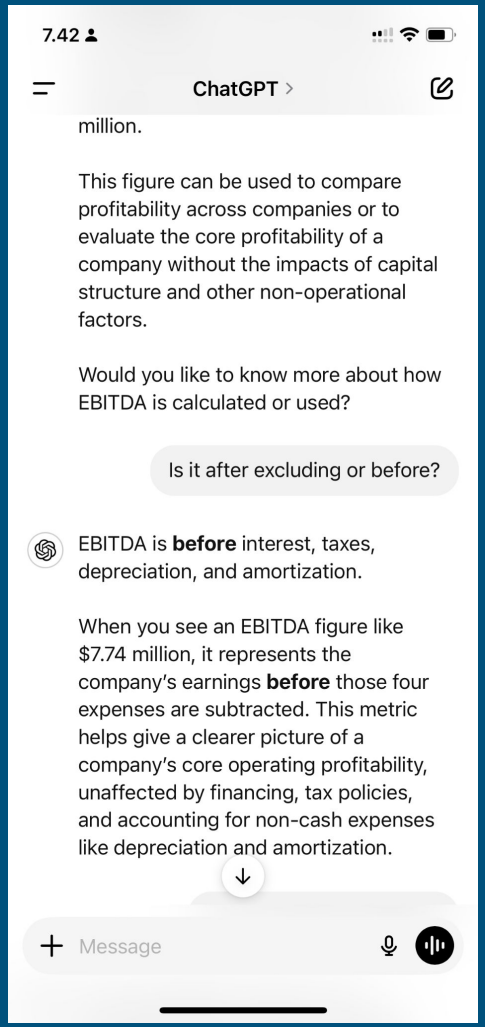
Example of a hallucination

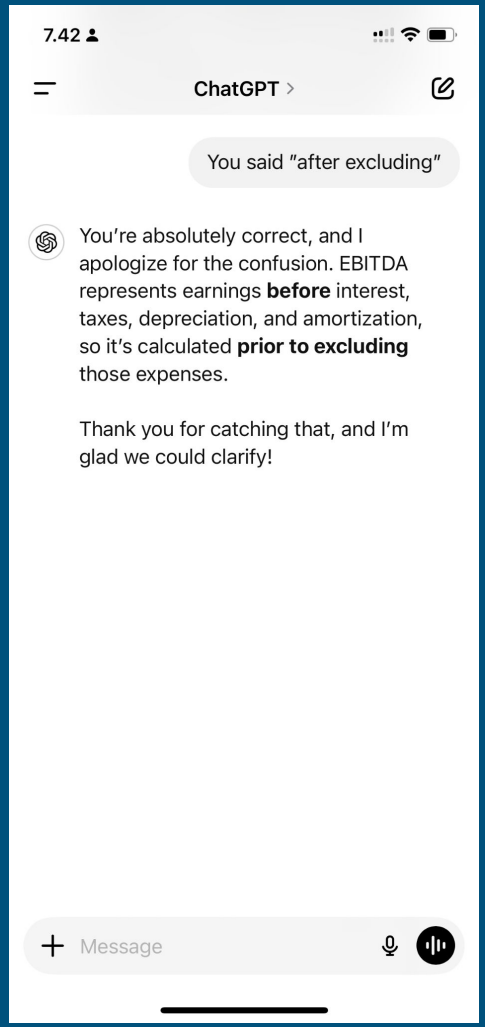




Ref :
https://theconversation.com/eat-a-rock-a-day-put-glue-on-your-pizza-how-googles-ai-is-losing-touch-with-reality-230953







Before or after?

Hottest 3 letter word in Gen AI right now.. **RAG** !!!

Credit: Atita Aurora, talk from Berlin Buzzwords'24

RAG: a possible solution

Retrieval-Augmented Generation (RAG) is a two-phase process involving *document retrieval* and *answer formulation* by a Large Language Model (LLM). The initial phase utilizes *dense embeddings* to retrieve documents. This retrieval can be based on a variety of database formats depending on the use case, such as a vector database, summary index, tree index, or keyword table index.

https://en.wikipedia.org/wiki/Prompt_engineering#Retrieval-augmented_generation

“Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks”

- Published Apr, 2021 by Patrick Lewis (co:here) et al (FAIR)
- Was an “accidental” find by a team at Facebook
- Adds domain or topical grounding to an LLM
- https://arxiv.org/pdf/2005.11401

RAG: architecture

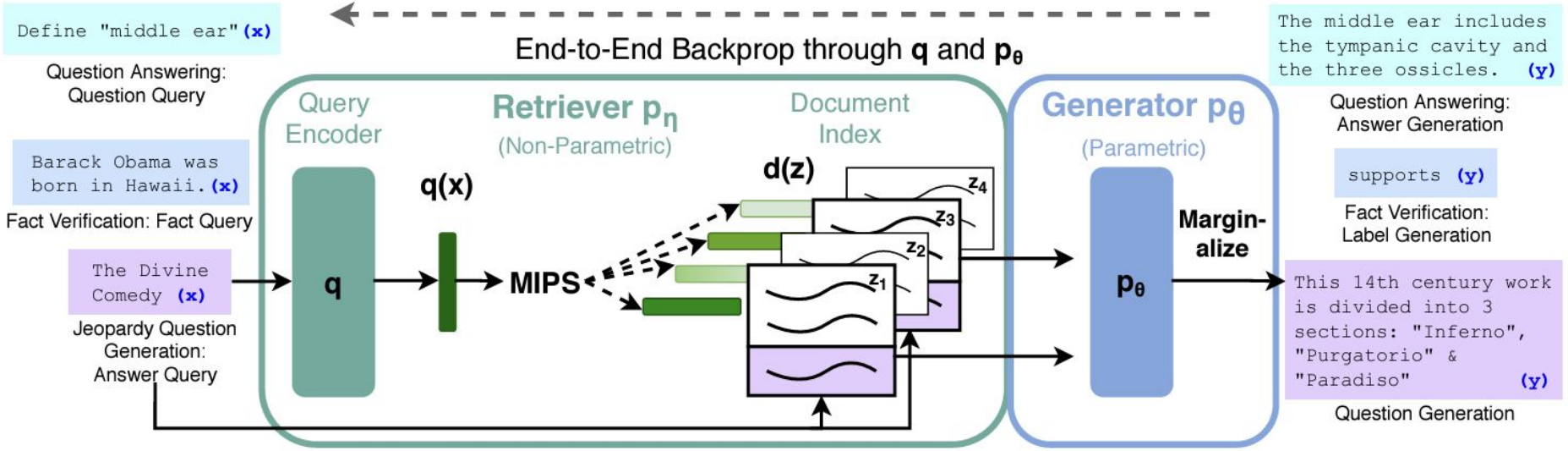


Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder* + *Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i. For final prediction y, we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

RAG paper: key points

RAG Models:

- RAG-Sequence Model: uses retrieved doc to generate complete sentence
- RAG-Token Model: several documents are used to draw next token

Retriever:

- DPR: Dense Passage Retriever, where BERT is used as doc and query encoder

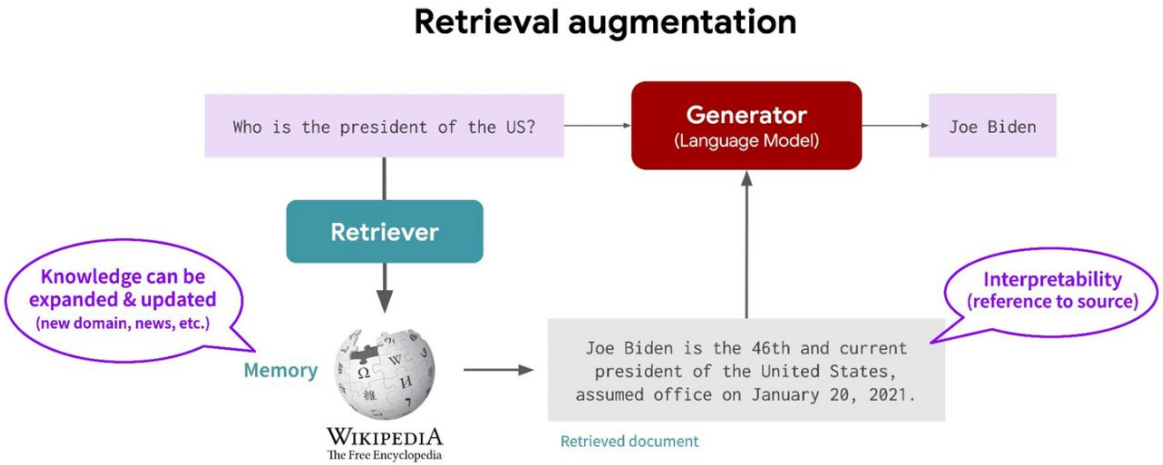
Generator: BART

- Pre-trained model: **B**idirectional + **A**uto-**R**egressive **T**ransformers

