

# “Talk to your Data”: Improving RAG solutions based on real-world experiences

think  
tecture



Sebastian Gingter  
sebastian.gingter@thinktecture.com  
Developer Consultant



# 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



# What to expect (and what not):

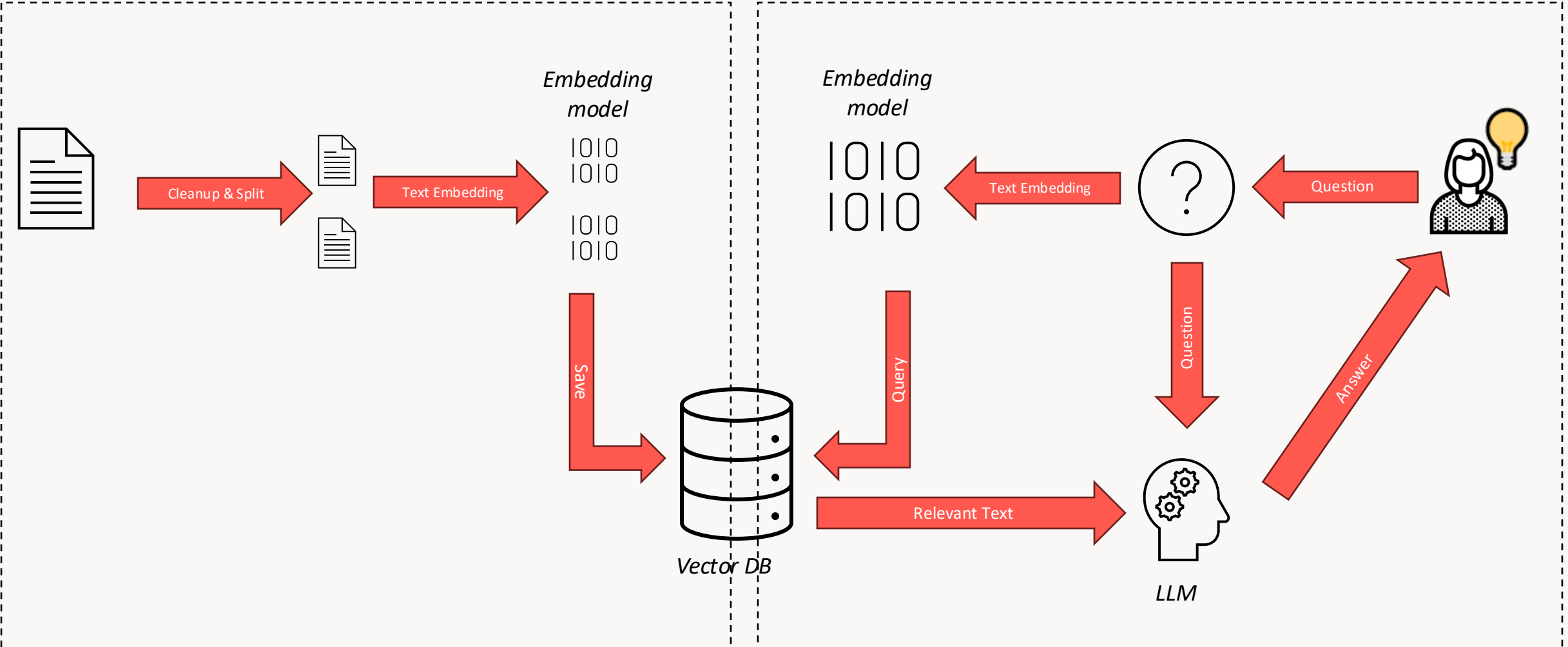
- Some background info and theory
- Overview over semantic search
- Problems and possible strategies
- Pragmatic approaches for your own data
- No C#, some Python 🤖
- No deep-dive into
  - LLMs
  - LangChain

# Agenda

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

# Introduction

# Use case: Retrieval-augmented generation (RAG)



## Other use-cases:

- Similarity determination
- Semantic search
- Semantic routing
- Semantic caching
- Categorization
- etc.

