Generative AI:
Real-World Chat mit Unternehmensdaten: Advanced RAG jenseits von “Hallo Welt”!

thinktecture

DWX DEVELOPER WEEK '24

Sebastian Gingter
sebastian.gingter@thinktecture.com
Developer Consultant

Überschneidung
mit dem Montag-Workshop

Generative AI

Real-World Chat mit Unternehmensdaten: Advanced RAG jenseits von “Hallo Welt”!

- Was Sie **ERWARTET**
  - Hintergrundwissen und Theorie zu RAG
  - Überblick über **Semantische Suche**
  - **Probleme** die auftreten können
  - Pragmatische Methoden für die Verwendung **eigener Daten** im RAG
  - Demos (**Python**)
- Was Sie **NICHT** erwartet
  - ChatGPT, CoPilot(s)
  - Grundlagen von ML
  - Deep Dives in LLMs, Vektor-Datenbanken, LangChain

Sebastian Gingter

Developer Consultant @ Thinktecture AG

- Generative AI in business settings
- Flexible and scalable backends
- All things .NET

- Pragmatic end-to-end architectures
- Developer productivity
- Software quality

sebastian.gingter@thinktecture.com @phoenixhawk https://www.thinktecture.com

Was Euch erwartet (und was nicht):

- Ein bisschen Hintergrund-Info & Theorie
- Überblick über das Themengebiet Semantische Suche
- Probleme und mögliche Strategien
- Pragmatische Ansätze für die eigenen Daten
- Kein C#, sondern Python 😱

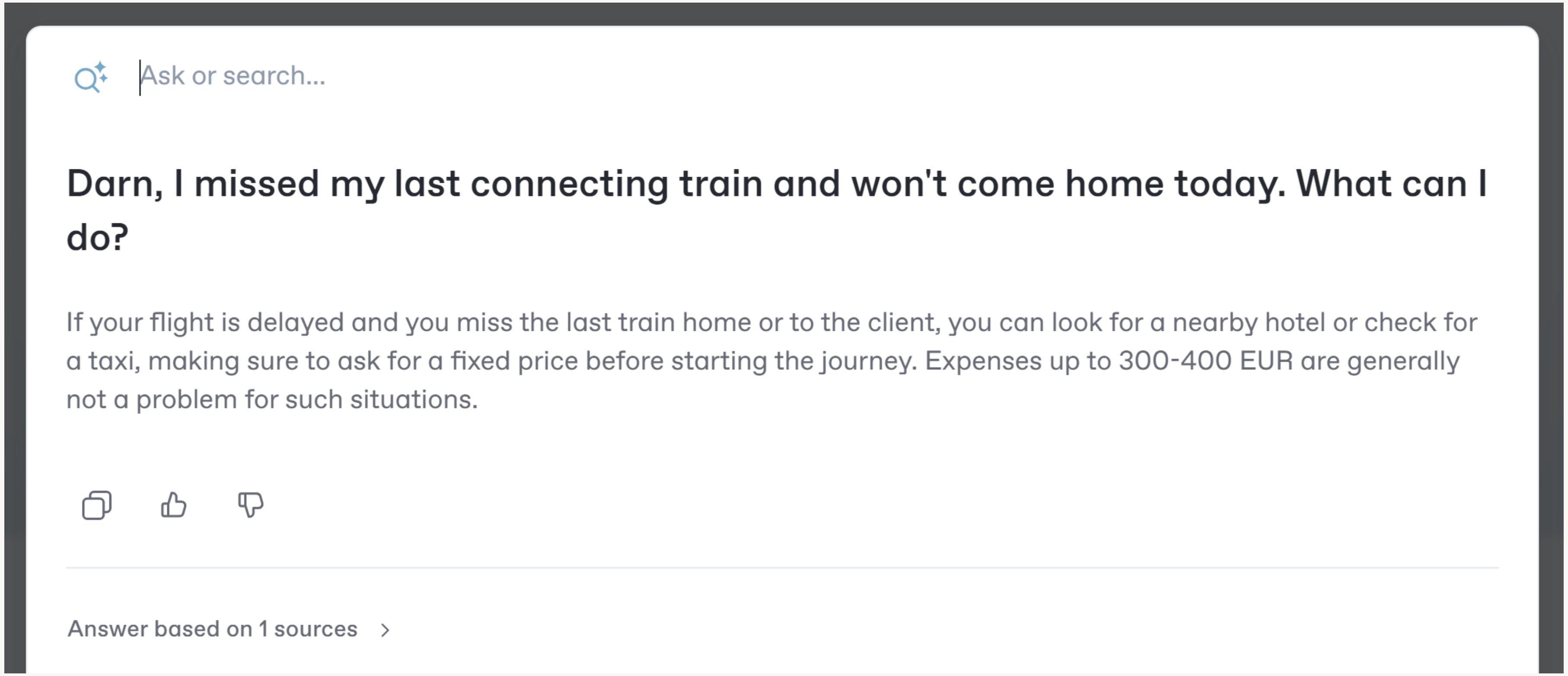
- Kein Deep-Dive in
  - LLMs
  - Vektor-Datenbanken
  - LangChain

Agenda

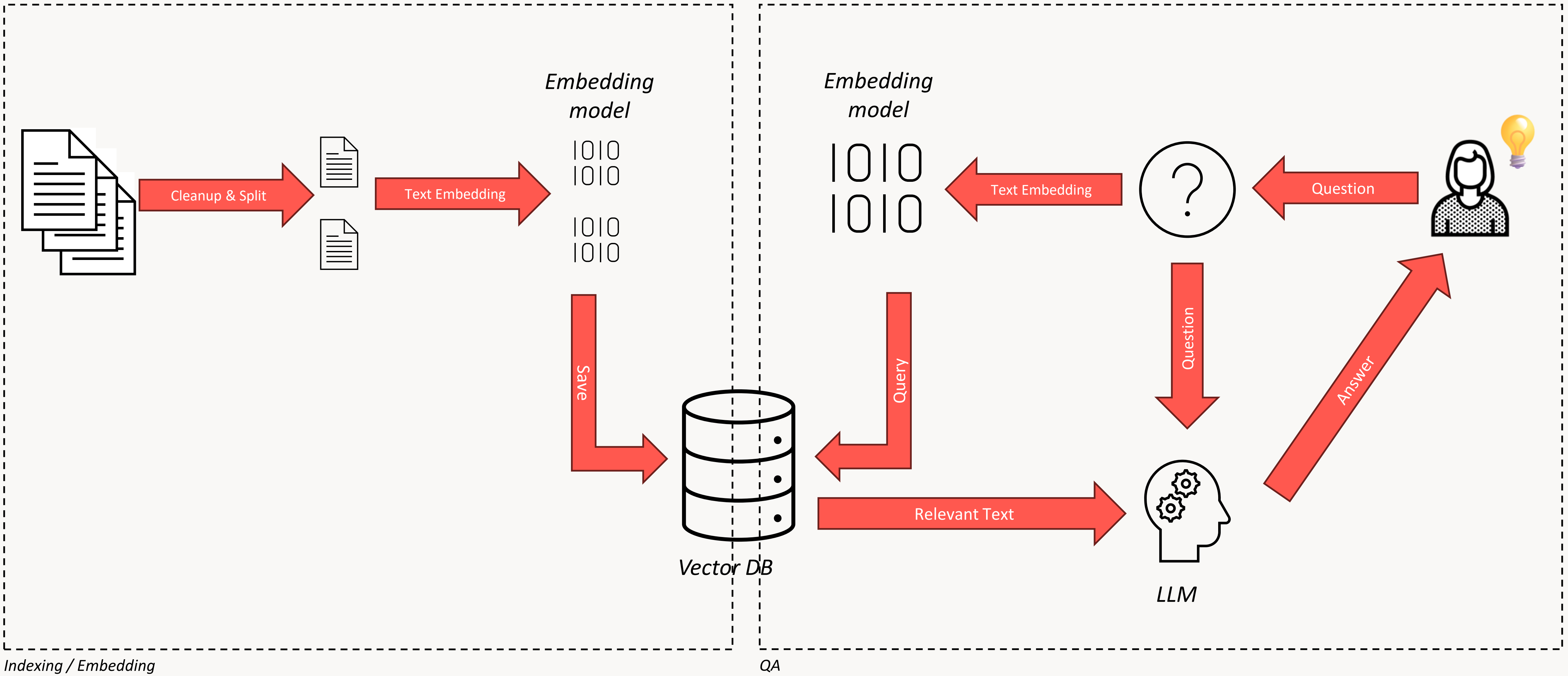
- Short Introduction to RAG
- Embeddings (and a bit of theory 😱)
- Vector-Databases
- Indexing
- Retrieval
- Not good enough? – Indexing II
    - HyDE & alternative indexing methods
- Conclusion

Introduction

Use-case: Talk to my internal data



Retrieval-augmented generation (RAG)
Indexing & (Semantic) search



Semantic Search

- Classic search: lexical
  - Compares words, parts of words and variants
  - Classic SQL: WHERE ‘content’ LIKE ‘%searchterm%’
  - We can search only for things where we know that its somewhere in the text
- New: Semantic search
  - Compares for the same contextual meaning
    - “Das Rudel rollt das runde Gerät auf dem Rasen herum”
    - “The pack enjoys rolling a round thing on the green grass”
    - “Die Hunde spielen auf der Wiese mit dem Ball”
    - “The dogs play with the ball on the meadow”

Semantic Search

- How to grasp “semantics”?
- Computers only calculate on numbers
    - Computing is “applied mathematics”
- AI also only calculates on numbers
- We need a numeric representation of meaning

    ➔ “Embeddings”

Embeddings

Introduction | Embeddings | Vector-DBs | Indexing | Retrieval | Indexing II | RAG

Embedding (math.)

