RAGing against the machine

A brief introduction to Retrieval Augmented Generation

Adam Ek (adam.ek@ai.se)
AI Sweden
June 12, 2025

Who am I?

- Ph.D. in Computational Linguistics from Gothenburg University.
- Leading the AI development on the SVEA project

SVEA

SWEDEN

- SVEA is a digital assistant for the public sector

SVEA

- SVEA is a digital assistant for the public sector
- The goal: Allow workers in the public sector to reliably search and converse with

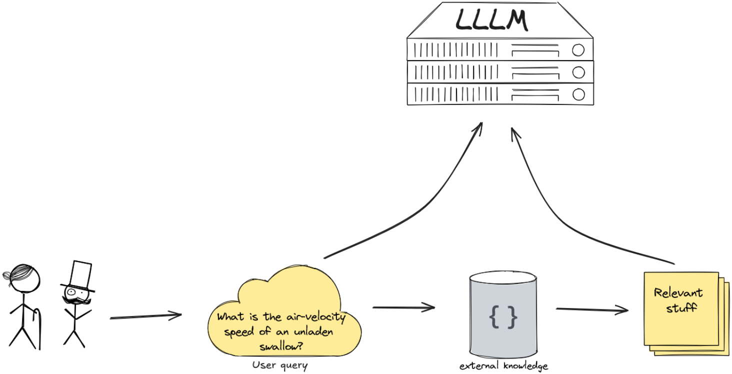
SVEA

- SVEA is a digital assistant for the public sector
- The goal: Allow workers in the public sector to reliably search and converse with
  - Internal databases (local guidelines and routines, HR, …)
  - Public databases (national guidelines, laws and regulations, …)

SVEA

- SVEA is a digital assistant for the public sector
- The goal: Allow workers in the public sector to reliably search and converse with
  - Internal databases (local guidelines and routines, HR, …)
  - Public databases (national guidelines, laws and regulations, …)
- The how: Retrieval Augmented Generation (LLM + database)

Retrieval Augmented Generation (RAG)



Why not just a LLM?

- The LLM has a memory, internal knowledge, and a typically large context window

Why not just a LLM?

AI SWEDEN

- The LLM has a memory, internal knowledge, and a typically large context window
  1. The relevant knowledge may be **hard** to locate
  2. The LLM decides what sources to use
  3. The knowledge is a snapshot

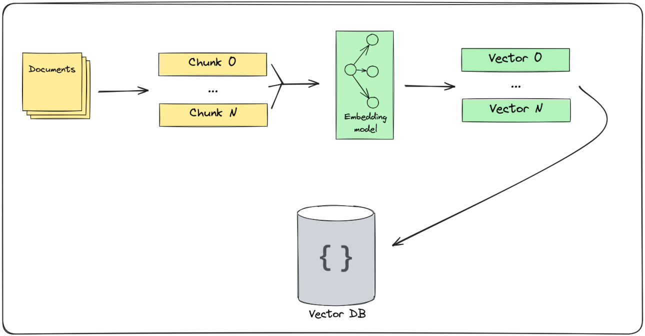
Why not just a LLM?

- The LLM has a memory, internal knowledge, and a typically large context window
  1. The relevant knowledge may be **hard** to locate
  2. The LLM decides what sources to use
  3. The knowledge is a snapshot
- Why a vector database?

Why not just a LLM?