RAG

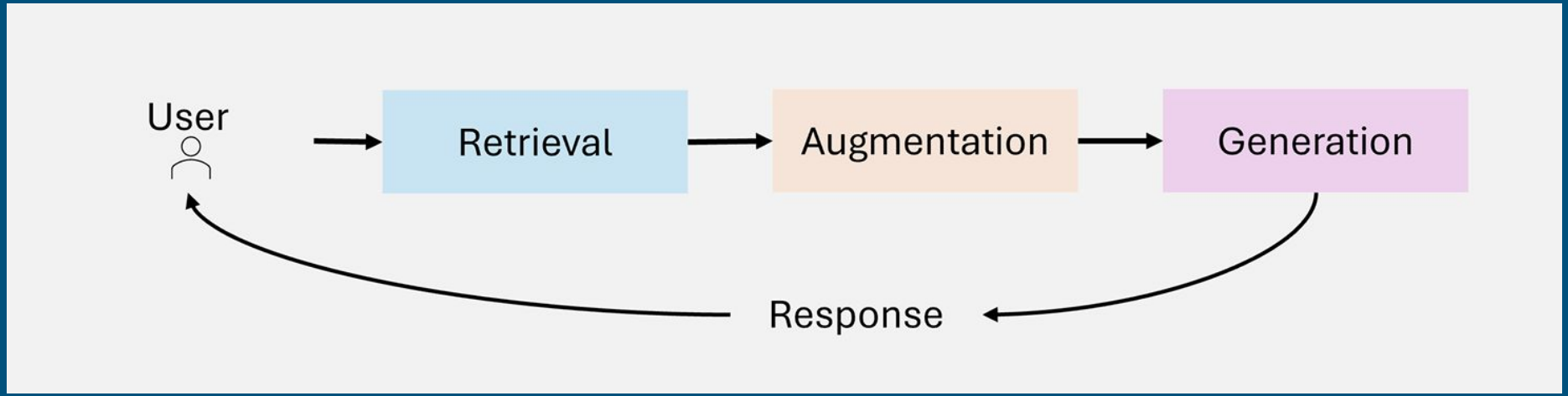
The most popular current application with LLM.

- Simple and human-interpretable alternative to finetuning in LLMs.
- Adds a level of grounding for generated results.
- For some applications (e.g. user data), scale is less important compared to consistency and availability at streaming updates.

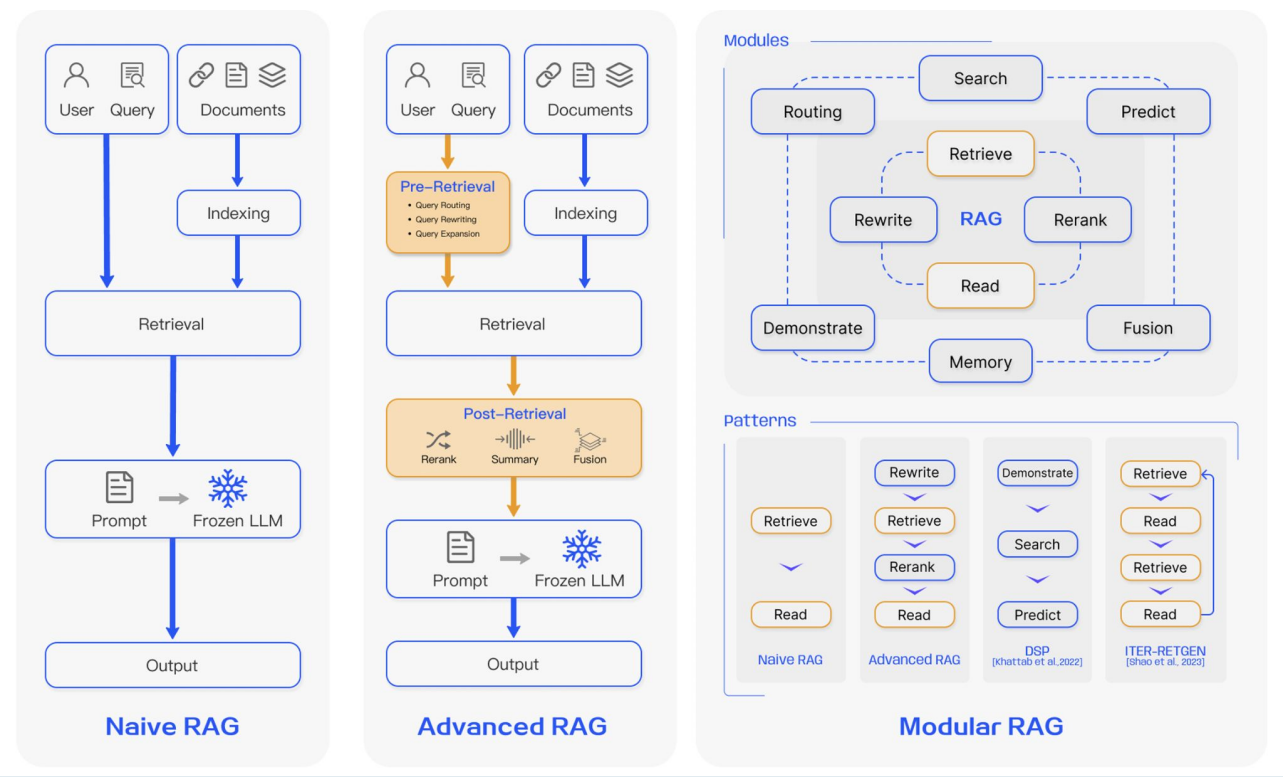


https://big-ann-benchmarks.com/neurips23_slides/ANNS_for_recommendation_systems_Yury.pdf

Simply put



Types of RAG



Pre-Retrieval:

- Query rewriting / expansion
- Add metadata
- Hybrid retrieval (keywords + vectors)

Post-retrieval:

- Chunk re-ranking
- Context compression

Modules:

- Fusion = || vector search + re-ranking

RAG vs Fine-tuning

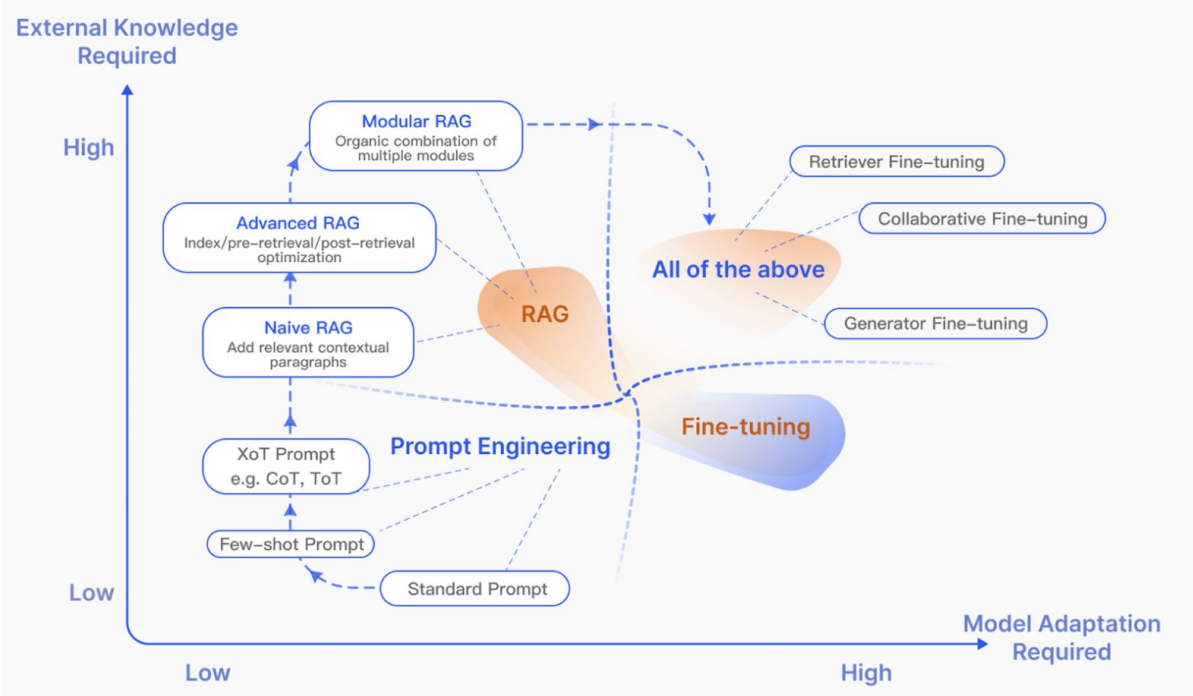


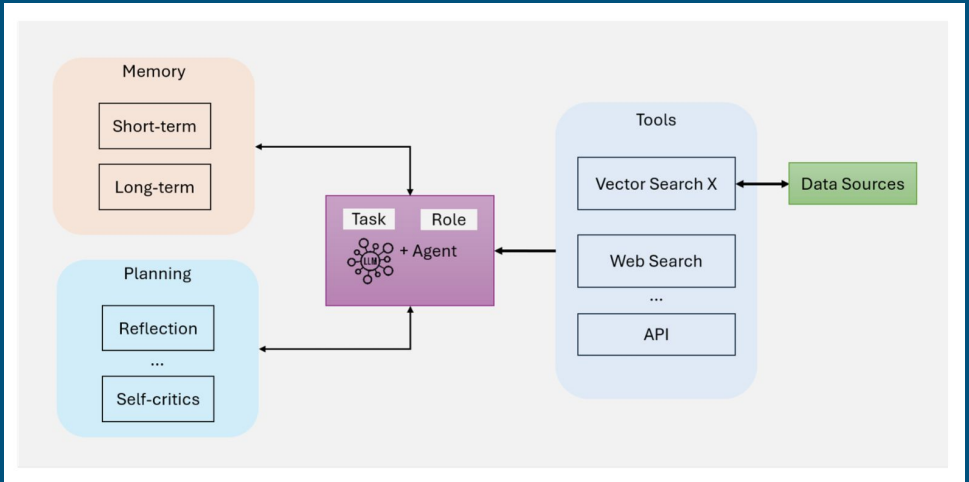
Fig. 4. RAG compared with other model optimization methods in the aspects of "External Knowledge Required" and "Model Adaption Required". Prompt Engineering requires low modifications to the model and external knowledge, focusing on harnessing the capabilities of LLMs themselves. Fine-tuning, on the other hand, involves further training the model. In the early stages of RAG (Naive RAG), there is a low demand for model modifications. As research progresses, Modular RAG has become more integrated with fine-tuning techniques.