- Topologic: Value of a high dimensional space is “embedded” into a lower dimensional space
- Natural / human language is very complex (high dimensional)
  - Task: Map high complexity to lower complexity / dimensions
- Injective function
- Similar to hash, or a lossy compression

Embeddings

- Embedding model (specialized ML model) converting text into a numeric representation of its meaning

- Representation is a vector in an n-dimensional space
  - n floating point values
  - OpenAI
    - “text-embedding-ada-002” uses 1536 dimensions
    - “text-embedding-3-small” 512 and 1536
    - “text-embedding-3-large” 256, 1024 and 3072
  - Huggingface 🤗 models have a very wide range of dimensions

https://huggingface.co/spaces/mteb/leaderboard & https://openai.com/blog/new-embedding-models-and-api-updates

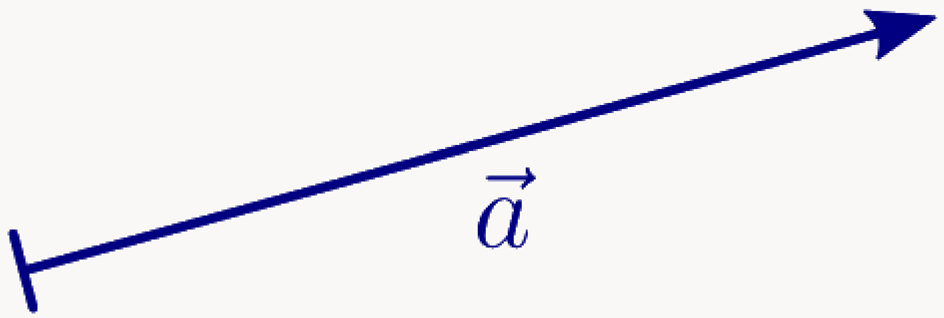
Embeddings

- Embedding models are unique
- Each dimension has a different meaning, individual to the model
- Vectors from different models are incompatible with each other
- Some embedding models are multi-language, but not all
- In an LLM, also the first step is to embed the input into a lower dimensional space

What is a vector?

- Mathematical quantity with a direction and length

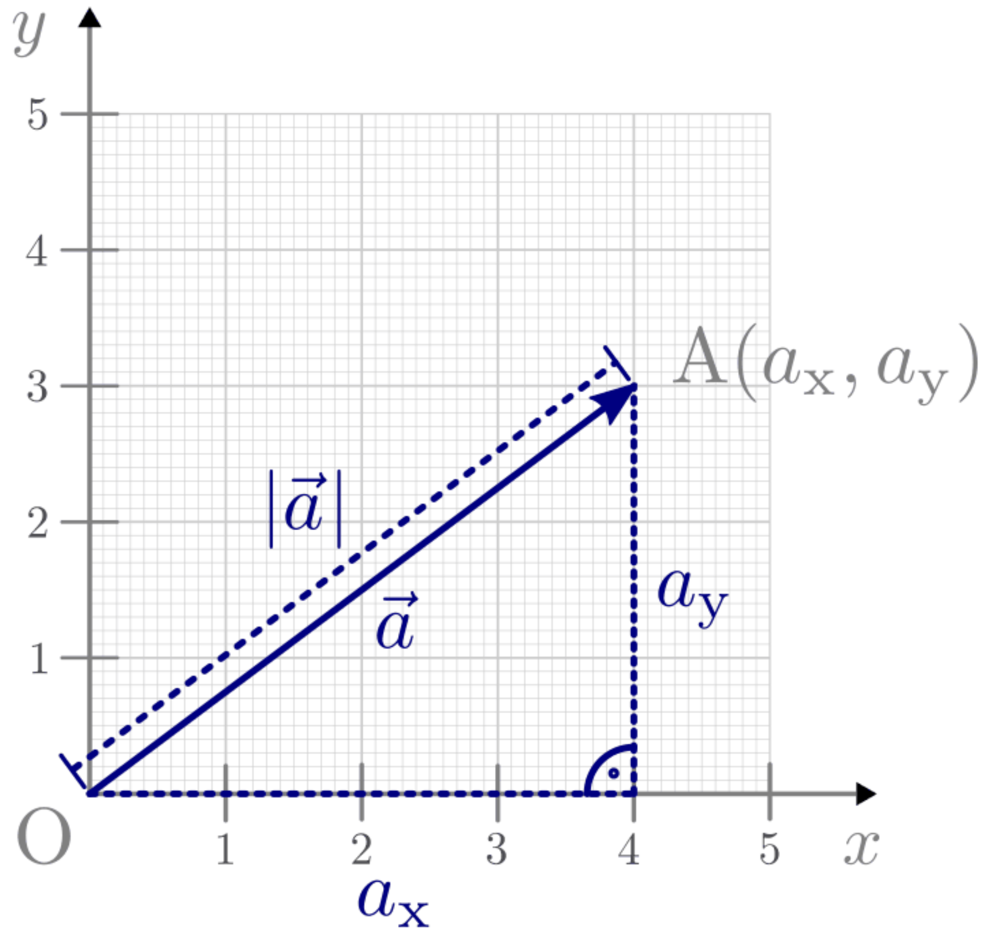
- $\vec{a} = \begin{pmatrix} a_x \\ a_y \end{pmatrix}$