# 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

# Semantic Search

- We need a numeric representation of text
  - Tokens
- We need a numeric representation of meaning
  - Embeddings

# Tokens

- Similar to char tables (e.g. ASCII), just with larger elements
- Tokens are parts of text
  - Words
  - Syllables
  - Punctuation
  - ...
- Tokens are translated to token IDs
- Example: <https://platform.openai.com/tokenizer>

# Embeddings

# 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

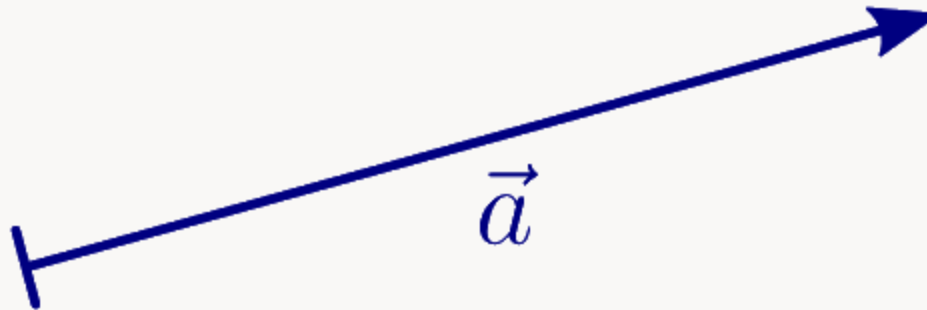
# Embeddings

- Embedding models are unique
- Each dimension has a different meaning, individual to the model
- Vectors from different models are incompatible with each other
  - they live in different vector spaces
- 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

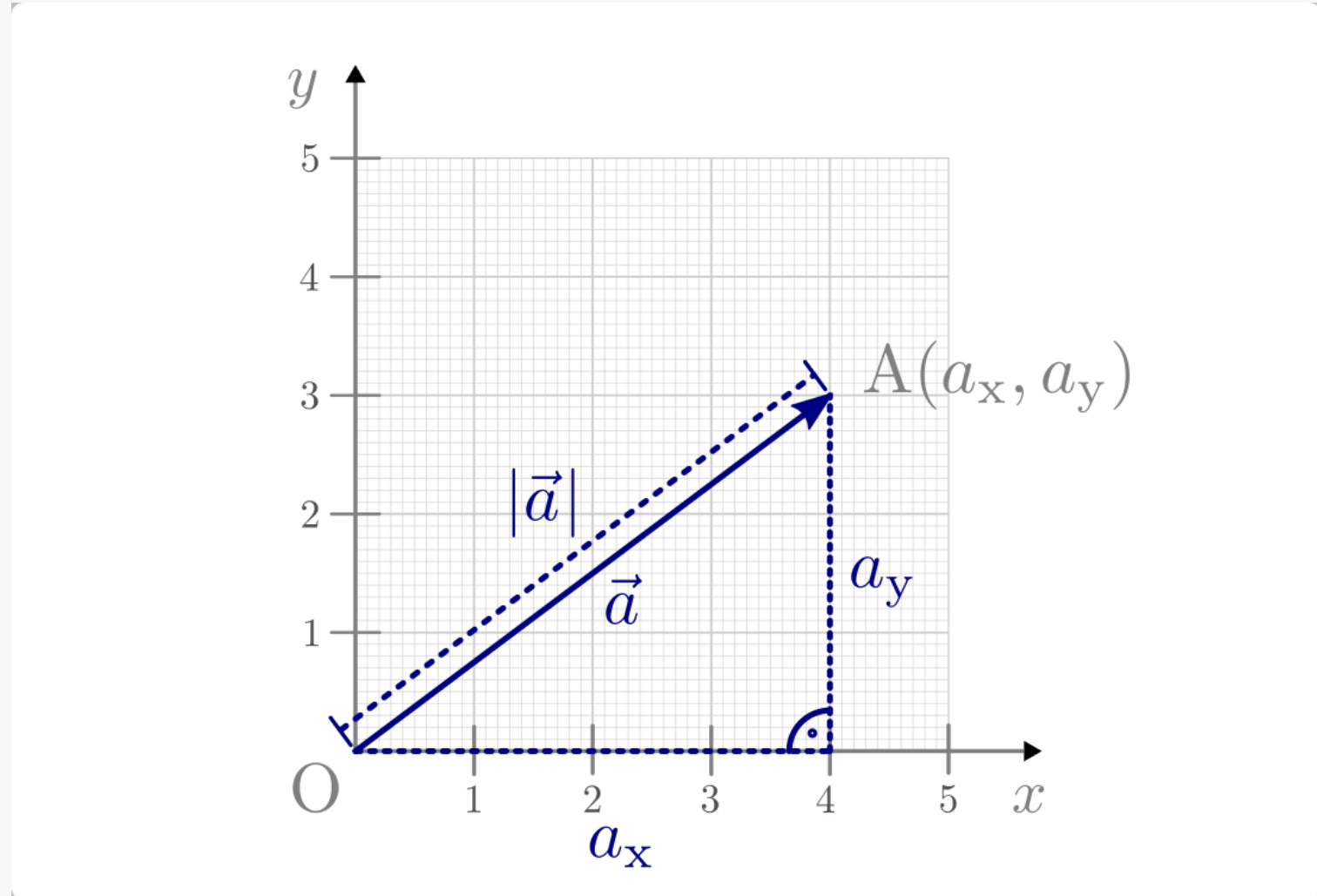
- $\vec{a} = \begin{pmatrix} a_x \\ a_y \end{pmatrix}$