Agentic RAG

- Paradigm shift from ‘static’ RAG systems by introducing autonomous agents
- Agents:
  - dynamic decision-making
  - Workflow optimization
- Iterative refinement
- Adaptive retrieval
- Address complex, real-time, multi-domain queries

AI Agent: components

- **LLM** (with defined role and task)
- **Memory**
  - Short-term tracks immediate conversation state
  - Long-term: accumulated knowledge and experiences
- **Planning**
  - Reflection, query routing, self-critique
- **Tools**
  - Vector / web search, APIs



LLM

System prompt (role+task) for the agent:

"You are an AI Research Librarian helping students in an LLM/RAG course.
Your job:

- Understand the student's research question.
- Search their local paper library and the web for relevant sources.
- Summarize and explain papers in simple language.
- Keep track of what the student has already read and their preferences.
  Always:
- Show which sources you used.
- Admit when you're unsure and suggest how to verify."

Memory (short-term)

User: “I want to build a RAG system for legal contracts. What should I read first?”

Agent (internally tracks in **short-term** memory):

- Current topic: **RAG for legal contracts**
- User level: says they’re in an **LLM/RAG course**, so likely intermediate
- Last question: **“what should I read first?”** (guides next response)

```json
{
  "current_topic": "RAG for legal contracts",
  "user_level": "intermediate student",
  "last_question": "what should I read first?"
}
```

Memory (long-term)

- Accumulates over time, over multiple interactions with the user
- Won’t recommend the papers the user has already read for their question: “How do I evaluate my RAG system?”

json

```
{
  "user_profile": {
    "name": "N/A or anonymized",
    "background": "CS undergrad, some Python, new to RAG",
    "interests": ["RAG", "legal domain", "evaluation"],
    "reading_history": [
      "Lewis et al. 2020 - Retrieval-Augmented Generation",
      "Self-RAG 2023",
      "GraphRAG blog (Microsoft, 2025)"
    ],
    "preferences": {
      "wants_simple_explanations": true,
      "likes_concrete examples": true
    }
  }
}
```

Planning

“Give me three key recent papers on GraphRAG for enterprise search and summarize how they differ.”

1. Clarify and interpret the task
    a. Topic: GraphRAG
    b. Domain: enterprise search
    c. Output: 3 papers + comparison summary
2. Query routing: local paper index first, then web search if < 3 papers
3. Retrieval: "GraphRAG enterprise retrieval knowledge graph RAG", year >= 2024
4. Draft an answer: “Paper A vs B vs C”
5. Reflection / self-critique: “Did I use at least 2 explicitly graph-based RAG methods? If not, redo retrieval.”

Planning #2

5. If 1 paper is missing or too generic, run a new vector search query: "knowledge graph retrieval-augmented generation"

6. Respond to user

- Show paper names, 2–3 line summary each.
- End with a 1–2 paragraph comparison.

Tools

Tool: `papers_vector_search(query, top_k, filters)`

json

Copy code

```
{
  "tool": "papers_vector_search",
  "input": {
    "query": "GraphRAG knowledge graph retrieval-augmented generation for enterprise sear
    "top_k": 10,
    "filters": { "year": { "gte": 2024 } }
  }
}
```

AI Agent can:

- Decide whether to retrieve information or not
- Decide which tool to use to retrieve relevant information
- (Re)formulate the query itself
- Evaluate the retrieved context and decide whether it needs to re-iterate

	Vanilla RAG	Agentic RAG
Access to external tools	No	Yes
Query pre-processing	No	Yes
Multi-step retrieval	No	Yes
Validation of retrieved information	No	Yes

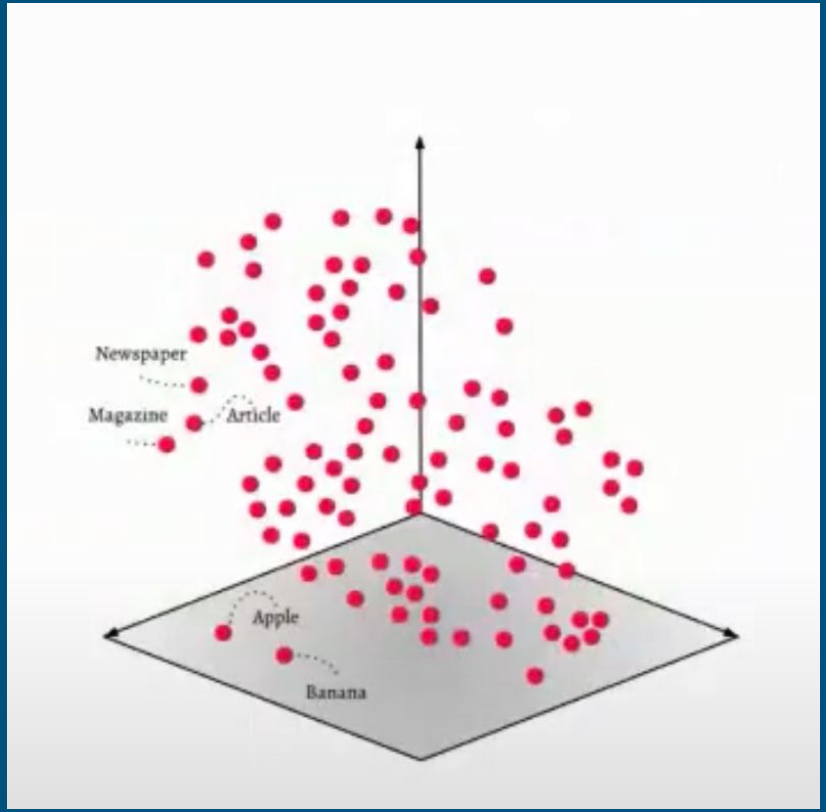
https://weaviate.io/blog/what-is-agentic-rag#:~:text=is%20Agentic%20RAG%3F-,%E2%80%8B,of%20the%20non%2Dagentic%20pipeline

Vector search in a nutshell

Vector search is a way to represent and search your objects (documents, songs, images..) in a geometric space (usually of high-dimension) in the form of an embedding (a vector of numbers: [0.9, -0.1, 0.15, ...])

- At small scale you can apply exact KNN search
- At larger scale you need to use ANN search:

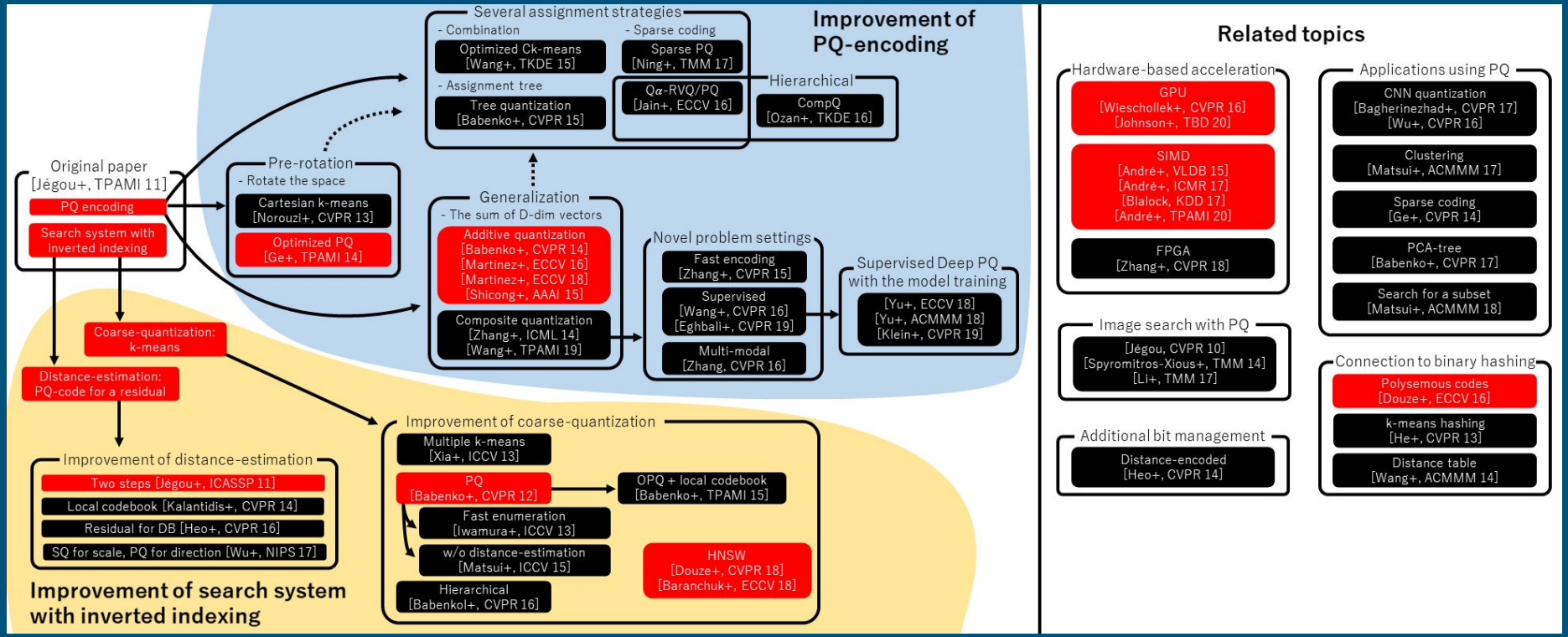
  trade some precision for speed



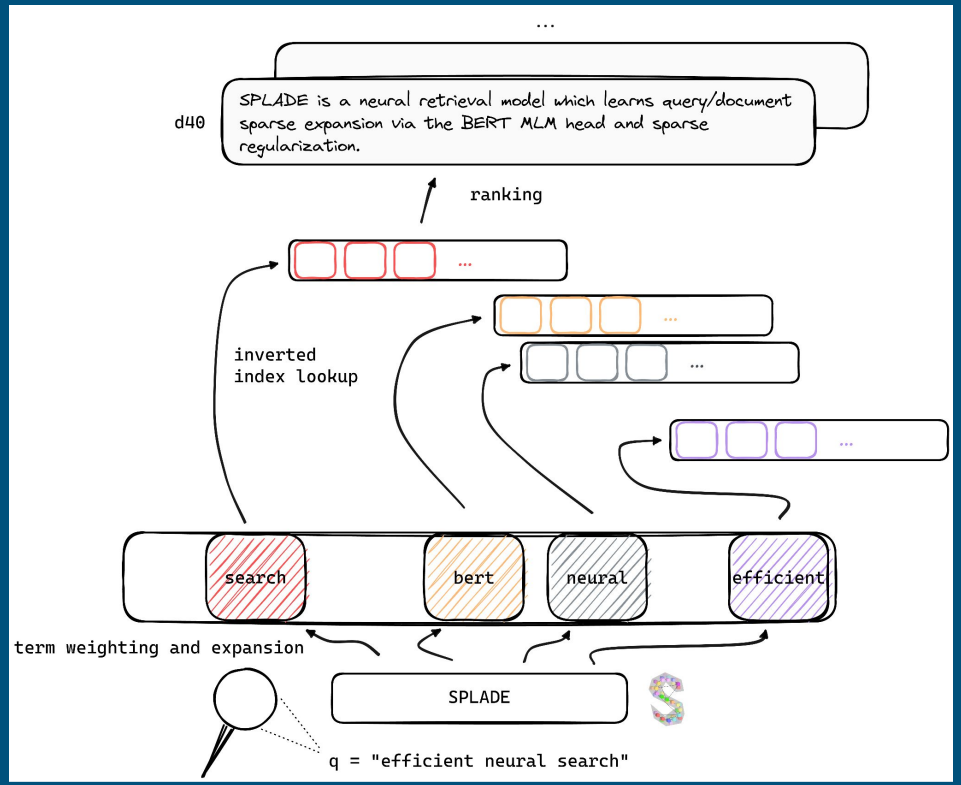
Credit: Weaviate V1.0 release - virtual meetup

ANN algorithms

Source: FAISS



SPLADE: Sparse Lexical and Expansion Model for First Stage Retrieval



Use cases

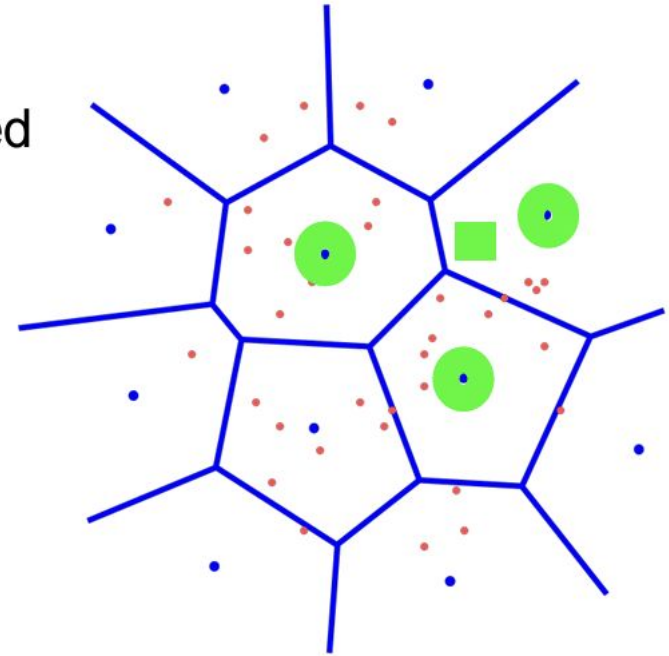
- Semantic search
- Image similarity
- Sound search
- Multimodality: searching images with text
- Recommenders
- E-commerce zero-hit long-tail (similarity search)

Big players in the game

- Spotify: ANNOY (Erik Bernhardsson, ex-Spotify, CEO Modal Labs)
- Microsoft (Bing team): Zoom, DiskANN, SPTAG
  - Azure Cognitive Search
- Amazon: KNN based on HNSW in OpenSearch (via nmslib)
- Google: ScaNN
- Yahoo! Japan: NGT (the fastest algorithm)
- Facebook: FAISS, PQ = Product Quantization (CPU & GPU)
- Baidu: IPDG (Baidu Cloud)
- Yandex
- NVidia
- Intell

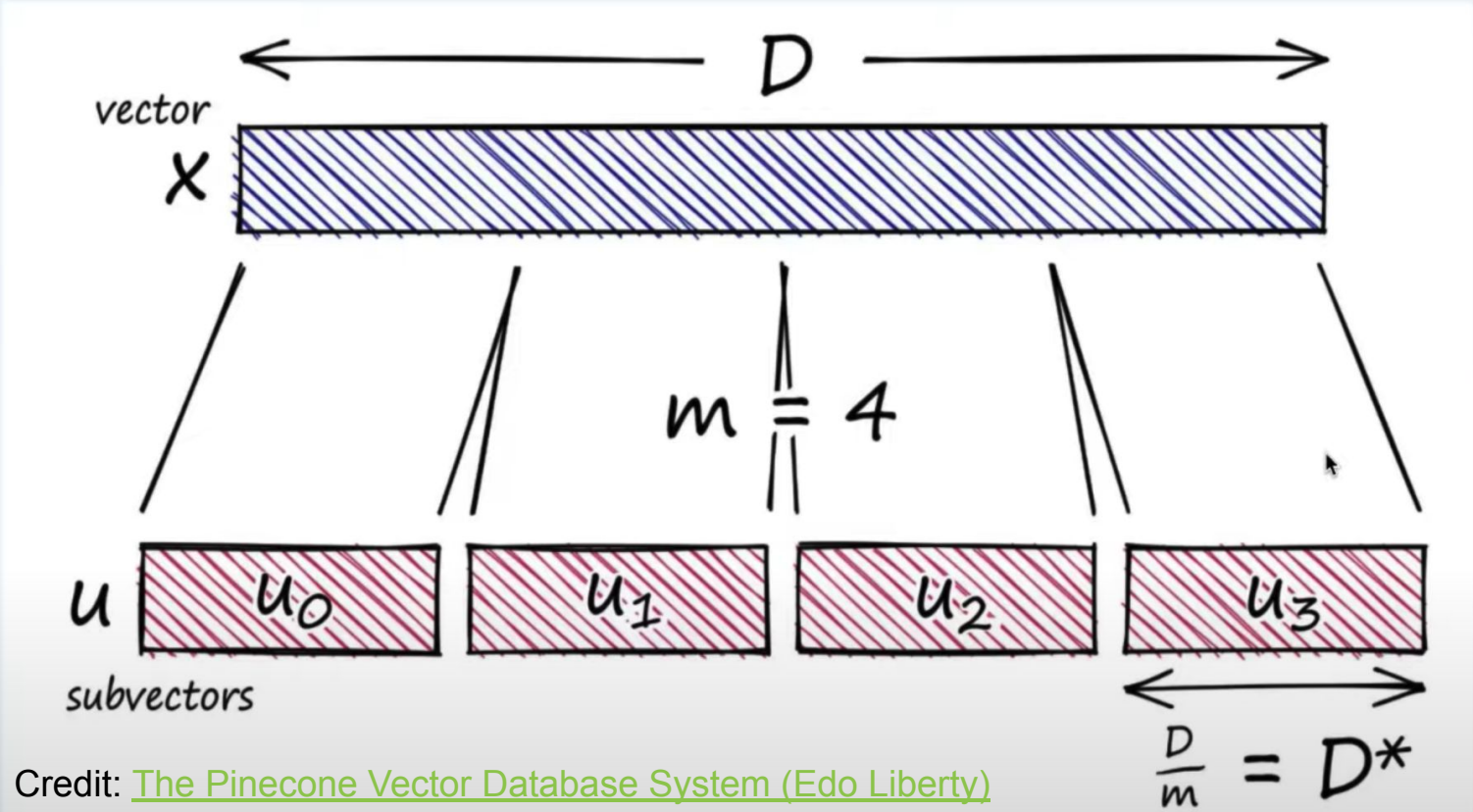
The Inverted File

- Cluster the space into k clusters of vectors
  - assign vectors to nearest centroid
  - Aka. “coarse quantizer”
- index = inverted list structure
  - maps centroid id → list of vectors assigned to it
- search procedure:
  - 1. find np << k nearest centroids to query vector
  - 2. scan the lists corresponding to the np centroids
- Objective: reduce the number of distance computations!