https://mathinsight.org/vector_introduction

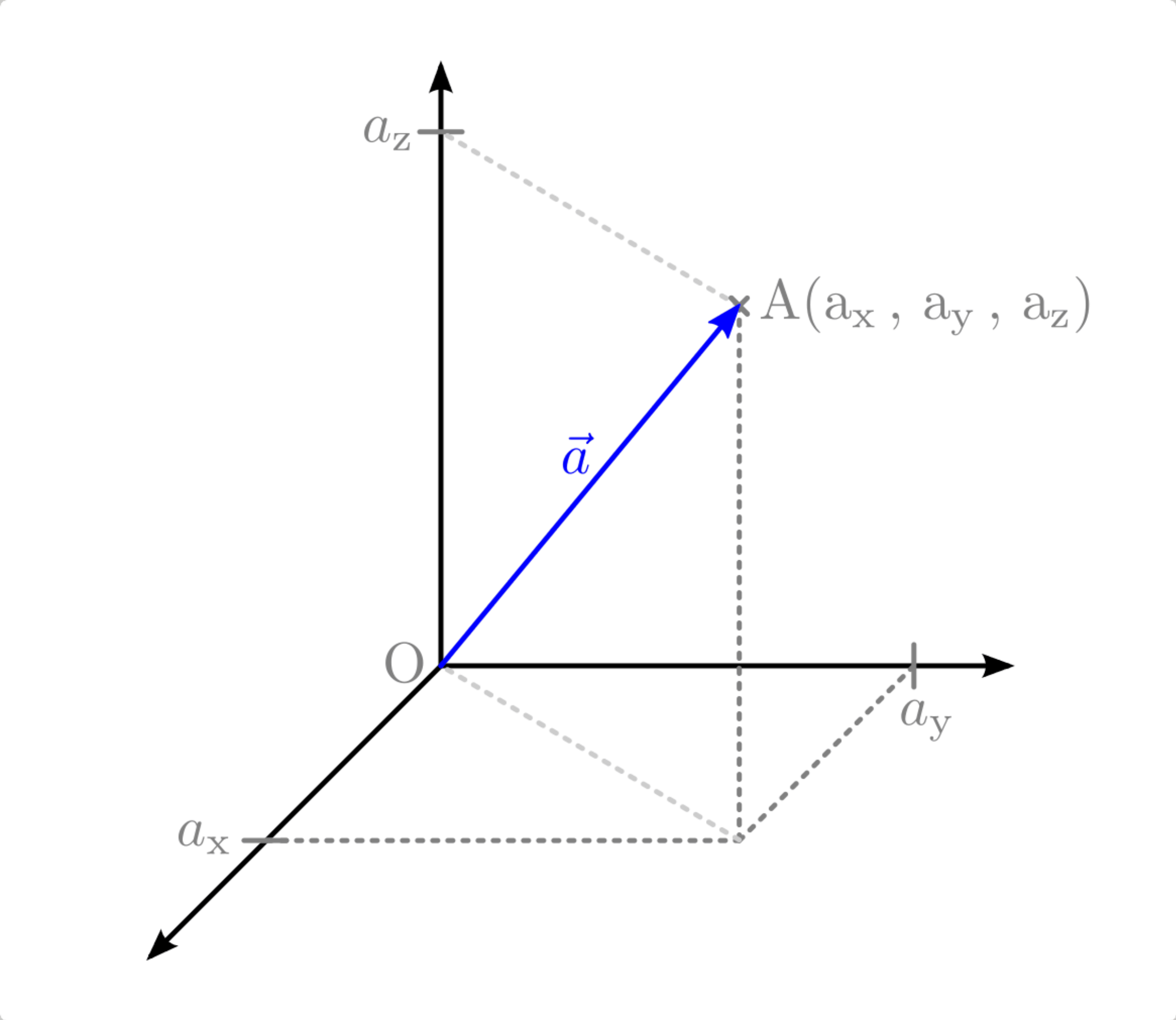
Vectors in 2D

$$\vec{a} = \begin{pmatrix} a_x \\ a_y \end{pmatrix}$$



Vectors in 3D

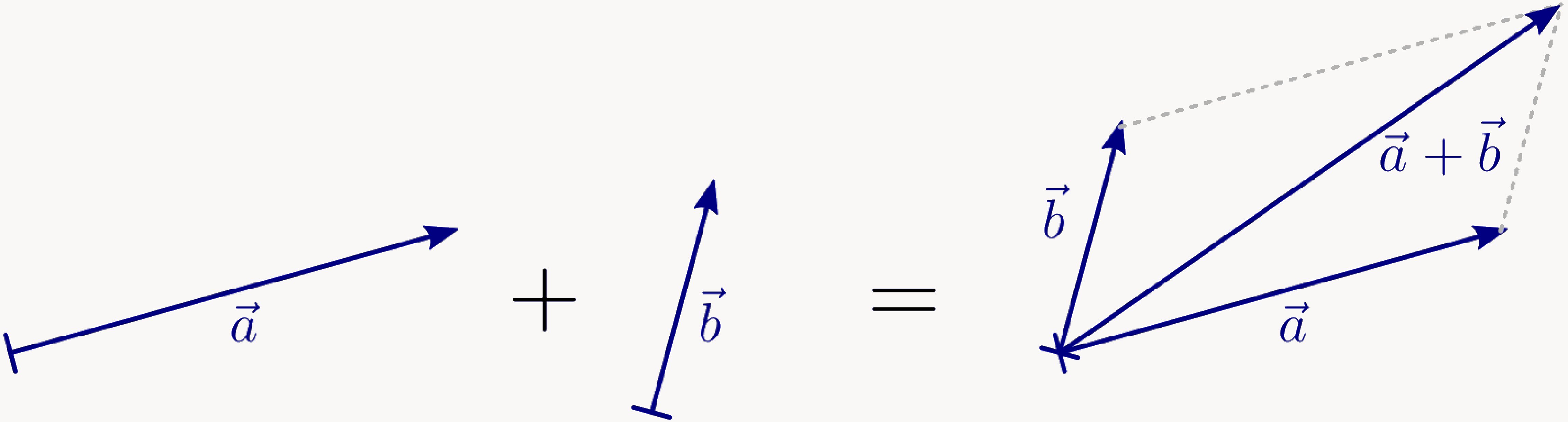
$$\vec{a} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}$$



Vectors in multidimensional space

$$\vec{a} = \begin{pmatrix} a_u \\ a_v \\ a_w \\ a_x \\ a_y \\ a_z \end{pmatrix}$$