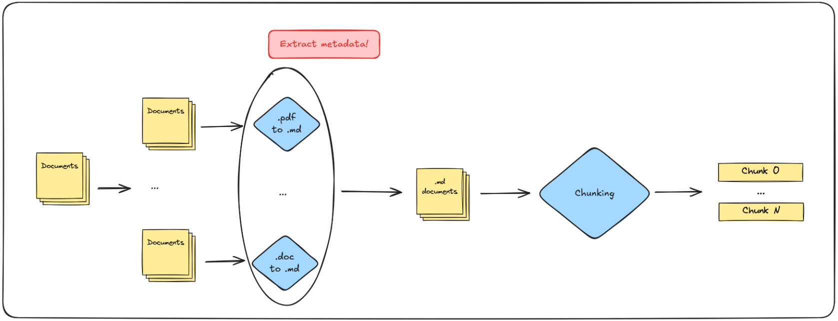
- The LLM has a memory, internal knowledge, and a typically large context window
  1. The relevant knowledge may be **hard** to locate
  2. The LLM decides what sources to use
  3. The knowledge is a snapshot
- Why a vector database?
  1. We can construct specialized retrievers for our data
  2. We can filter the data based on meta-knowledge of the data
  3. We can **update** the knowledge database
  4. It's fast!

Creating a vector database



Garbage in, garbage out

Preparing documents



Preprocessing documents

- Goal of the system: find pieces of information **in documents**

Preprocessing documents

- Goal of the system: find pieces of information **in documents**
- Filter the data so the relevant information is **easy** to encode
  - Remove **bös**, clean/fix parsing errors, manage multi-modal information, …

Chunking strategies

- So we split documents into managable ”chunks”

Chunking strategies

- So we split documents into managable ”chunks”
  - Small chunks: Chunks may lack sufficient context to be useful.
  - Large chunks: Chunks may contain more noise, but also more context.

Chunking strategies

- So we split documents into managable ”chunks”
  - Small chunks: Chunks may lack sufficient context to be useful.
  - Large chunks: Chunks may contain more noise, but also more context.
- Fixed-size chunking: Take fixed-size chunks of N consecutive tokens

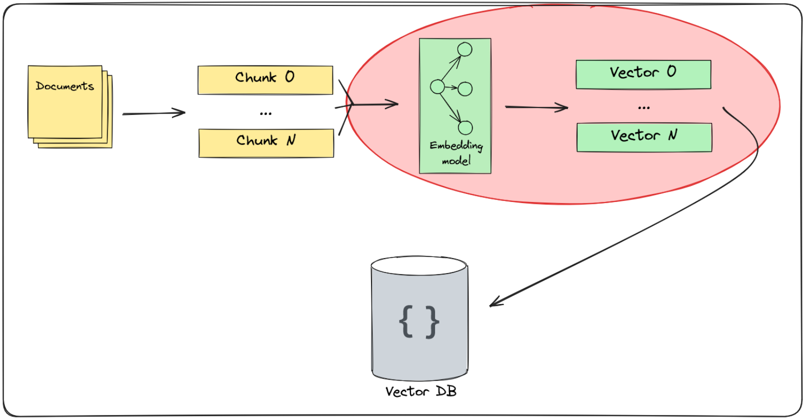
Chunking strategies

- So we split documents into managable ”chunks”
  - Small chunks: Chunks may lack sufficient context to be useful.
  - Large chunks: Chunks may contain more noise, but also more context.
- Fixed-size chunking: Take fixed-size chunks of N consecutive tokens
- Semantic chunking: Identify semantic coherent structures

Chunking strategies

- So we split documents into managable ”chunks”
  - Small chunks: Chunks may lack sufficient context to be useful.
  - Large chunks: Chunks may contain more noise, but also more context.
- Fixed-size chunking: Take fixed-size chunks of N consecutive tokens
- Semantic chunking: Identify semantic coherent structures
- Hierarchical chunking: Identify a tree structure based on e.g. headers, paragraph-breaks