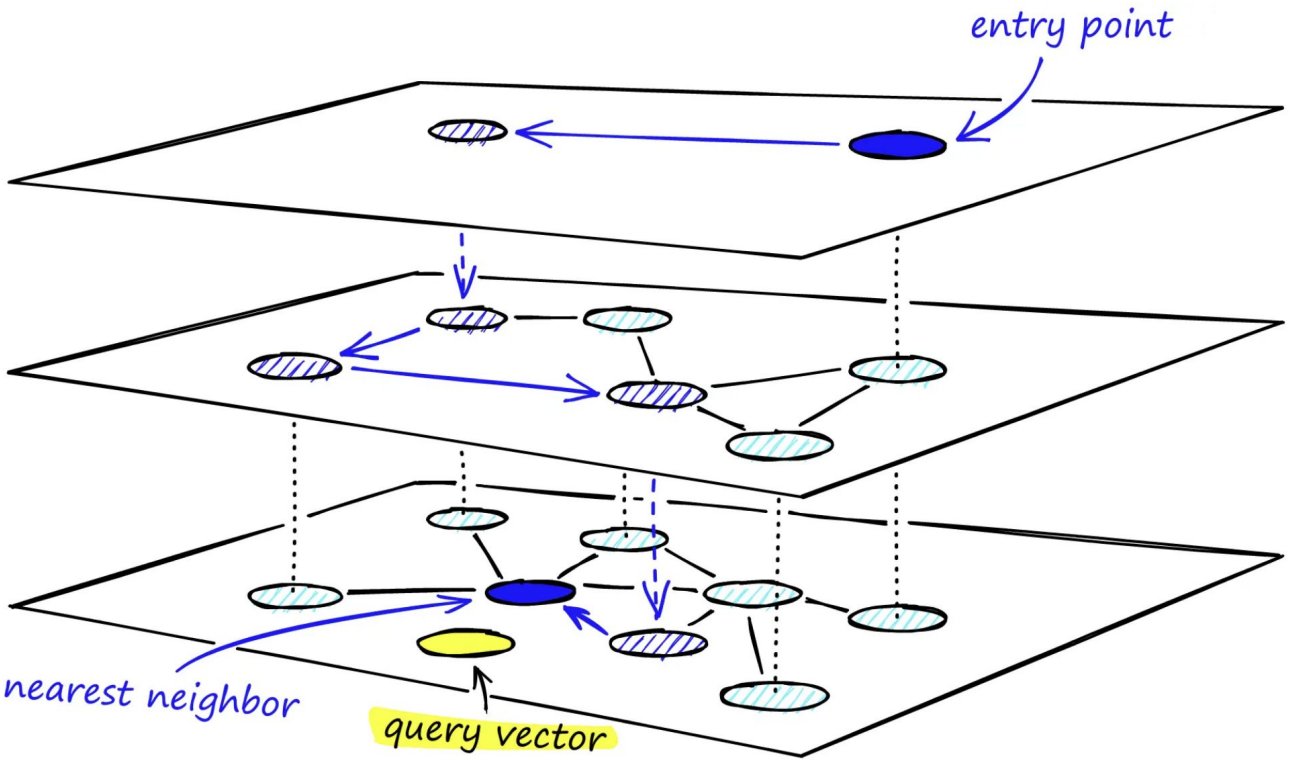
https://edoliberty.github.io/vector-search-class-notes/class_notes/Class_08_quantization.pdf

PQ (Product Quantization)



Credit: The Pinecone Vector Database System (Edo Liberty)

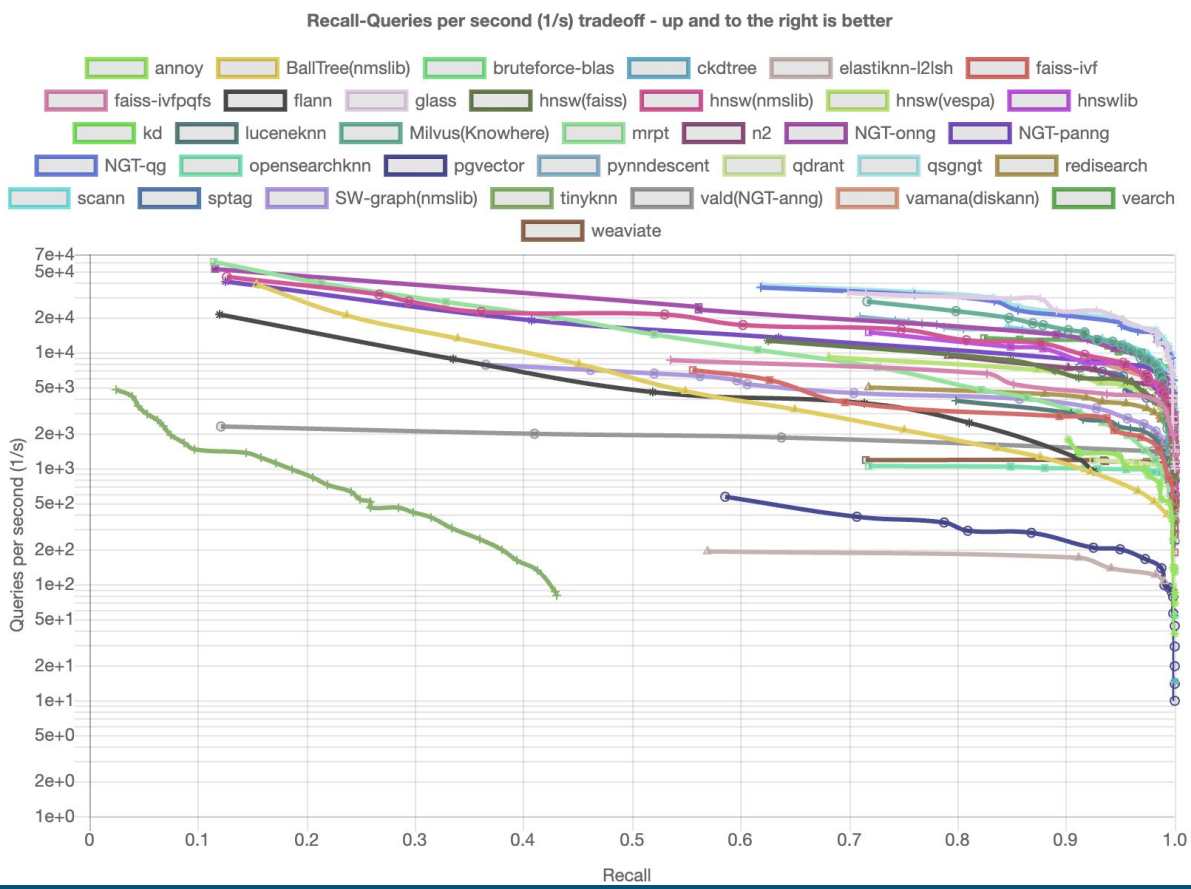
Hierarchical Navigable Small Worlds (HNSW)



The search process through the multi-layer structure of an HNSW graph.

Plots for fashion-mnist-784-euclidean (k = 10)

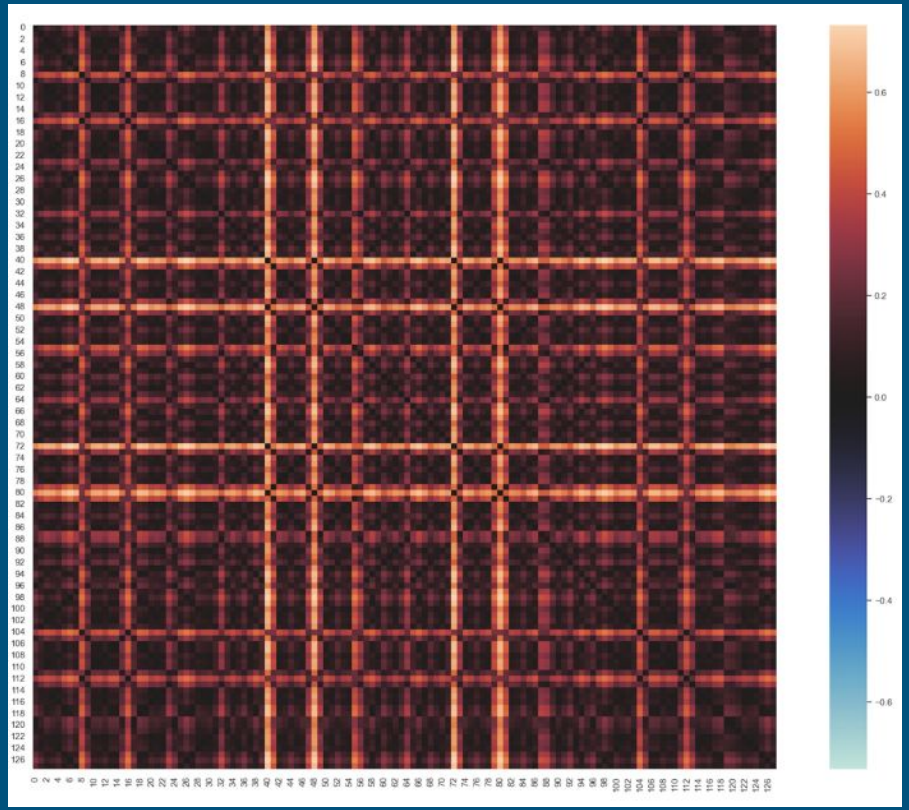
Recall/Queries per second (1/s)