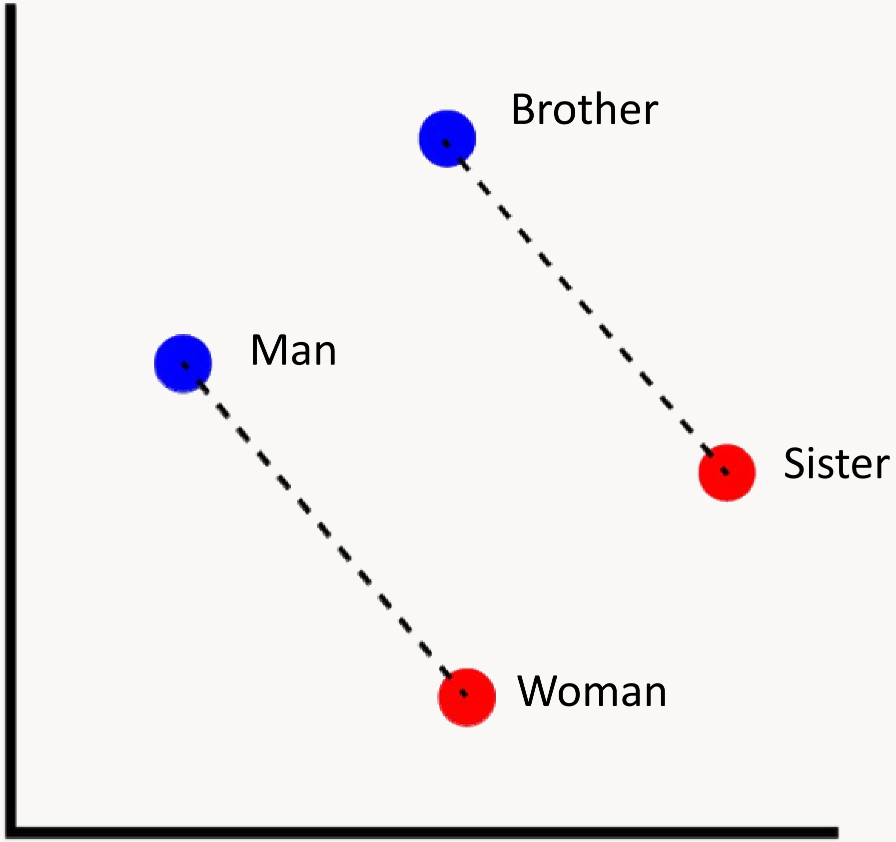
Calculation with vectors



Word2Vec

Mikolov et al., Google, 2013

$$Brother - Man + Woman \approx Sister$$



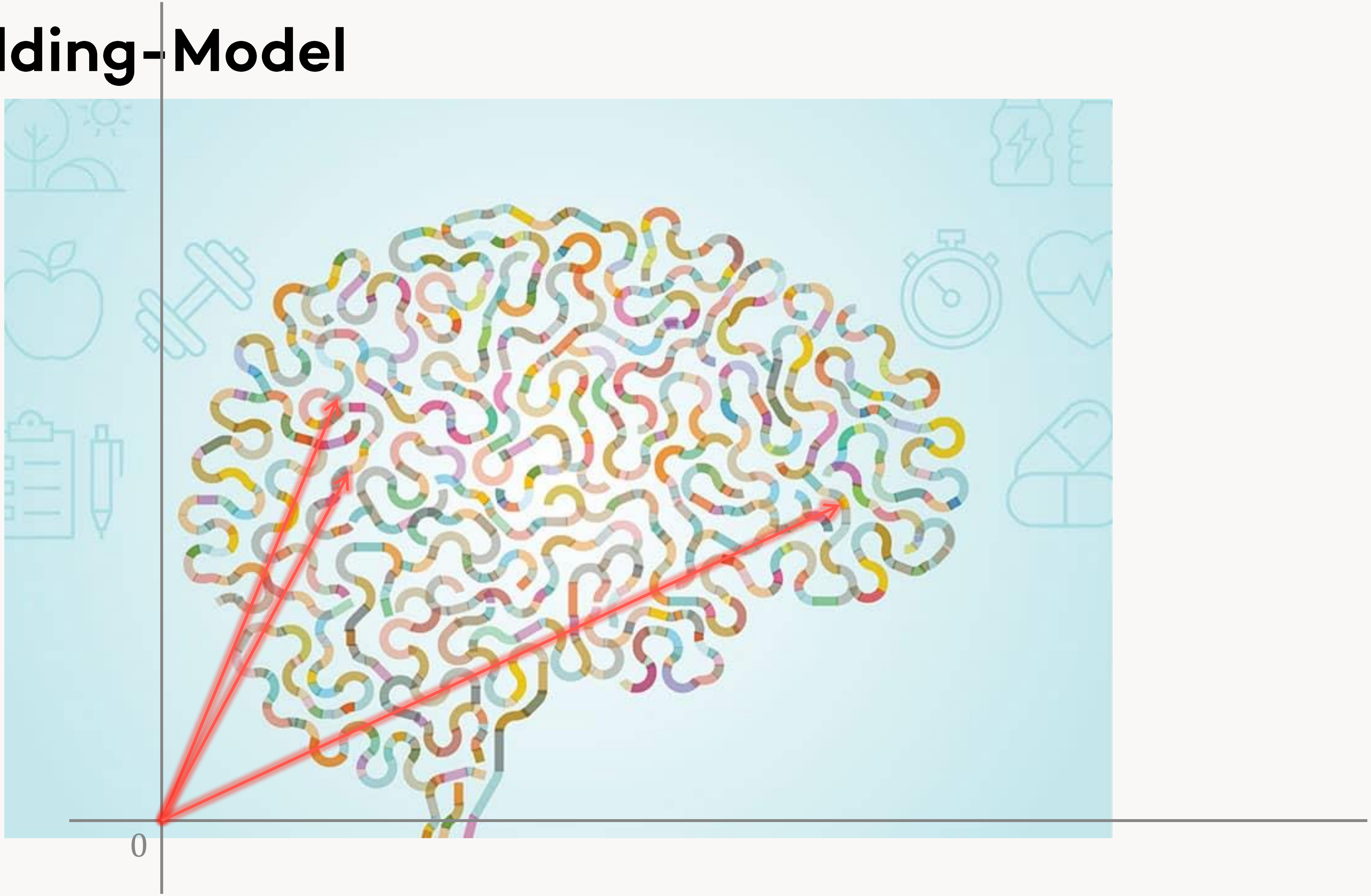
https://arxiv.org/abs/1301.3781

Embedding-Model

- Task: Create a vector from an input
    - Extract meaning / semantics
- Embedding models usually are very shallow & fast
  Word2Vec is only two layers
- Similar to the first step of an LLM
    - Convert text to values for input layer

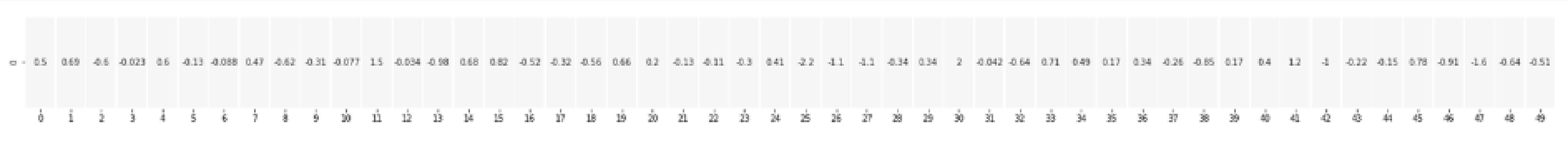
- This comparison is very simplified, but one could say:
    - The embedding model ‘maps’ the meaning into the model’s ‘brain’

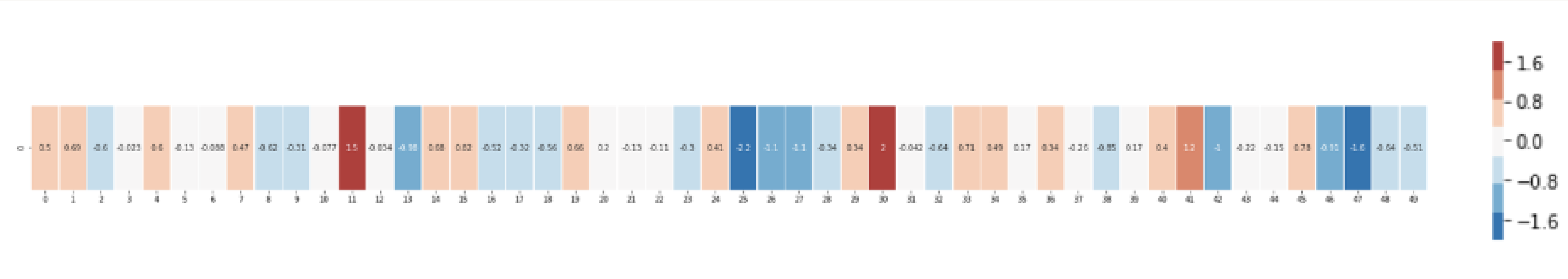
Embedding-Model



Embedding-Model

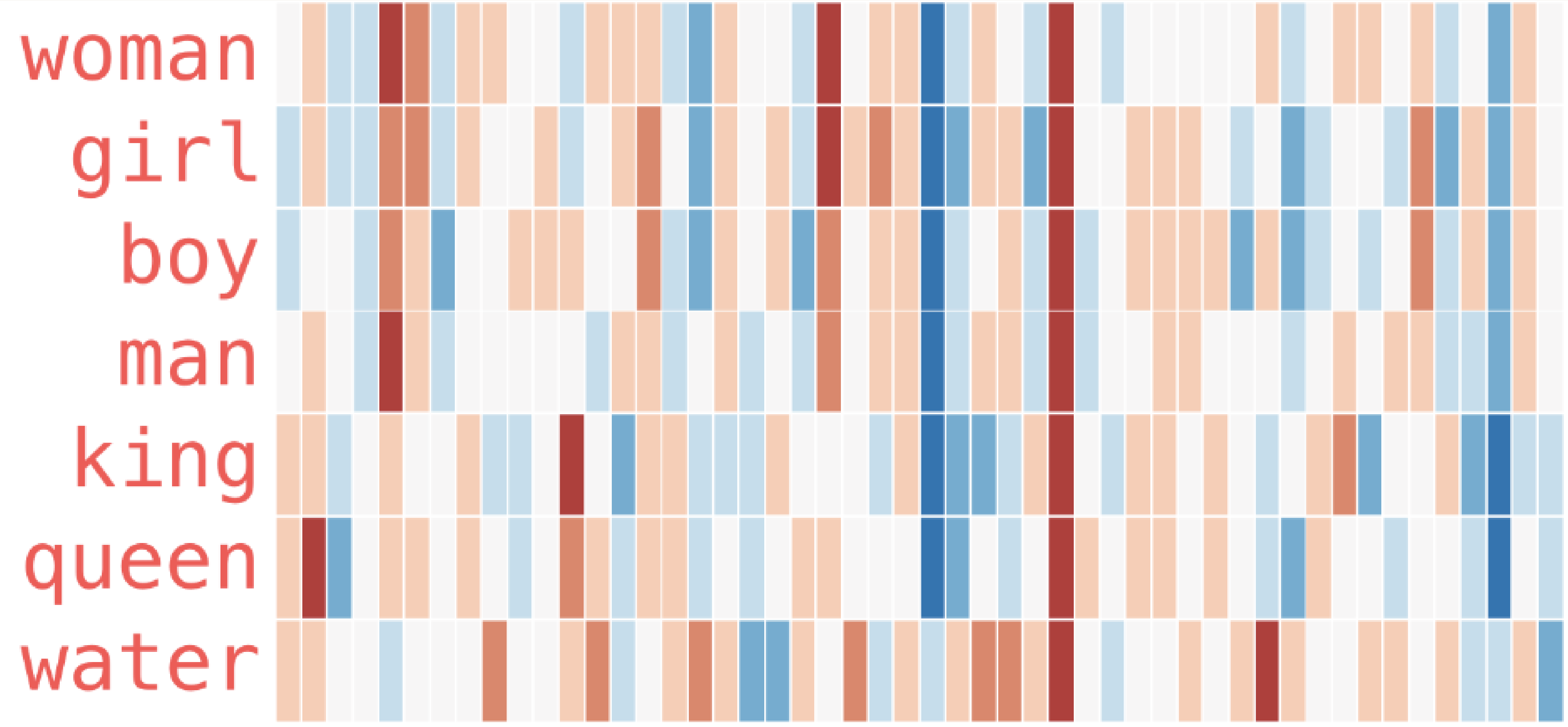
[0.50451 , 0.68607 , -0.59517 , -0.022801, 0.60046 , -0.13498 , -0.08813 , 0.47377 , -0.61798 , -0.31012 , -0.076666, 1.493 , -0.034189, -0.98173 , 0.68229 , 0.81722 , -0.51874 , -0.31503 , -0.55809 , 0.66421 , 0.1961 , -0.13495 , -0.11476 , -0.30344 , 0.41177 , -2.223 , -1.0756 , -1.0783 , -0.34354 , 0.33505 , 1.9927 , -0.04234 , -0.64319 , 0.71125 , 0.49159 , 0.16754 , 0.34344 , -0.25663 , -0.8523 , 0.1661 , 0.40102 , 1.1685 , -1.0137 , -0.21585 , -0.15155 , 0.78321 , -0.91241 , -1.6106 , -0.64426 , -0.51042]





http://jalammar.github.io/illustrated-word2vec/

Embedding-Model



http://jalammar.github.io/illustrated-word2vec/

Recap Embeddings

- Embedding model: “Analog to digital converter for text”
- Embeds the high-dimensional natural language meaning into a lower dimensional-space (the model’s ‘brain’)
- No magic, just applied mathematics
- Math. representation: Vector of n dimensions
- Technical representation: array of floating point numbers

Important: Model quality is key

- Select your embedding model carefully for **your use case!**

Model	**Hit rate**
intfloat/multilingual-e5-large-instruct	~ 50%
T-Systems-onsite/german-roberta-sentence-transformer-v2	< 70 %
danielheinz/e5-base-sts-en-de	> 80%