Embedding chunks



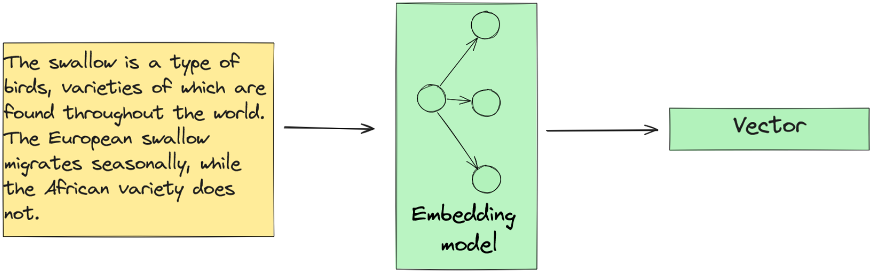
Embedding chunks

- We use an embedding model to extract embeddings

Embedding chunks

- We use an embedding model to extract embeddings
  - A model specifically designed to **encode** information
  - Much smaller than LLLMs (~ 100-600 million parameters)

Embedding model



Embedding chunks

- We can use an **arbitrary** embedding model

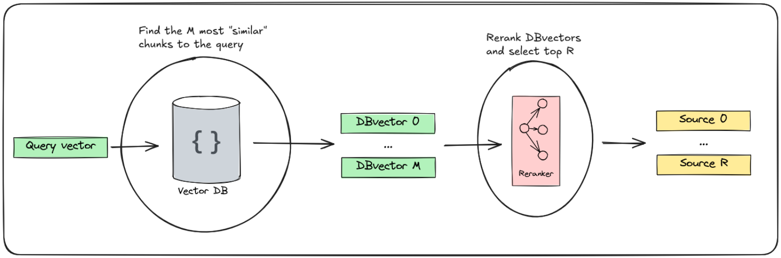
Embedding chunks

- We can use an **arbitrary** embedding model
- We have full control of the level of specialization

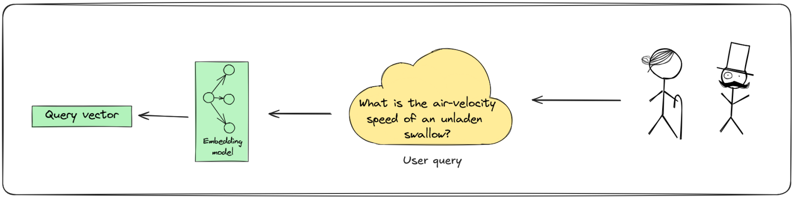
Embedding chunks

- We can use an **arbitrary** embedding model
- We have full control of the level of specialization
- Optimize the trade-off between speed (smaller models) and efficiency (larger models)

Obtaining results from a vector database

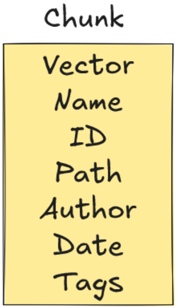


Question embedding



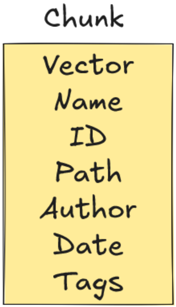
Indexing

- A vector database contains chunks, and their fields:



Indexing

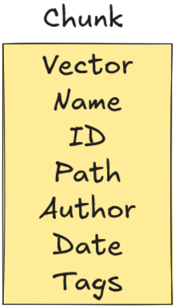
- A vector database contains chunks, and their fields:
- What do we search for?



Indexing

- A vector database contains chunks, and their fields:
- What do we search for?

  Semantic and contextual similarity of a chunk to the query!



Finding chunks

- We search in our vector datadase for the top M most similar chunks

Finding chunks

- We search in our vector datadase for the top M most similar chunks
- What similarity!? Cosine, Euclidean, or any other vector distance metric!