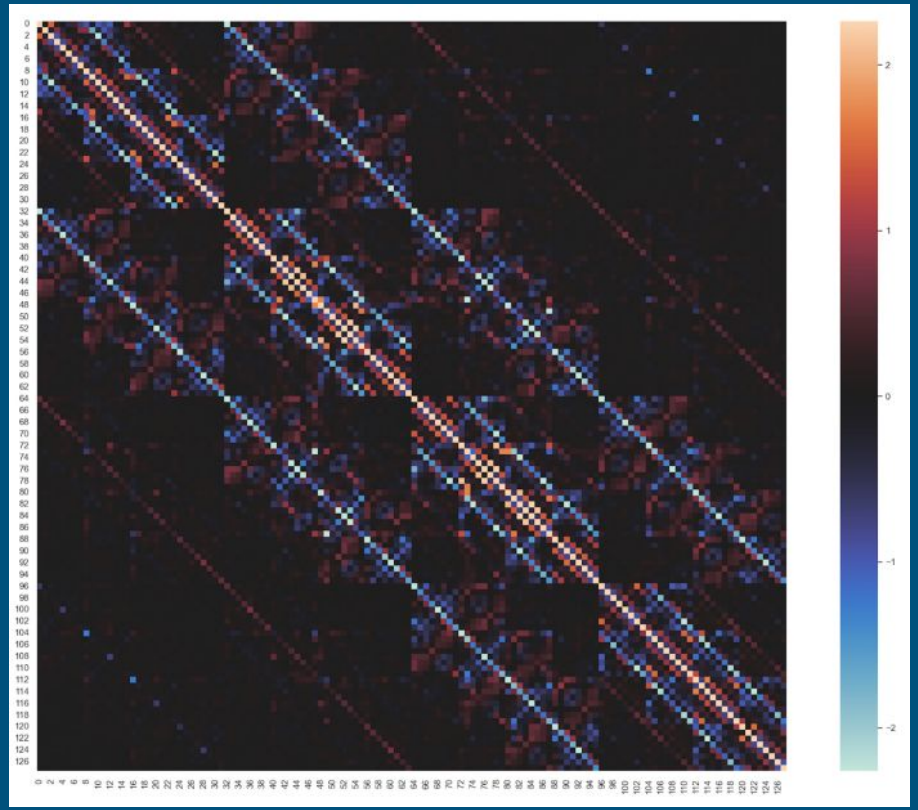
ANN benchmarks: https://ann-benchmarks.com/fashion-mnist-784-euclidean_10_euclidean.html (million scale)

Billion-Scale ANN benchmarks: https://big-ann-benchmarks.com/

BuddyPQ: improving over FAISS (Max Irwin, Alex Semenov, Alex Klibisz, Leo Joffe, Aarne, Dima): https://tinyurl.com/ystumd4u



BIGANN dataset Kolmogorov-Smirnov dimension test matrix for the first 100000 points. A higher number indicates a less similar distribution



Variance Inflation Factor (Multicollinearity) (~2.25)

Gemma

https://ai.google.dev/gemma/docs/base

Gemma (base)

Gemma 2 and Gemma are the core models of the Gemma family of open models. These generative artificial intelligence (AI) models are built from the same research and technology used to create the Gemini models.

Gemma base models are well-suited for a variety of text generation tasks, including question answering, summarization, reasoning, and can be further tuned for specific use cases. The base Gemma models are provided in a range of sizes, from 2 billion up to 27 billion parameters, and provide the following advantages:

Open models

Gemma models are provided with open weights and permit responsible commercial use, allowing you to tune and deploy them in your own projects and applications.

High performance

Gemma models provide state-of-the-art performance on AI tasks compared to other open models of similar size.

Wide framework support

Gemma is supported on a wide variety of tools and systems, including Keras 3.0, native PyTorch, JAX, and Hugging Face Transformers.

Pure vector (semantic) search

```
Query: What is signal boosting?

Results:
Score: 0.5567
8.1 Basic signals boosting  In section 4.2.2, we built our first signals
boosting model on the RetroTech dataset, enabling a significant boost in
relevance for the most frequently searched and clicked search results. In this
section, we'll quickly recap the process of creating a simple signals boosting
model, which we will build upon in the upcoming sections to cater to some more
advanced needs.  You'll recall from section 4.2.2 that signals boosting models
aggregate all useful activity signals on documents (such as click signals) that
occur as the result of a specific query. We used a search for ipad, and boosted
each document based on how many total times it was previously clicked in the
results for that search. Figure 8.1 demonstrates the before (no signal boosting)
and after (signals boosting on) search results for the query ipad.  Figure 8.1
Before and after applying a signals boosting model. Signals boosting improves
relevance by pushing the most popular items to the top of the search results.
Page number: 224
```

RAG: **Before (vanilla LLM)** and After

```
input_text = "What is signal boosting?"
input_ids = tokenizer(input_text, return_tensors="pt")

outputs = llm_model.generate(**input_ids, max_new_tokens=256)
#print(outputs)
print(tokenizer.decode(outputs[0]))
```

```
<bos>What is signal boosting?

Signal boosting is a technique used in telecommunications and wireless communication systems to amplify weak signals and improve their range.
It involves using a booster device, typically a parabolic dish antenna, to concentrate and amplify the signal before it is transmitted.

Here's a breakdown of how signal boosting works:

1. **Weak signal reception:** A base station or other wireless device receives a weak signal from a distant transmitter.
2. **Signal amplification:** The base station uses a booster to amplify the signal. This can be done by increasing the power of the signal or
by using a preamplifier to boost the signal's amplitude.
3. **Signal transmission:** The amplified signal is then transmitted through the wireless channel.
4. **Signal reception at the destination:** At the receiving end, the amplified signal is received by the base station or other device.

Signal boosting can be used to improve the range of a wireless signal by overcoming the natural attenuation of the signal as it travels throu
gh the air. It can also be used to improve the quality of the signal by reducing interference from other sources.

Here are some of the benefits of using signal boosting:

* **Increased range:** Signal boosting can extend the range of a wireless signal by overcoming the natural attenuation of the signal
```

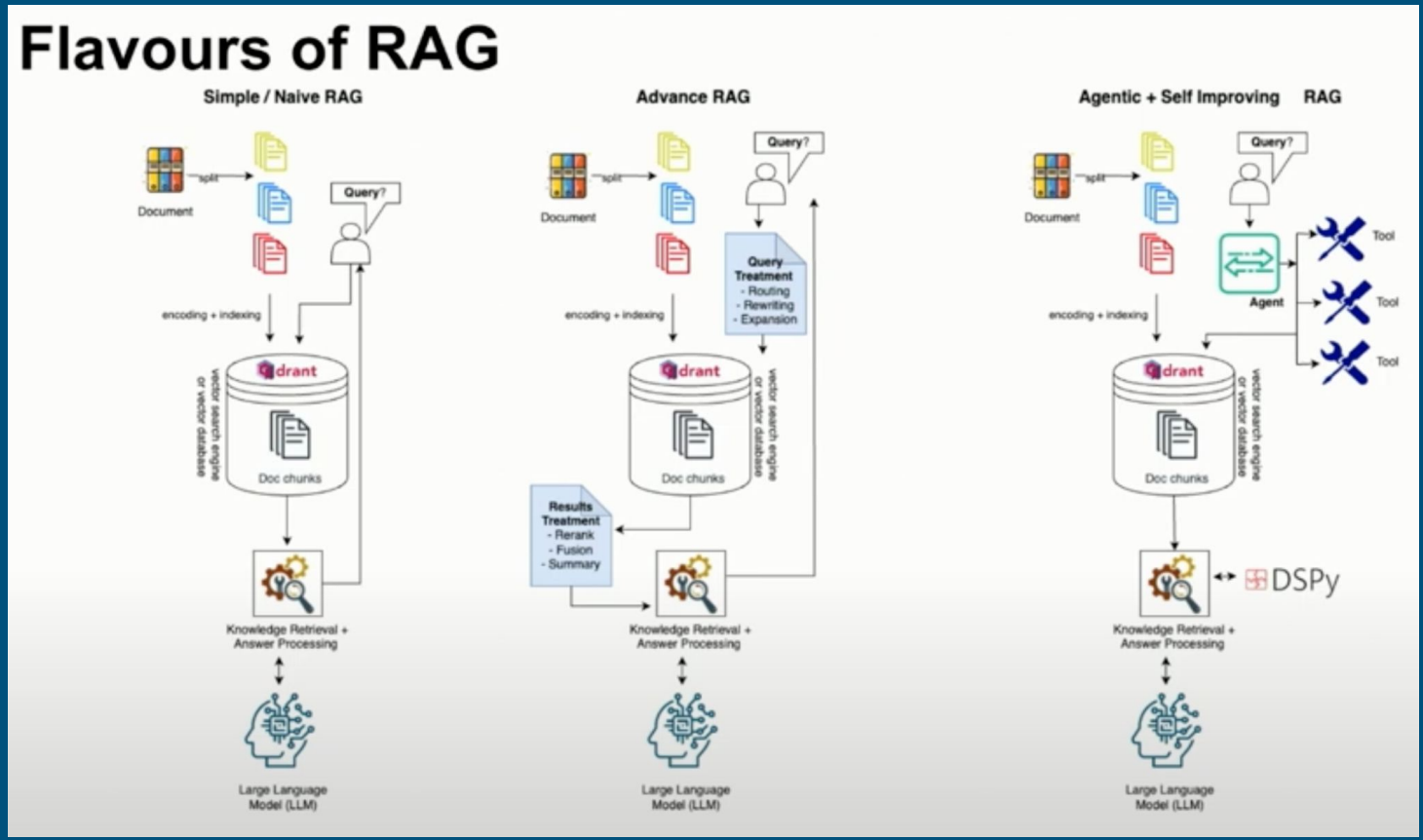
RAG: Before and **After (with RAG)**

```
query = "What is signal boosting?"
print(f"Query: {query}")

# Answer query with context and return context
answer, context_items = ask(query=query,
                            temperature=0.7,
                            max_new_tokens=512,
                            return_answer_only=False)

print(f"Answer:\n")
print_wrapped(answer)
print(f"Context items:")
context_items
```

```
Query: What is signal boosting?
[INFO] Time taken to get scores on 1080 embeddings: 0.00081 seconds.
Answer:

Sure, here's the answer to the user's query:  The passage describes how signal
boosting is a technique that aggregates and boosts signals (like click events)
on documents based on their relevance to a specific query. This helps to improve
the relevance of results in a search engine.
```