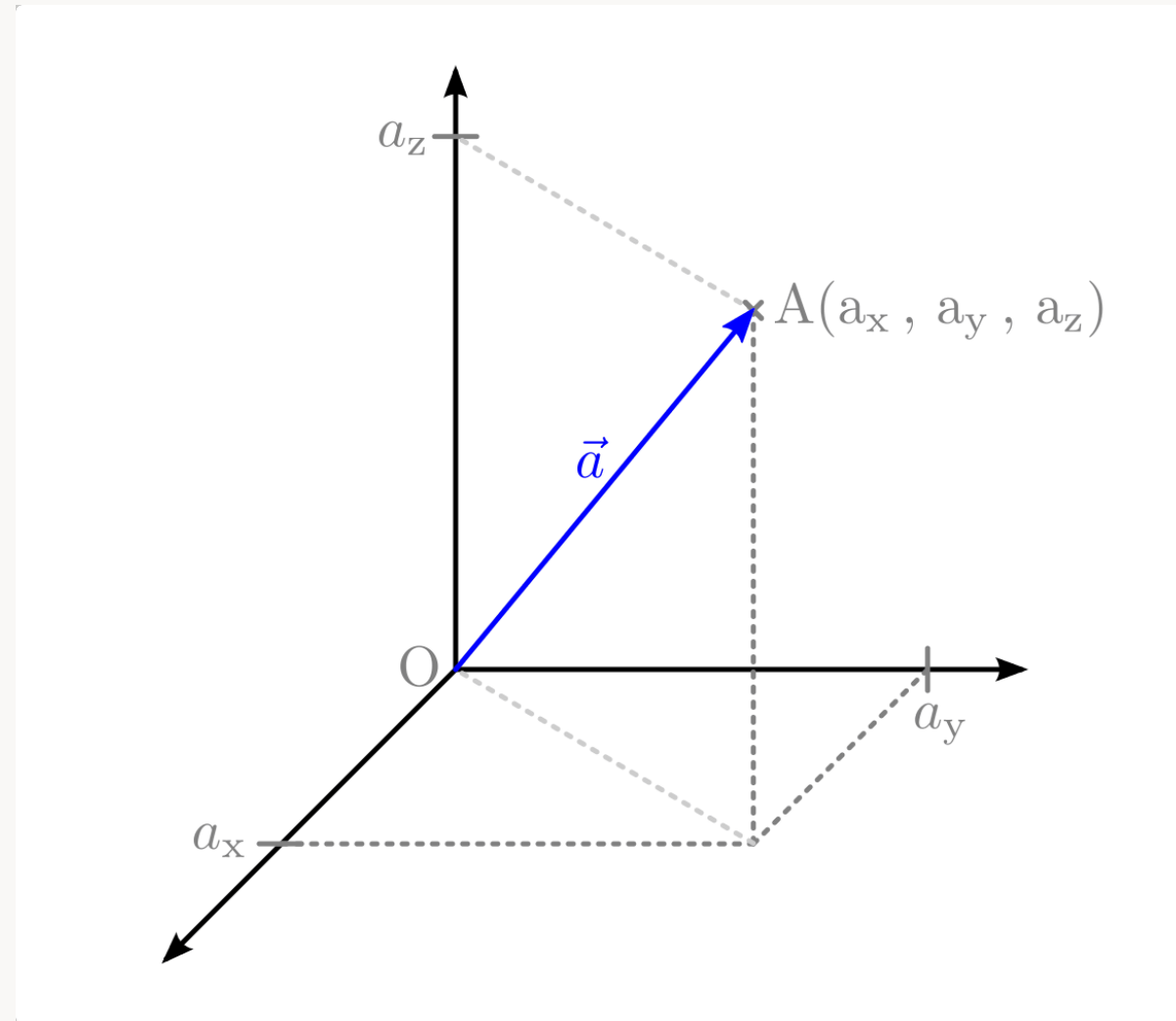
# Vectors in 2D

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



