RAG: Retrieval Augmented Generation - Lecture 5

Dmitry Kan, PhD
Senior Product Manager, Search (TomTom)
Host of Vector Podcast

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

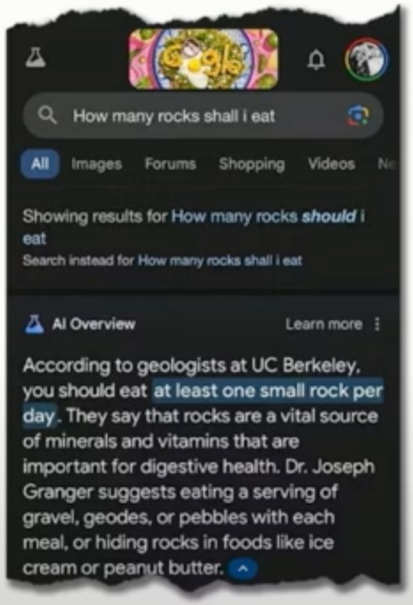
LLMs: strengths

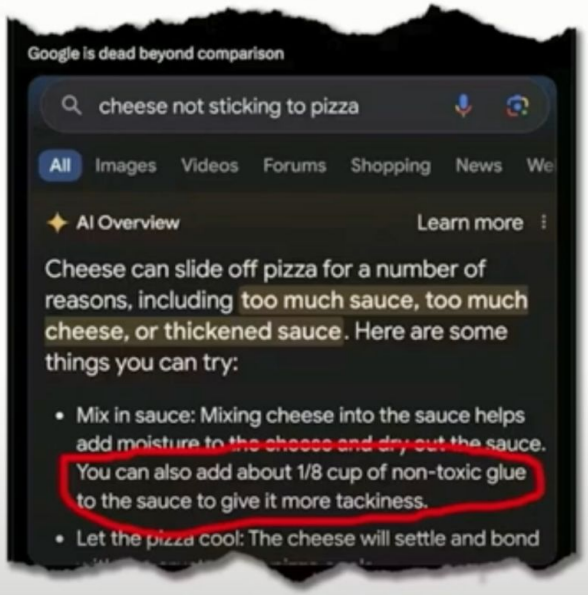
Let’s fill in this slide together!

LLMs: weaknesses

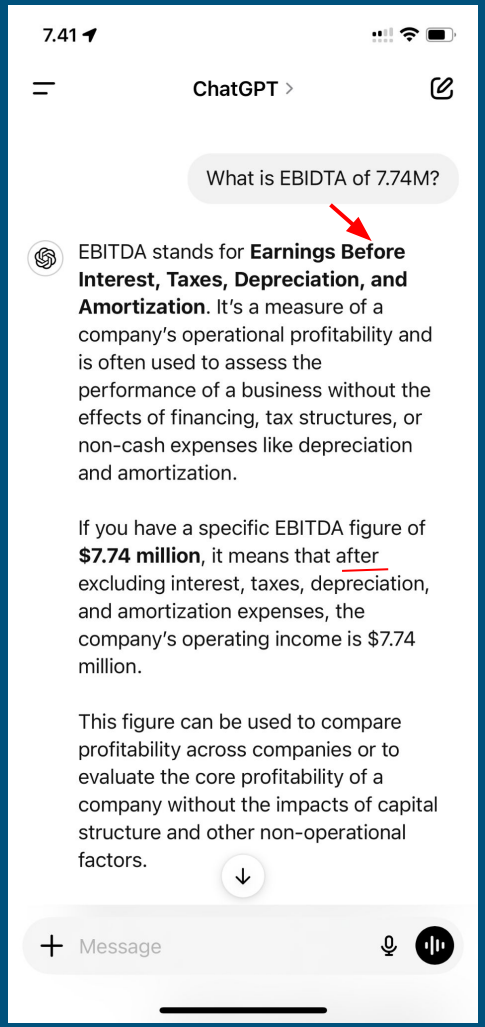
Let’s fill in this slide together!