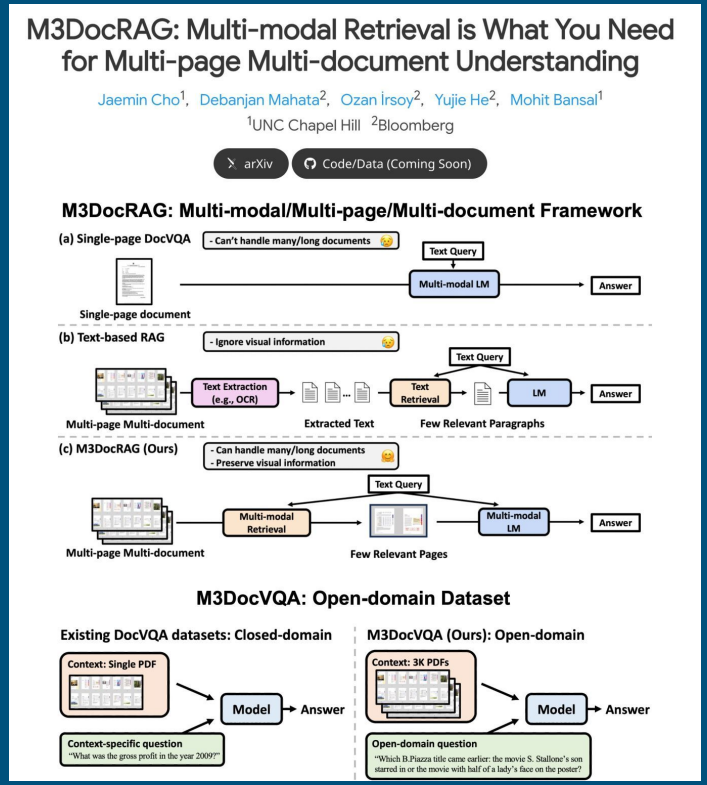

SHOW ME THE CODE!
imgflip.com

Flavours of RAG (2024)



https://2024.berlinbuzzwords.de/sessions/?id=YSJLBF

RAG Architectures



https://www.linkedin.com/feed/update/urn:li:activity:7260759982239895552/

How to pick relevant metrics?

Take an example of RAG built on Documentation

Quality of Answer
- **Answer Correctness**
  - **Query Fulfillment**
    - Completeness (**SelfCheckGPT**)
  - **Faithfulness** and **Groundedness**
    - **Context Utilization**
      - Derived from Document Chunks
        - **Context Sufficiency**
          - Quality of Retrieved Chunks - **Precision / Recall / nDCG**
- **Helpfulness**
- **Bias-Free**
- **Non-Malicious**
- Privacy Compliance
  - No Personal Information Shared (**PII**)
- Policy **Compliance**
- **Conciseness**
  - Designated **Number of Tokens** (Cost)
- **Latency** Requirements

Credit: Atita Aurora, talk from Berlin Buzzwords'24

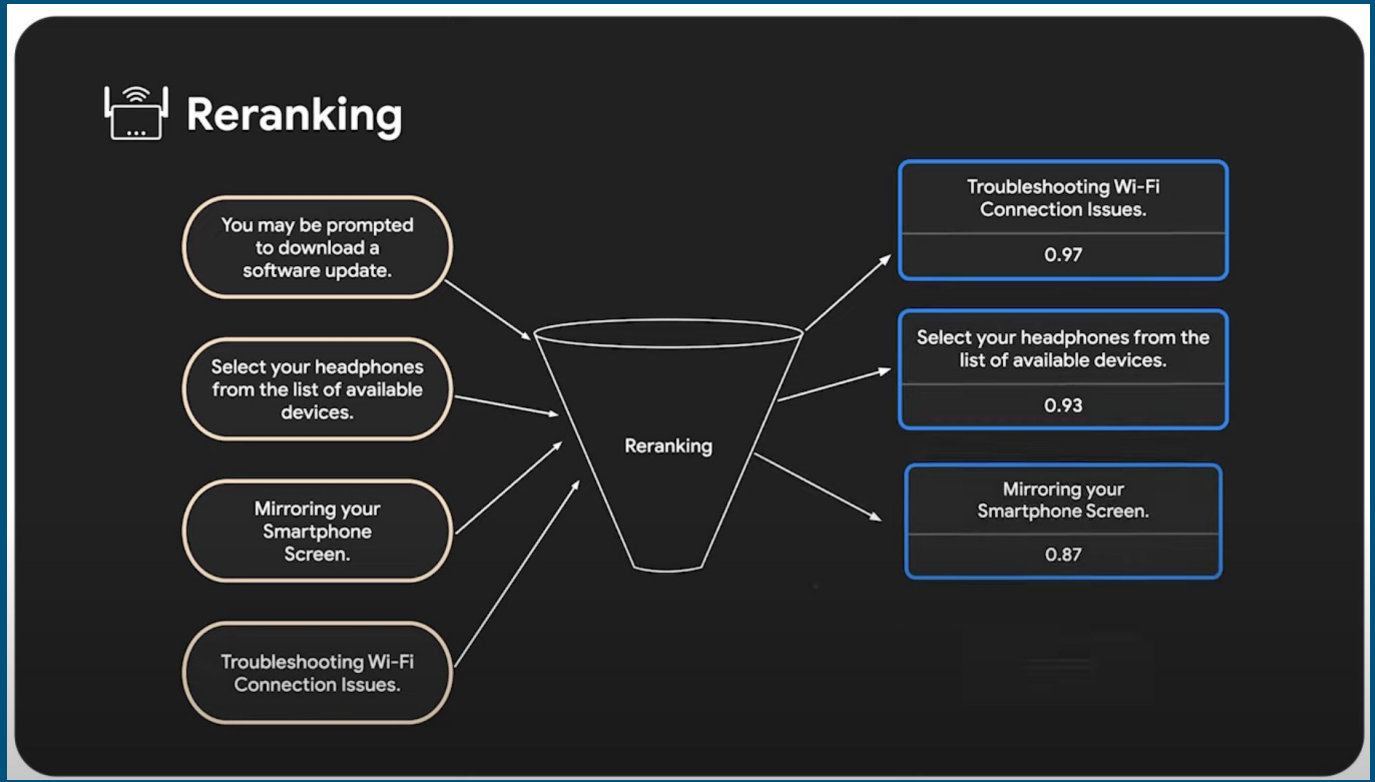
RAG Evaluation: tools

Credit: Atita Aurora, talk from Berlin Buzzwords'24

Ways to improve RAG

1. Change chunking algorithm
2. Augment each chunk with the list of questions it may answer (GAR)
3. Add metadata, like country, time, price, topic etc and use these as pre-filters before / during dense retrieval
4. Reranking: several sources of answers blended in a single list using a reranker model (e.g.: ColBert or DPR)
5. Change LLM (check studies by unsloth in Further study)

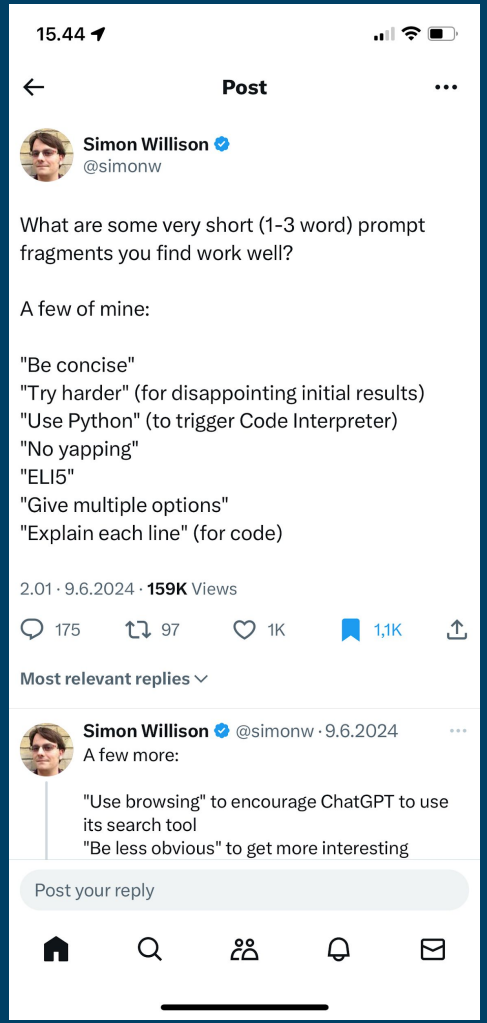
Reranking



https://www.youtube.com/watch?v=sGvXO7CVwc0

Working with LLMs:

- Science
- Great deal of engineering
- Black magic ->



Lab!