# Vectors in 3D

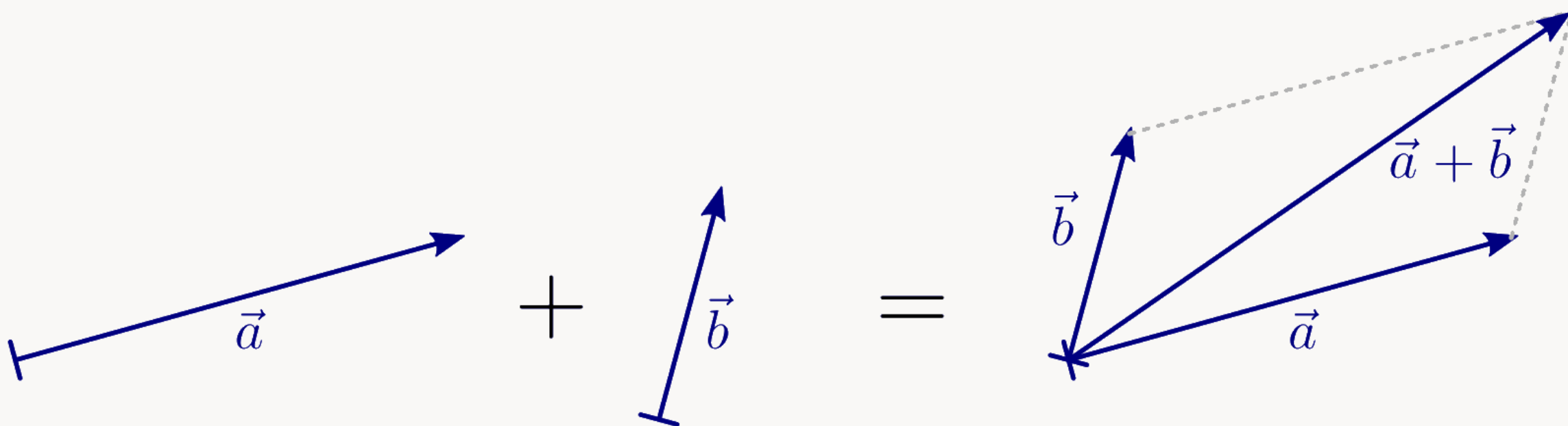
$$\vec{a} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}$$