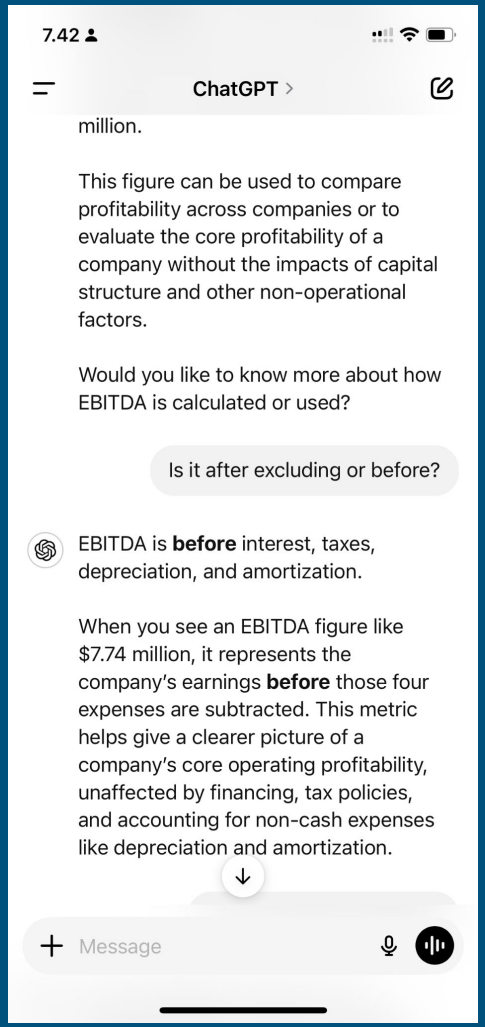
Example of a hallucination

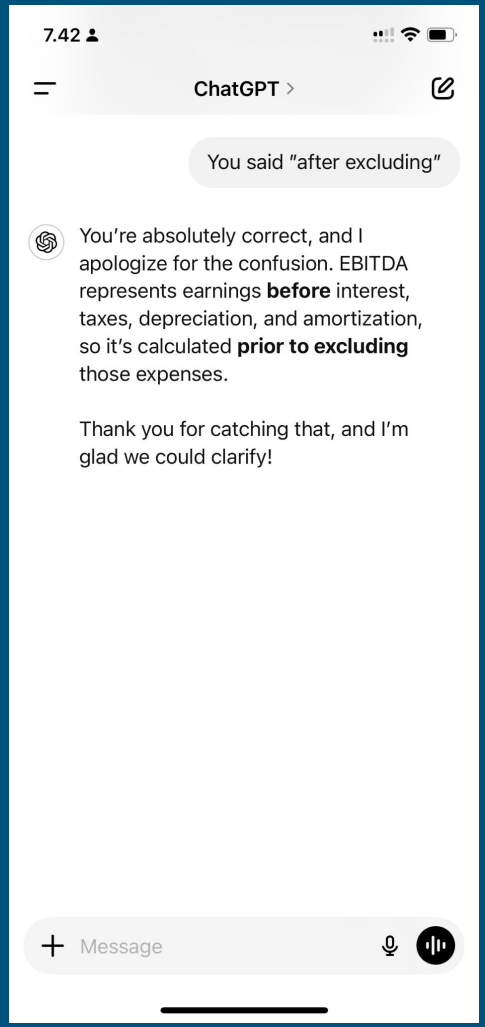




Ref :
https://theconversation.com/eat-a-rock-a-day-put-glue-on-your-pizza-how-googles-ai-is-losing-touch-with-reality-230953







Before or after?

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 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