Two groups of tasks:

- Basic / intermediate level (for learners of RAG) - don’t pick if you know RAG and see no value in this
- Tasks with an asterisk (for students that know RAG) - pick if you know RAG

Lab: basic / intermediate

Task 1: Change the notebook or streamlit UI to support pdf documents in a language other than English: Finnish, Swedish, German etc. Things to consider:

- Would the same embedding and LLM work for Finnish?
- What about extracting sentences and chunking: is there any change in terms of token length / chunk size?
- Can you assess the final quality?

Task 2: Research and implement alternative algorithm for chunking. For example, you can take a look at semantic chunking technique. Things to consider:

- Does this chunker apply to any language?
- Can you assess the quality of chunker on a handful of pages in your pdf document?
- What is the impact on quality of the overall RAG system pipeline?

Lab: advanced stuff

Task 3(*):

- Research agentic RAG. Pick a task, like checking stock price of a company, detect the respectful intent (“What is the stock price for Nvidia?”) and pull the price.
- You can also come up with your own task / tool to use and implement that instead.

Task 4(**):

- Research GraphRAG: https://www.youtube.com/watch?v=knDDGYHnnSI
- Take a look at Neo4j demo: https://neo4j.com/labs/genai-ecosystem/rag-demo/
- Build a KG for your domain of choice (it can be financial documents or research papers from arxiv) and demonstrate its power with RAG

Further study

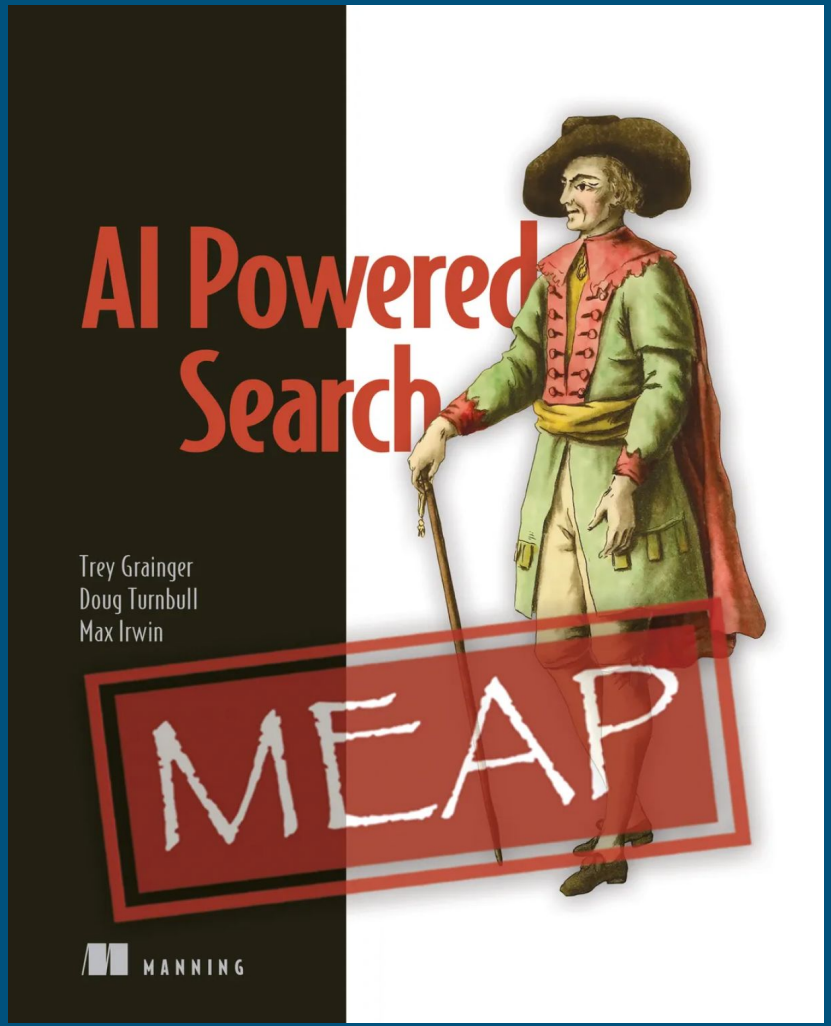
- Simple local RAG: https://github.com/mrdbourke/simple-local-rag
- Podcast with Patrick Lewis on RAG and LLMs: Apple Podcasts, Spotify
- Prompt engineering: https://docs.cohere.com/v2/docs/crafting-effective-prompts https://docs.cohere.com/v2/docs/advanced-prompt-engineering-techniques
- Unsloth: low-level technicals of LLMs: https://www.youtube.com/watch?v=pRM_P6UfdIc
- Gemma: https://ai.google.dev/gemma
- Is Semantic Chunking Worth the Computational Cost? https://arxiv.org/pdf/2410.13070
- Nano GraphRAG https://github.com/gusye1234/nano-graphrag
- RAG evaluation: https://www.youtube.com/watch?v=swI4SLDTFH0 and https://www.youtube.com/watch?v=DId2KP8Ykz4
- RAG Evaluation: notebooks by Qdrant's team: https://github.com/qdrant/qdrant-rag-eval/tree/master

Papers

1. Retrieval-Augmented Generation for Large Language Models: A Survey: https://arxiv.org/abs/2312.10997 (2024)
2. Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG: https://arxiv.org/abs/2501.09136 (2025)

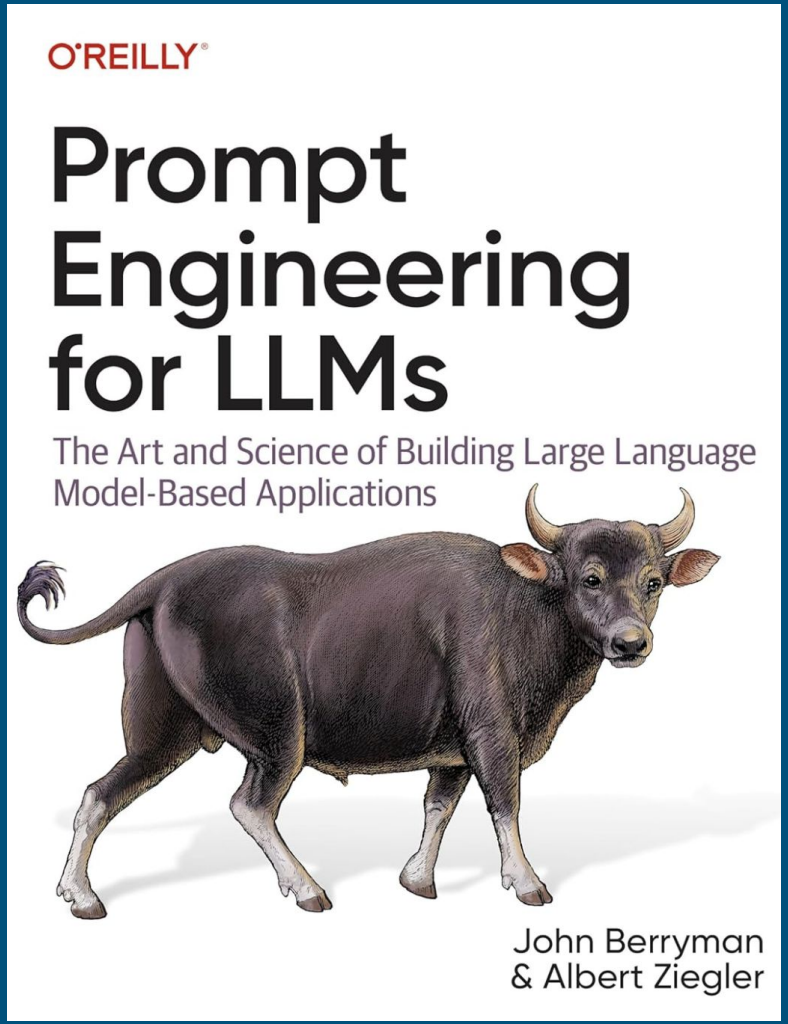
Chunking

https://github.com/FullStackRetrieval-com/RetrievalTutorials/blob/main/tutorials/LevelsOfTextSplitting/5_Levels_Of_Text_Splitting.ipynb



Authors:

- Trey (Presearch, Lucidworks, CareerBuilder.com), lead author
- Doug Turnbull (OpenSource Connections, Shopify, Reddit) - expert in Learning To Rank
- Max Irwin (MAX.IO, OpenSource Connections, Wolters Kluwer)



Authors:

- John Berryman - ex-GitHub Copilot, author of “Relevant Search” (co-authored with Doug Turnbull)
- Albert Zieger - ex-GitHub Copilot (co-creator)