# Vectors in multidimensional space

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

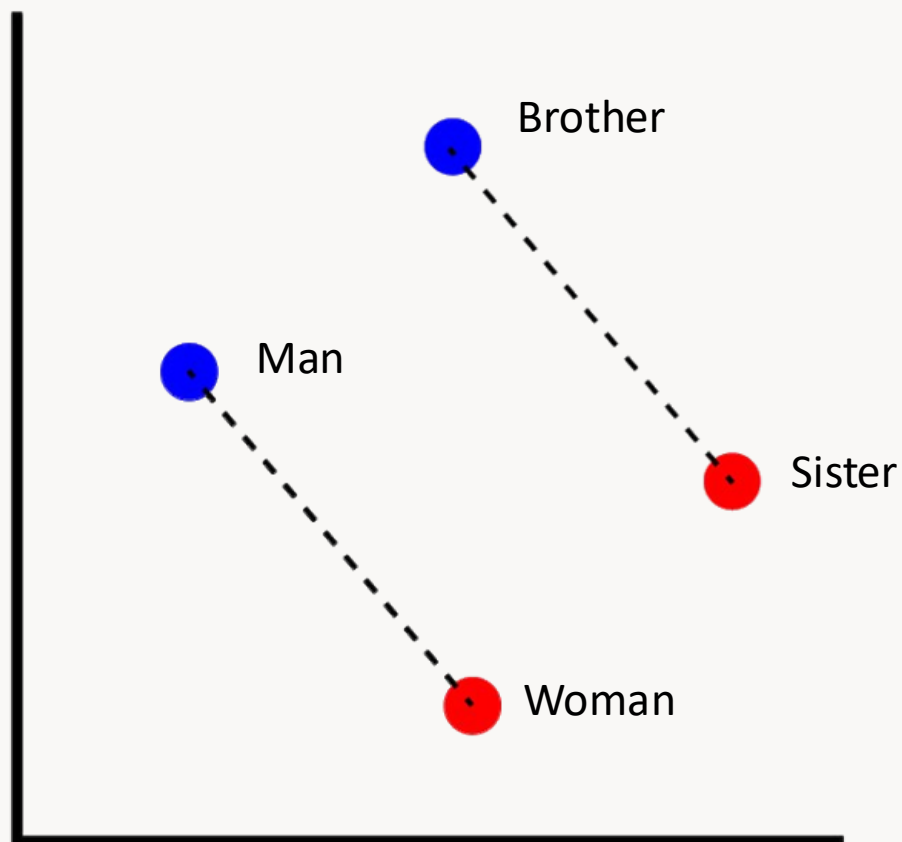
# Calculation with vectors



# Word2Vec

Mikolov et al., Google, 2013

$$\textit{Brother} - \textit{Man} + \textit{Woman} \approx \textit{Sister}$$



# Embedding-Model

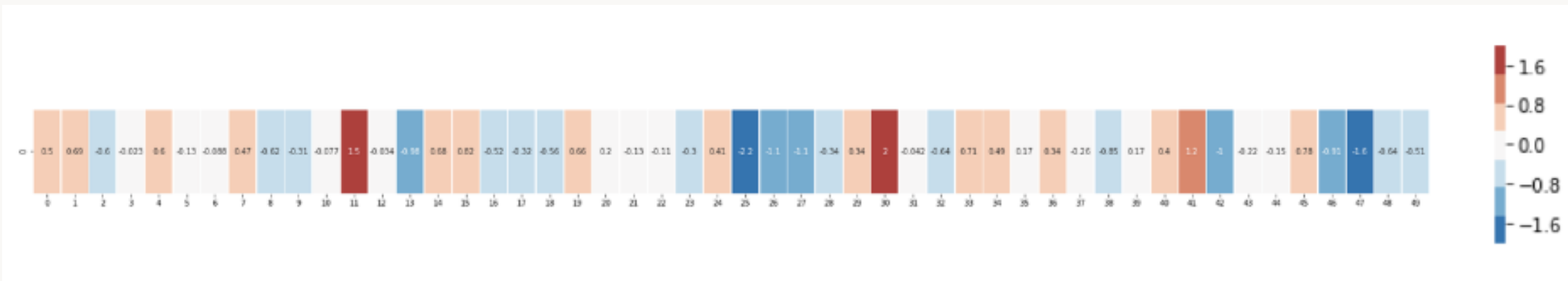
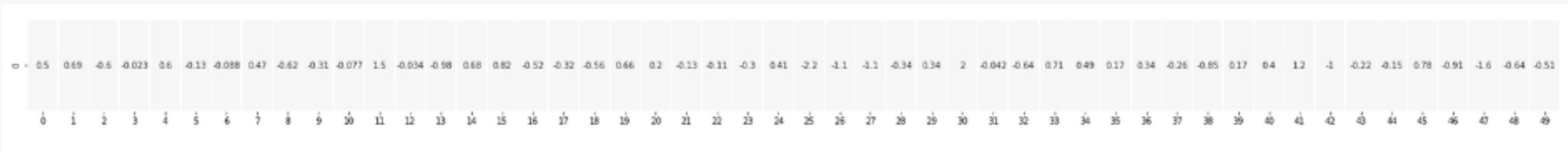
- Task: Create a vector from an input
  - Extract meaning / semantics
- Embedding models usually are very shallow & fast  
Word2Vec is only two layers
- Similar to the first step of an LLM
  - Convert text to values for input layer
- This comparison is very simplified, but one could say:
  - The embedding model 'maps' the meaning into the model's 'brain'