- Treat embedding models as exchangeable commodities
- Do evaluations on your data and your queries

DEMO

Embeddings

Sentence Transformers, local embedding model

Vector-Databases

Vector-Databases

- Mostly document-based
- Index: Embedding (vector)
- Document (content)
- Metadata
- Query functionalities

Vector-Databases

- Pinecone
- Milvus
- Chroma
- Weaviate
- Deep Lakee
- Qdrant
- Elasticsearch
- Vespa
- Vald
- ScaNN
- Pgvector
(PostgreSQL Extension)
- Faiss
- …

- … (probably) coming to a relational database near you soon(ish)
SQL Server Example: https://learn.microsoft.com/en-us/samples/azure-samples/azure-sql-db-openai/azure-sql-db-openai/

Vector-Databases

- (Search-)Algorithms
  - Cosine Similarity $S_{C(\mathrm{a,b})} = \frac{\mathrm{a} \cdot b}{\|a\| \times \|b\|}$
  - Manhatten Distance (L1 norm, taxicab)
  - Euclidean Distance (L2 norm)
  - Minkowski Distance (~ generalization of L1 and L2 norms)
  - L∞ (L-Infinity), Chebyshev Distance
  - Jaccard index / similarity coefficient (Tanimoto index)
  - Nearest Neighbour
  - Bregman divergence
  - …

Demo

Vector database

LangChain, Chroma, local embedding model

Indexing

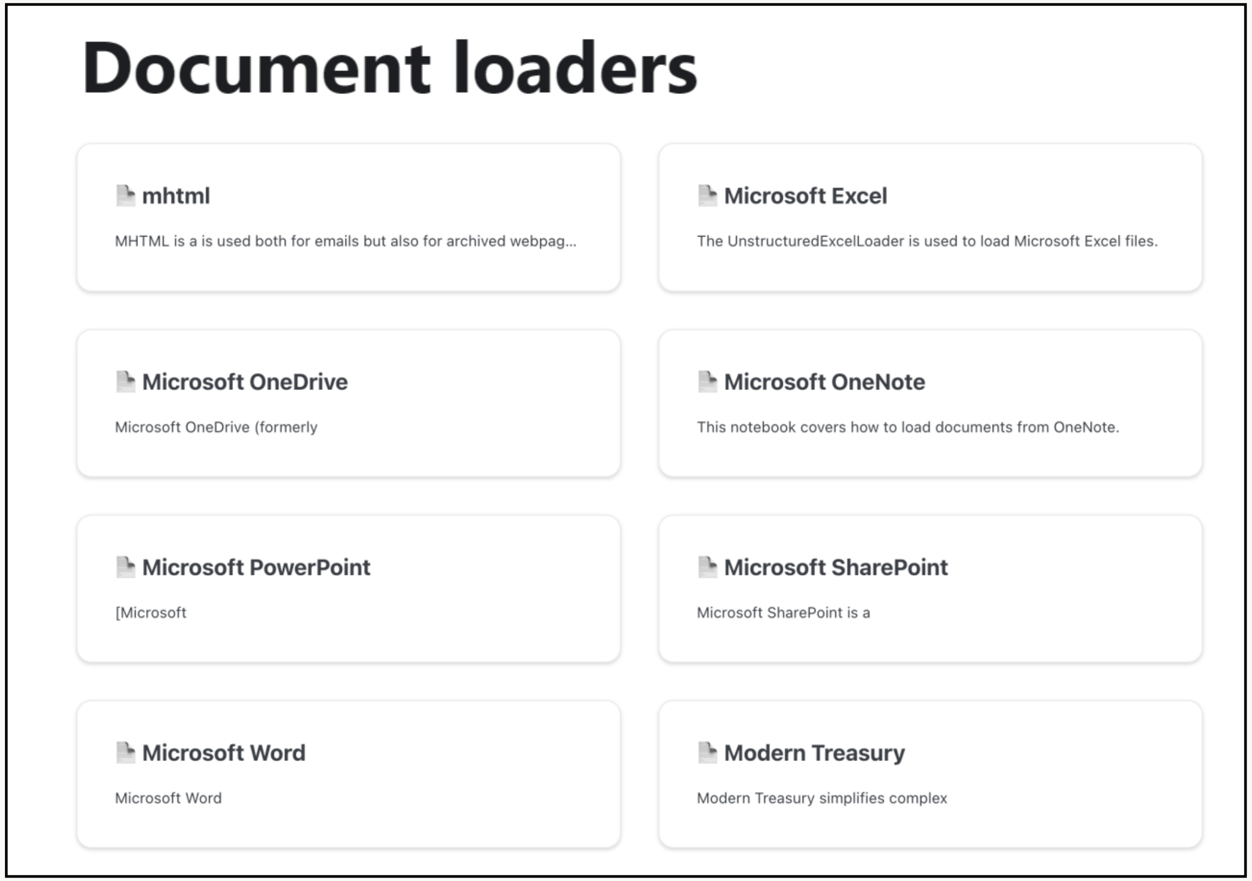
Indexing

Overview over the next topics:

- Loading
- Clean-up
- Splitting
- Embedding
- Storing

Loading

- Import documents from different sources, in different formats
- LangChain has very strong support for loading data
- Support for cleanup
- Support for splitting



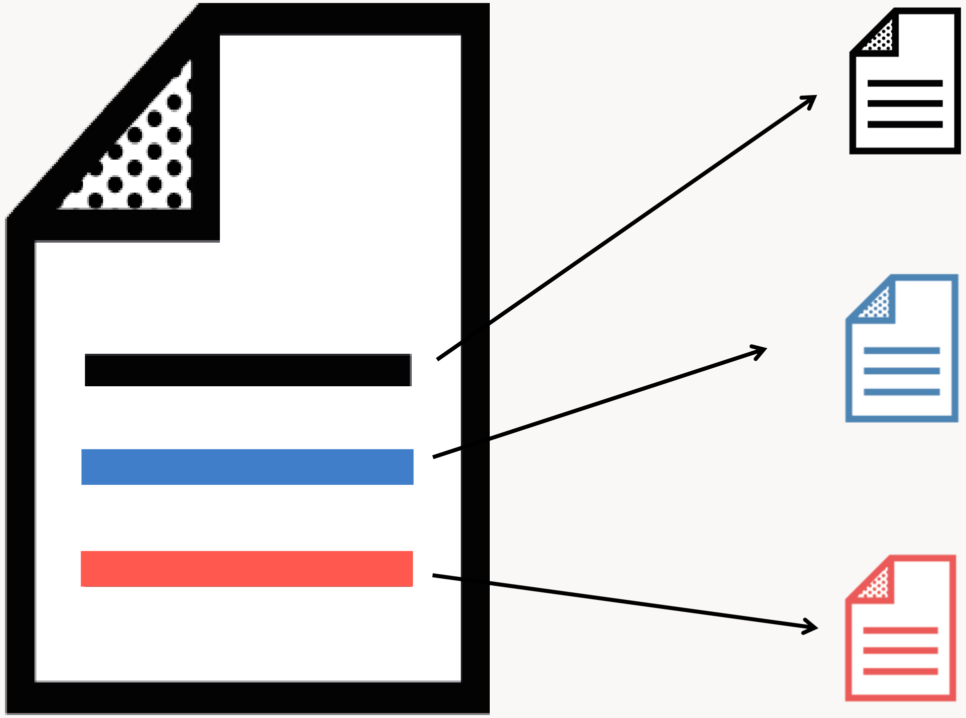
https://python.langchain.com/docs/integrations/document_loaders

Clean-up

- HTML Tags
- Formatting information
- Normalization
    - lowercasing
    - stemming, lemmatization
    - remove punctuation & stop words
- Enrichment
    - tagging
    - keywords, categories
    - metadata

Splitting (Text Segmentation)