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

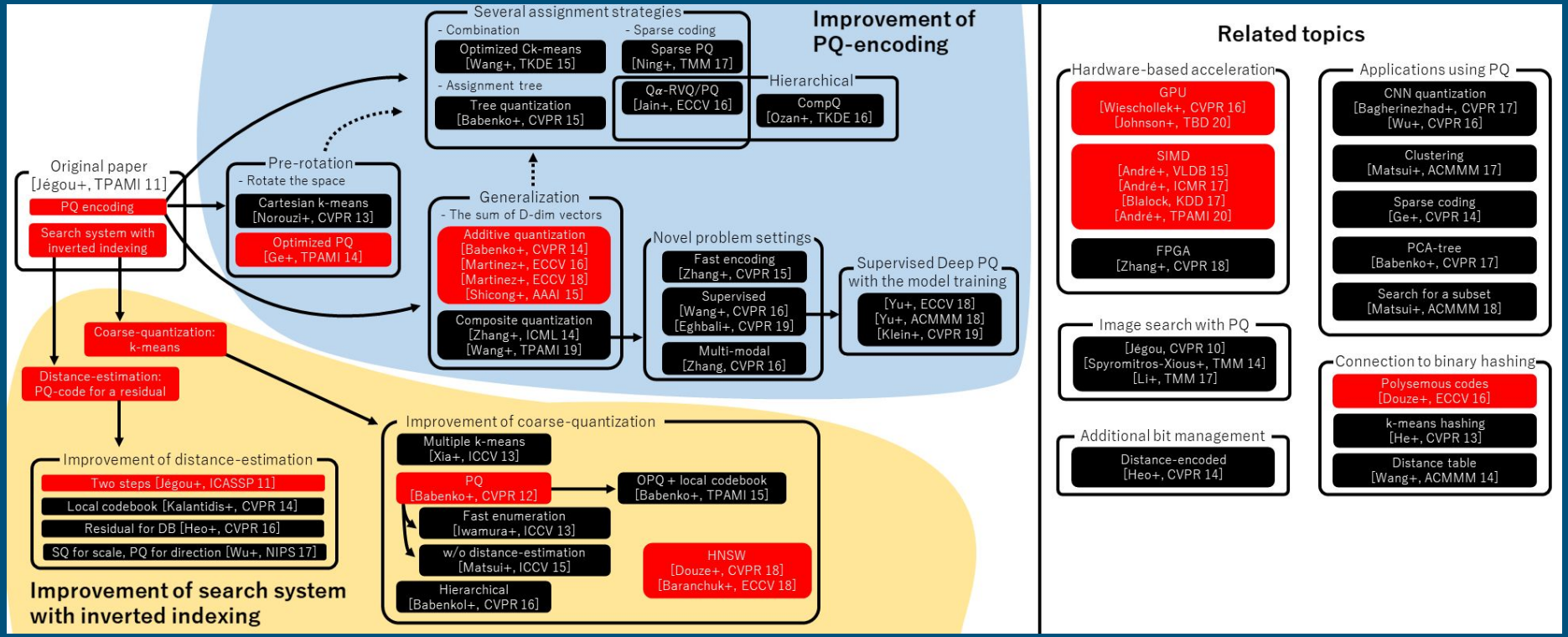
Gemma models provide state-of-the-art performance on AI tasks compared to other open models of similar size.

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.