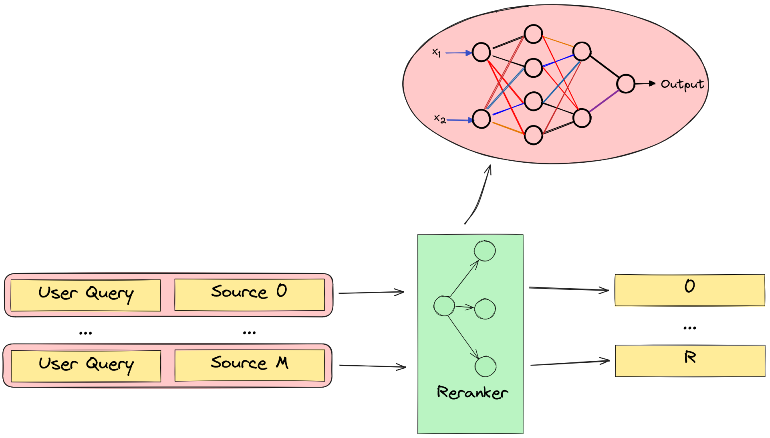
Finding chunks

- We search in our vector datadase for the top M most similar chunks
- What similarity!? Cosine, Euclidean, or any other vector distance metric!
- Reranking
  - The initial search results are fast, but fuzzy

Finding chunks

- We search in our vector datadase for the top M most similar chunks
- What similarity!? Cosine, Euclidean, or any other vector distance metric!
- Reranking
  - The initial search results are fast, but fuzzy
  - The LLM is sensitive to the order of chunks

Reranking



x_1
x_2
Output
User Query
Source 0
...
...
User Query
Source M
Reranker
0
...
R

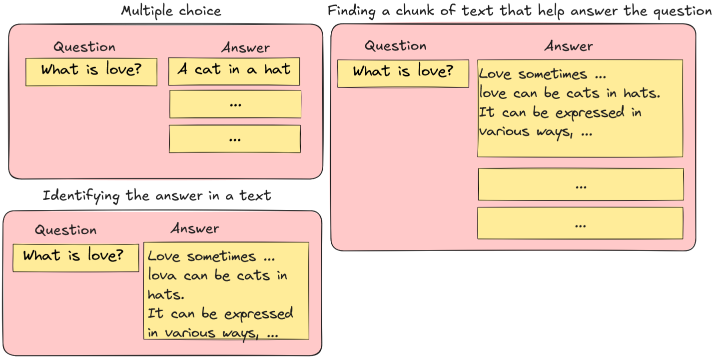
Retrieval mantra

- Embedding model should maximize fuzzy search
  - The search should contain chunks with the same topic, domain, …

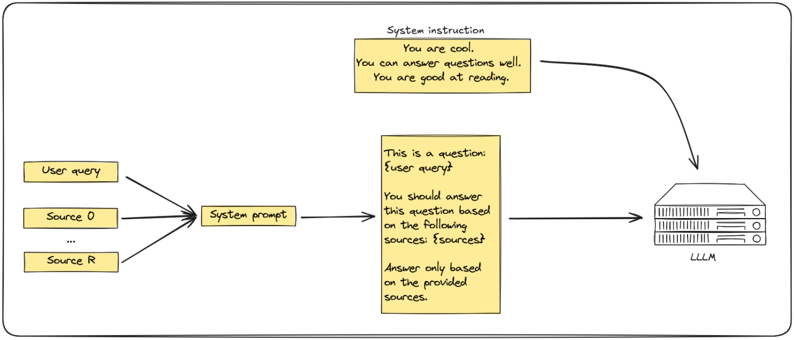
Retrieval mantra

- Embedding model should maximize fuzzy search
  - The search should contain chunks with the same topic, domain, …
- Reranking should maximize contextual fit
  - How well does the chunk answer the question?

Adapting embeddings models to data



Generating a response



Some resources

- Presentation/demo repo: `https://github.com/adamlek/intro-to-RAG`
- Embedding models: `https://cohere.com/blog/embedding-models`
- LLamaIndex: `https://www.llamaindex.ai/`
- Text cleaning: `https://www.analyticsvidhya.com/blog/2022/01/text-cleaning-methods-in-nlp/`
- Rag Evaluation: `https://cloud.google.com/blog/products/ai-machine-learning/optimizing-rag-retrieval`, `https://qdrant.tech/blog/rag-evaluation-guide/`