# Vectors from your Embedding-Model



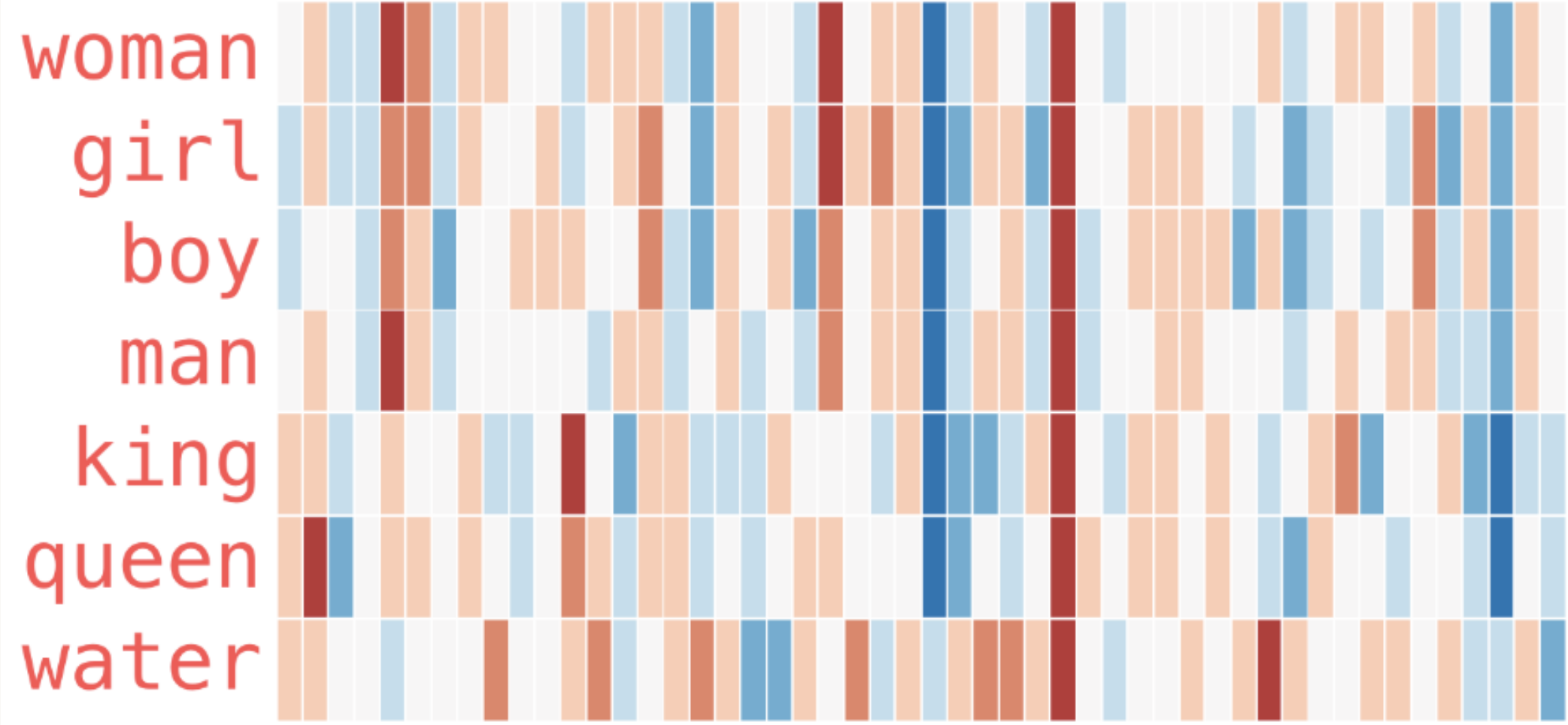
# Embedding-Model

[ 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]