ANN algorithms

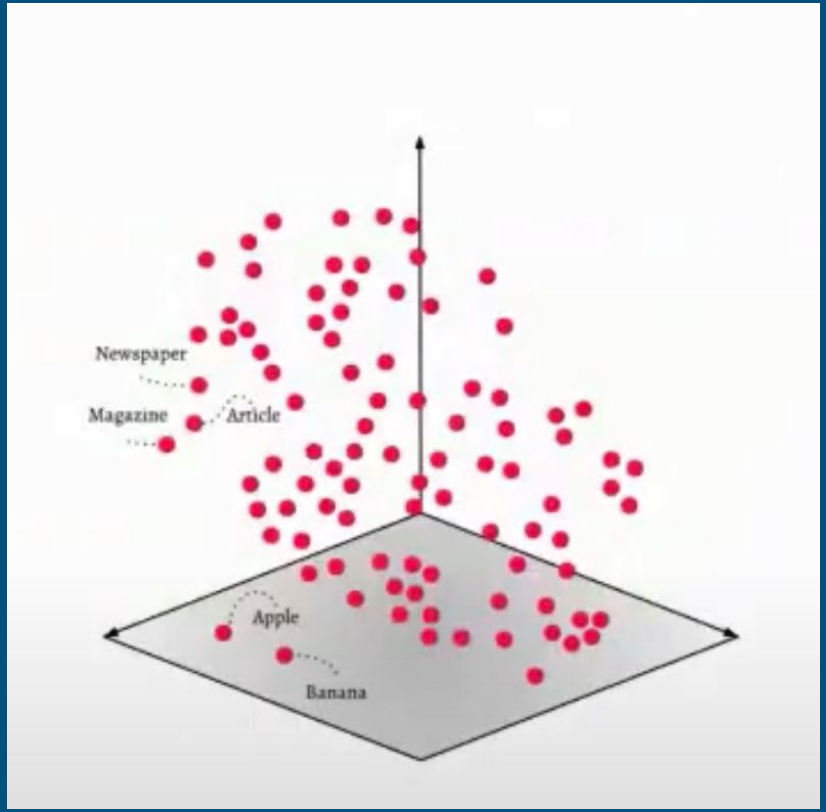
Source: FAISS



Vector search in a nutshell

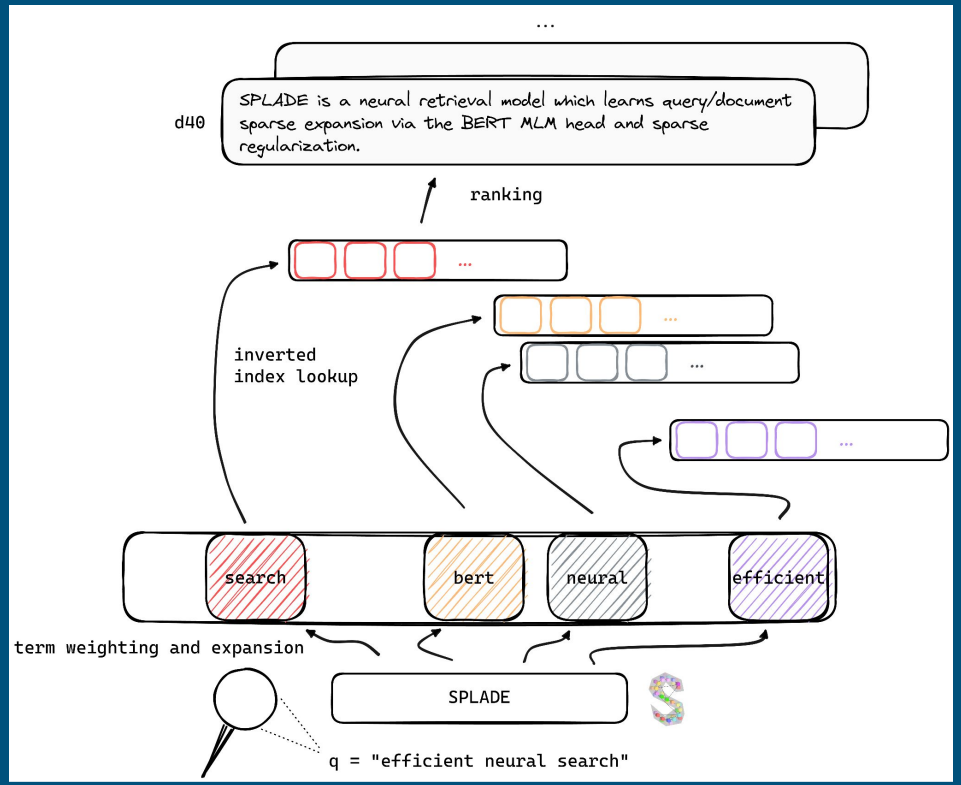
Vector search is a way to represent and search your objects (documents, songs, images..) in a geometric space (usually of high-dimension) in the form of an embedding (a vector of numbers: [0.9, -0.1, 0.15, …])

- 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

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
- Microsoft (Bing team): Zoom, DiskANN, SPTAG
  - Azure Cognitive Search
- Amazon: KNN based on HNSW in OpenSearch
- Google: ScaNN
- Yahoo! Japan: NGT
- Facebook: FAISS, PQ (CPU & GPU)
- Baidu: IPDG (Baidu Cloud)
- Yandex
- NVidia
- Intell