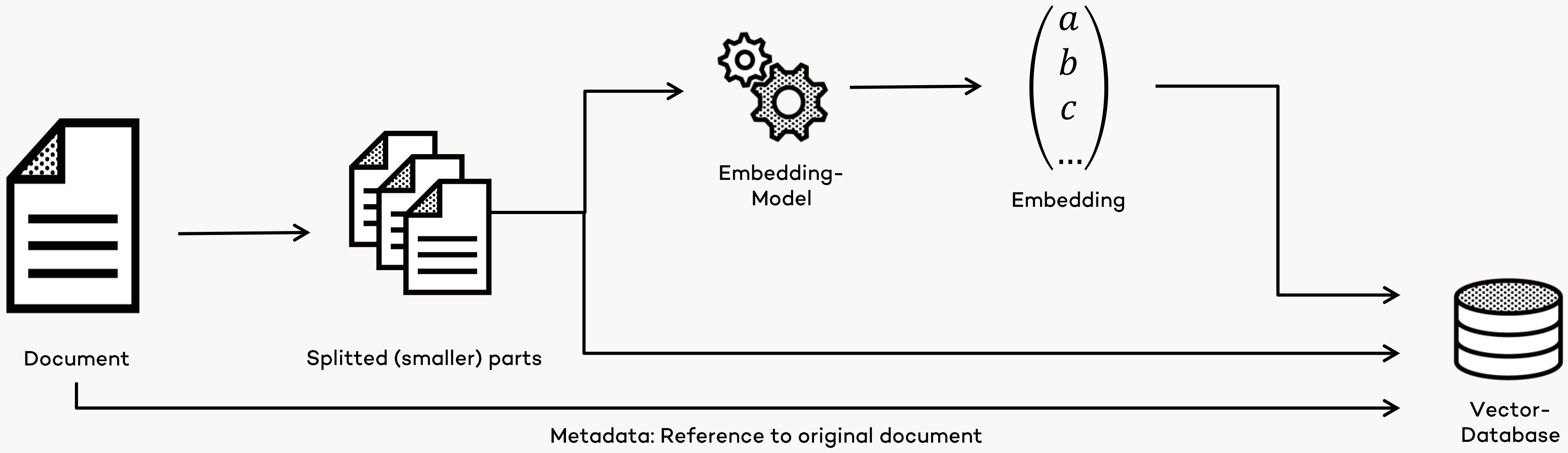
- Document is too large / too much content / not concise enough



- by size (text length)
- by character (\n\n)
- by paragraph, sentence, words (until small enough)
- by size (tokens)
- overlapping chunks (token-wise)
- by semantics (semantic chunker)

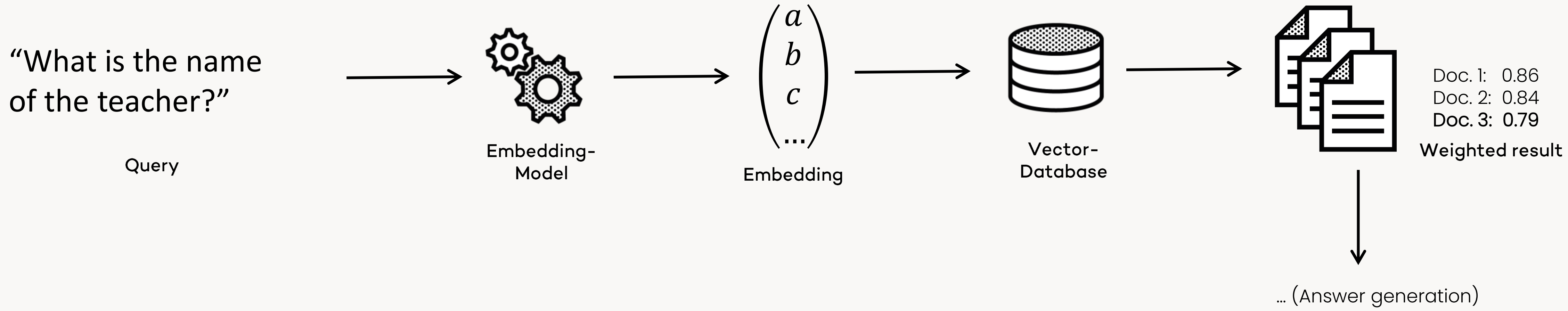
Vector-Databases

- Indexing



Retrieval (Search)

Retrieval


"What is the name of the teacher?"
Query
Embedding-Model
a
b
c
...
Embedding
Vector-Database
Doc. 1: 0.86
Doc. 2: 0.84
Doc. 3: 0.79
Weighted result
... (Answer generation)

DEMO

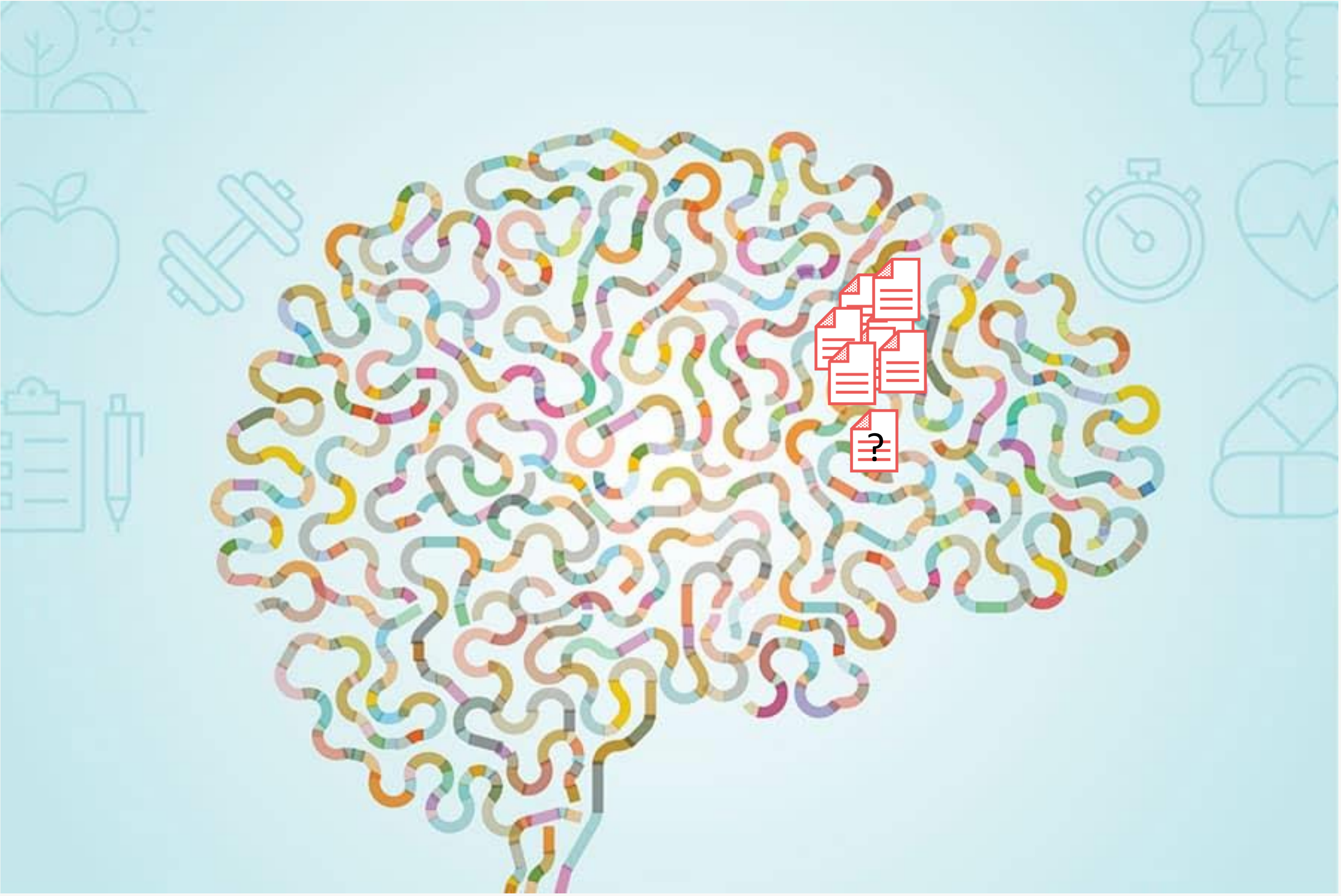
Store and retrieval

LangChain, Chroma, local embedding model, OpenAI GPT

Indexing II
Not good enough?

Introduction | Embeddings | Vector-DBs | Indexing | Retrieval | **Indexing II** | RAG

Not good enough?