# Embedding-Model



# Important

- Select your Embedding Model carefully for your use case
- e.g.
  - intfloat/multilingual-e5-large-instruct ~ 50 %
  - T-Systems-onsite/german-roberta-sentence-transformer-v2 < 70 %
  - danielheinz/e5-base-sts-en-de > 80 %
- Maybe fine-tuning of the embedding model might be an option
- As of now: Treat embedding models as exchangeable commodities!

# 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

# DEMO

## Embeddings

Sentence Transformers, local embedding model

# Indexing









# Indexing

- 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

## Document loaders

 <b>mhtml</b> MHTML is a is used both for emails but also for archived webpag...	 <b>Microsoft Excel</b> The UnstructuredExcelLoader is used to load Microsoft Excel files.
 <b>Microsoft OneDrive</b> Microsoft OneDrive (formerly	 <b>Microsoft OneNote</b> This notebook covers how to load documents from OneNote.
 <b>Microsoft PowerPoint</b> [Microsoft	 <b>Microsoft SharePoint</b> Microsoft SharePoint is a
 <b>Microsoft Word</b> Microsoft Word	 <b>Modern Treasury</b> Modern Treasury simplifies complex

# Clean-up

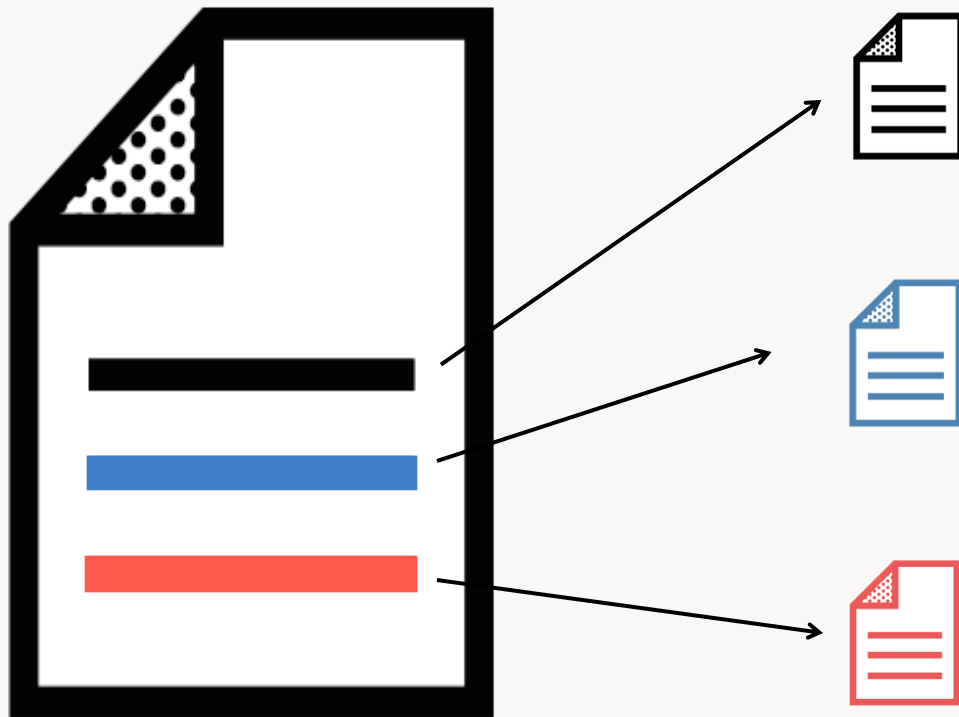
- HTML Tags
- Formatting information
- Normalization
  - lowercasing
  - stemming, lemmatization
  - remove punctuation & stop words
- Enrichment
  - tagging
  - keywords, categories
  - metadata