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
```

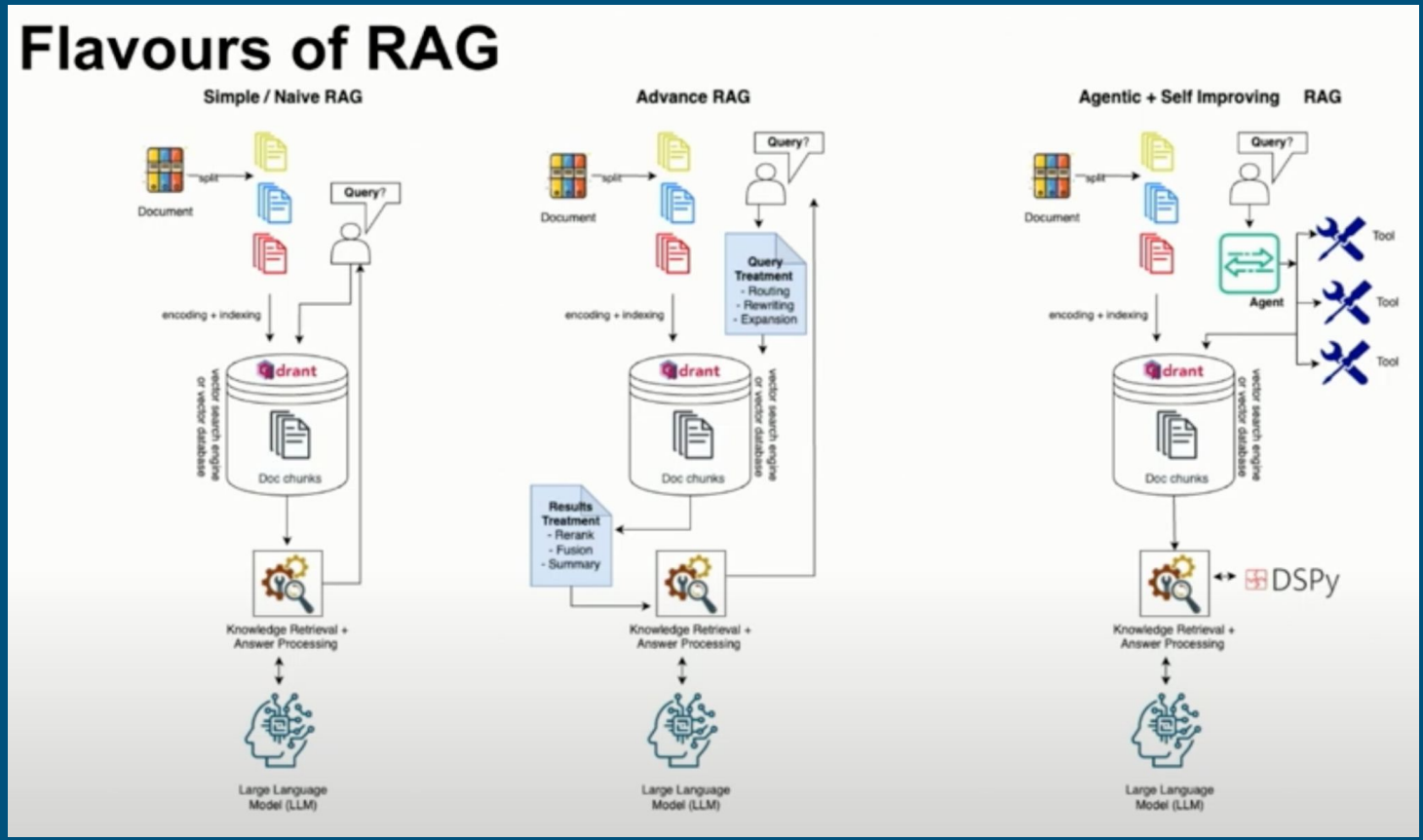
RAG: Before and **After (with RAG)**

```
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.
```

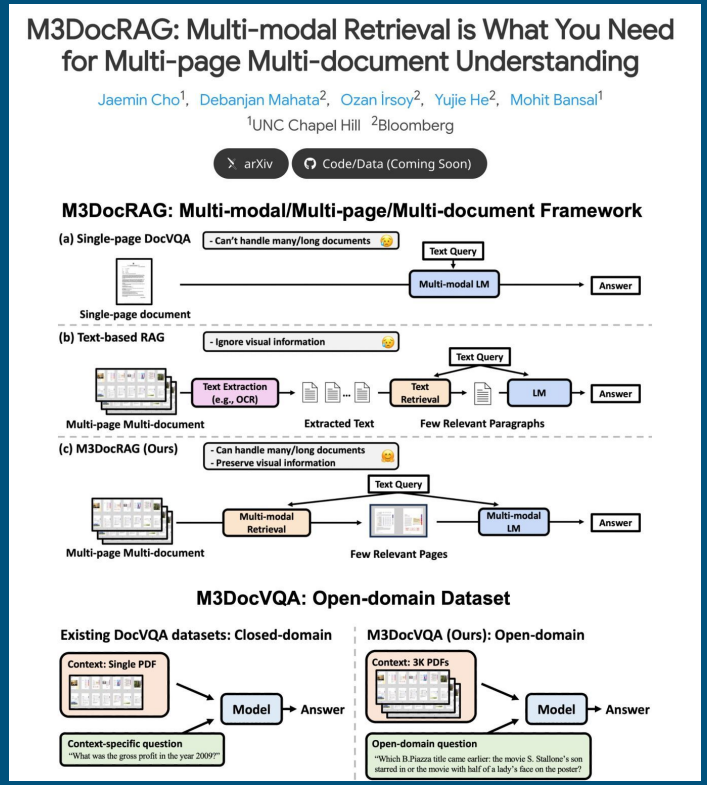

SHOW ME THE CODE!
imgflip.com

Flavours of RAG



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

RAG Architectures



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

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

- Podcast with Patrick Lewis on RAG and LLMs: Apple Podcasts, Spotify
- 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
- https://github.com/gusye1234/nano-graphrag
- RAG evaluation: https://www.youtube.com/watch?v=swI4SLDTFH0 and https://www.youtube.com/watch?v=DId2KP8Ykz4