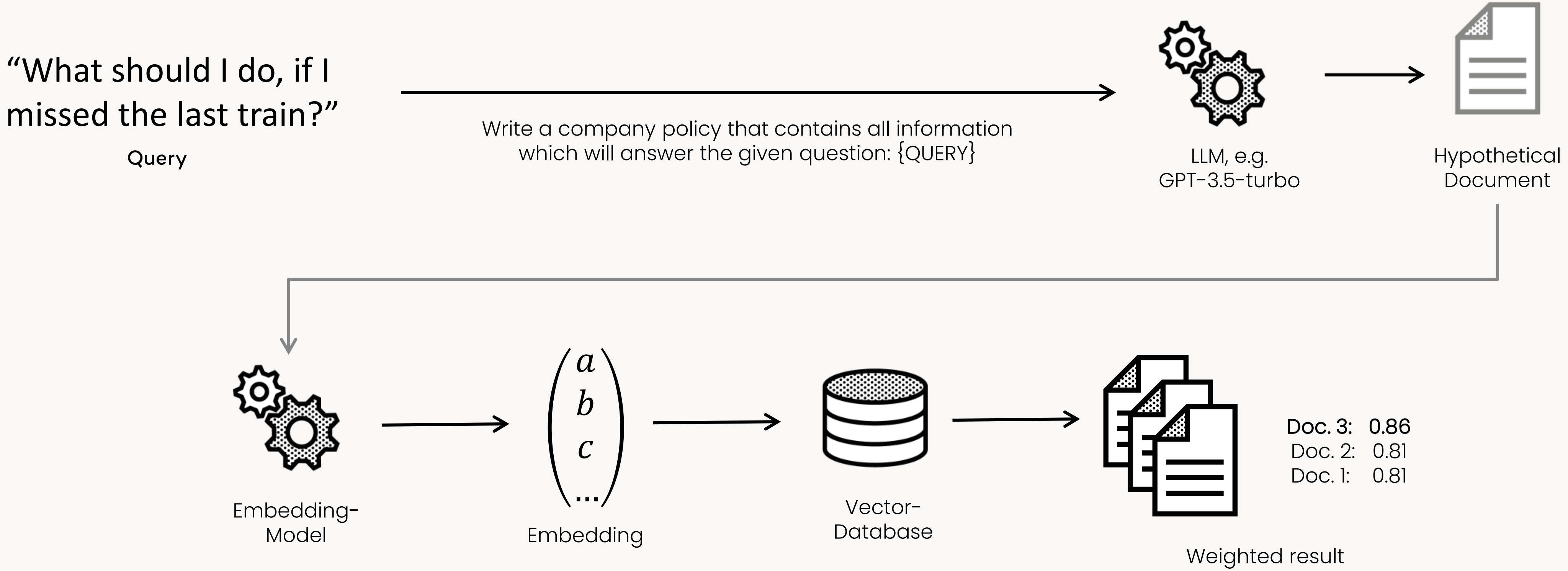


Not good enough?

- Semantic search still only searches your indexed documents
- It’s just as good as your embeddings
- Sh*t in, sh*t out

HyDE (Hypothetical Document Embedddings)

- Search for a hypothetical Document



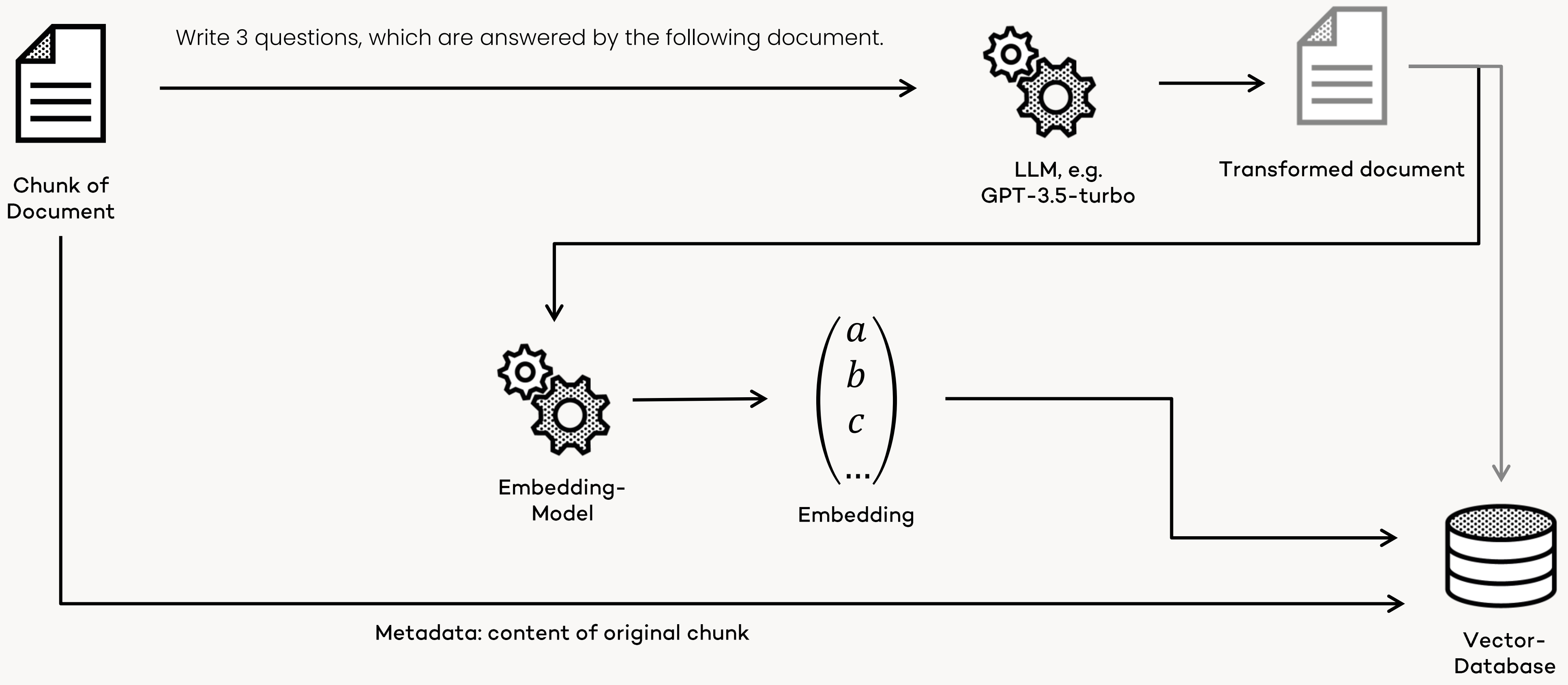
https://arxiv.org/abs/2212.10496

What else?

- Downside of HyDE:
  - Each request needs to be transformed through an LLM (slow & expensive)
  - A lot of requests will probably be very similar to each other
  - Each time a different hyp. document is generated, even for an extremely similar request
    - Leads to very different results each time
- Idea: Alternative indexing
  - Transform the document, not the query

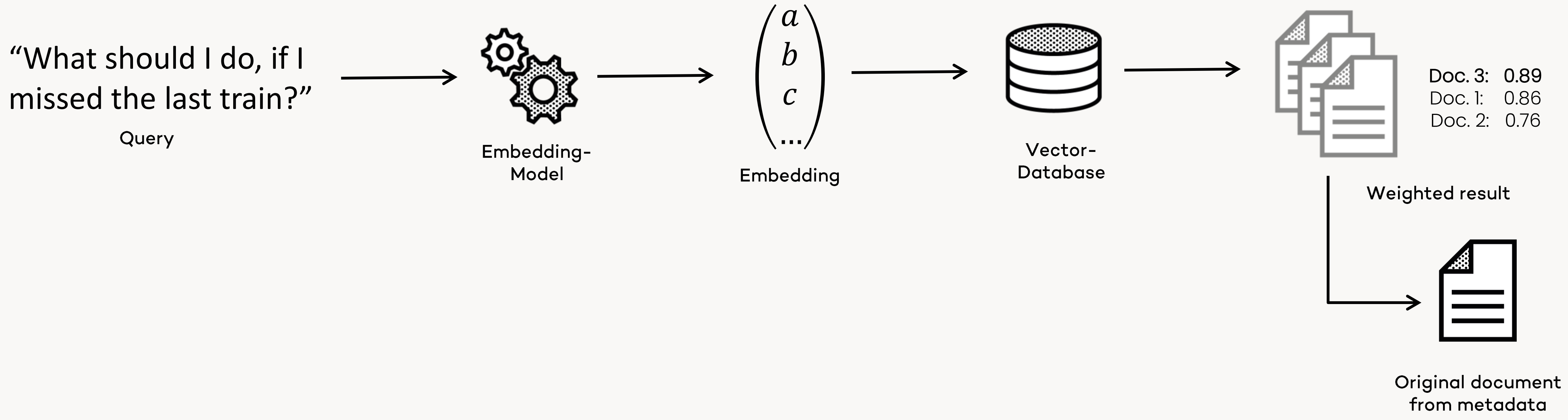
Alternative Indexing

HyQE: Hypothetical Question Embedding



Alternative Indexing

- Retrieval



Demo

Compare embeddings

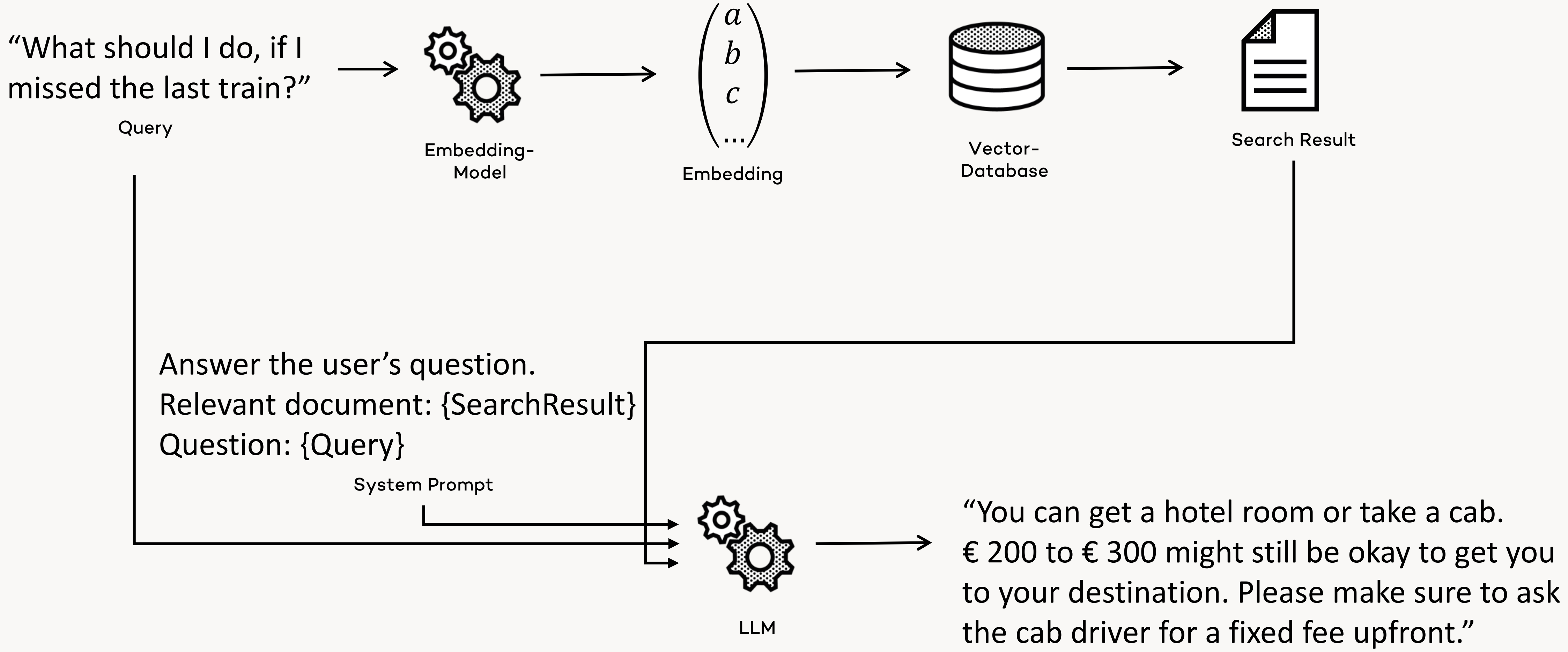
LangChain, Qdrant, OpenAI GPT

Recap: Not good enough?

- Tune text cleanup, segmentation, splitting
- HyDE or HyQE or alternative indexing
  - How many questions?
  - With or without summary
- Other approaches
  - Only generate summary
  - Extract “Intent” from user input and search by that
  - Transform document and query to a common search embedding
  - HyKSS: Hybrid Keyword and Semantic Search
    https://www.deg.byu.edu/papers/HyKSS.pdf
- **Always** evaluate approaches with your own data & queries
- The actual / final approach is more involved
  as it seems on the first glance

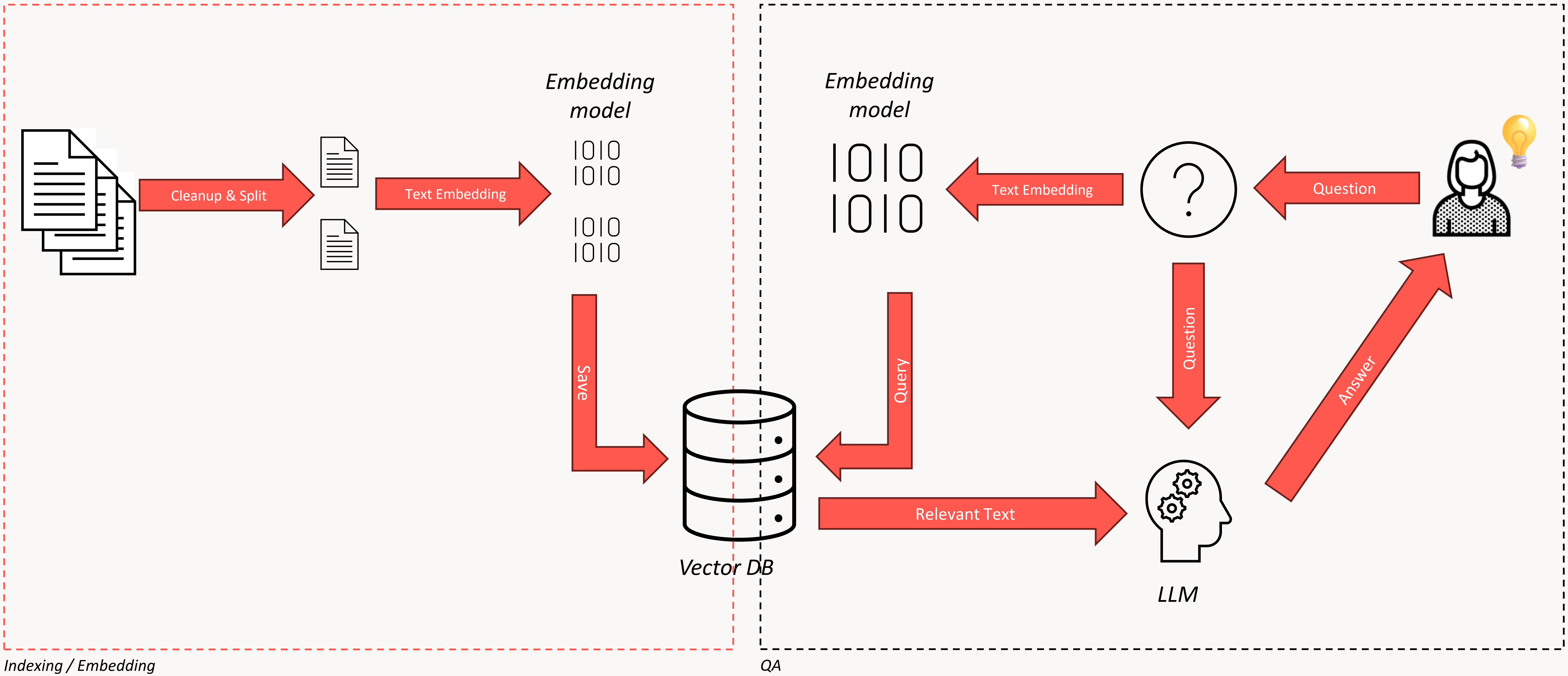
Responses: RAG

RAG (Retrieval Augmented Generation)



Retrieval-augmented generation (RAG)

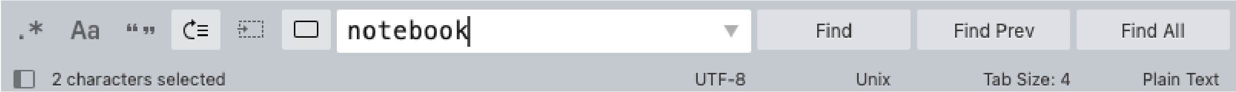
Indexing & (Semantic) search

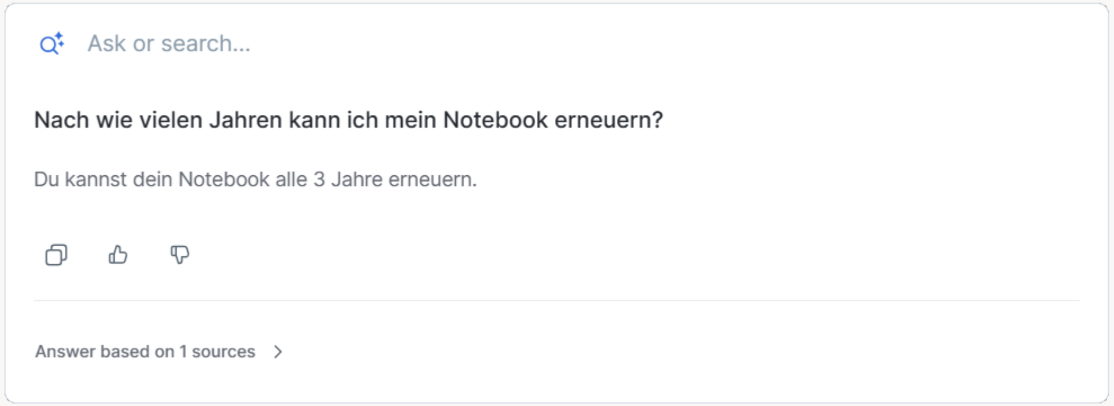


Conclusion

Conclusion

- Semantic search is a first and fast Generative AI business use-case
- Quality of results depend heavily on data quality and preparation pipeline
- RAG pattern can produce breathtaking good results without the need for user training





Thank you!

Demos:

https://github.com/thinktecture-labs/dwx-2024-advanced-rag

Sebastian Gingter

https://thinktecture.com/sebastian-gingter

Generative AI
Real-World Chat mit Unternehmensdaten: Advanced RAG jenseits von “Hallo Welt”!

Slides & Code

https://www.thinktecture.com/de/sebastian-gingter

Sebastian Gingter
sebastian.gingter@thinktecture.com
Developer Consultant