# Splitting (Text Segmentation)

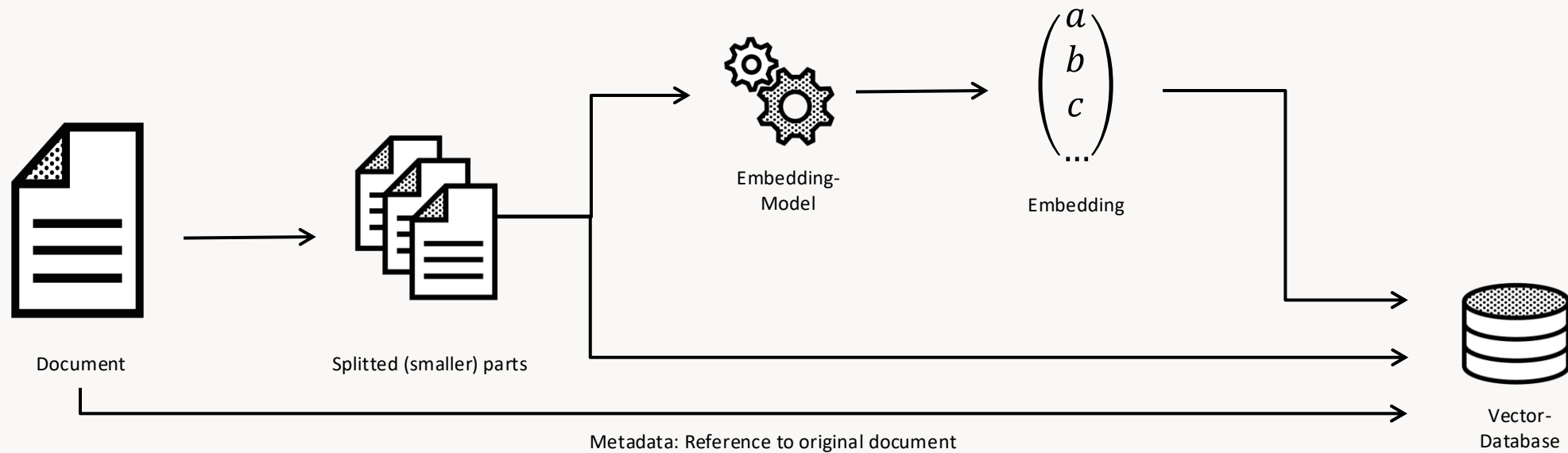
- Document is too large / too much content / not concise enough



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

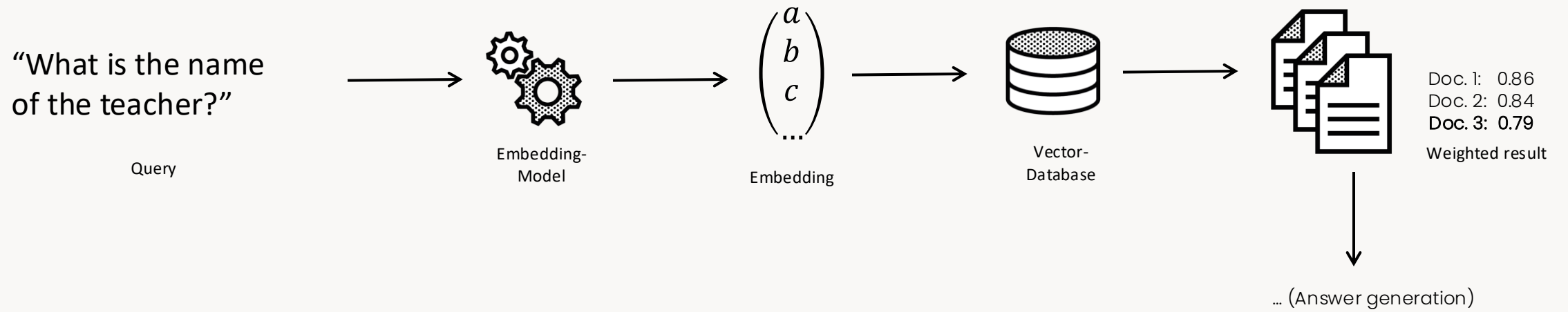
# Vector-Databases

- Indexing



# Retrieval (Search)

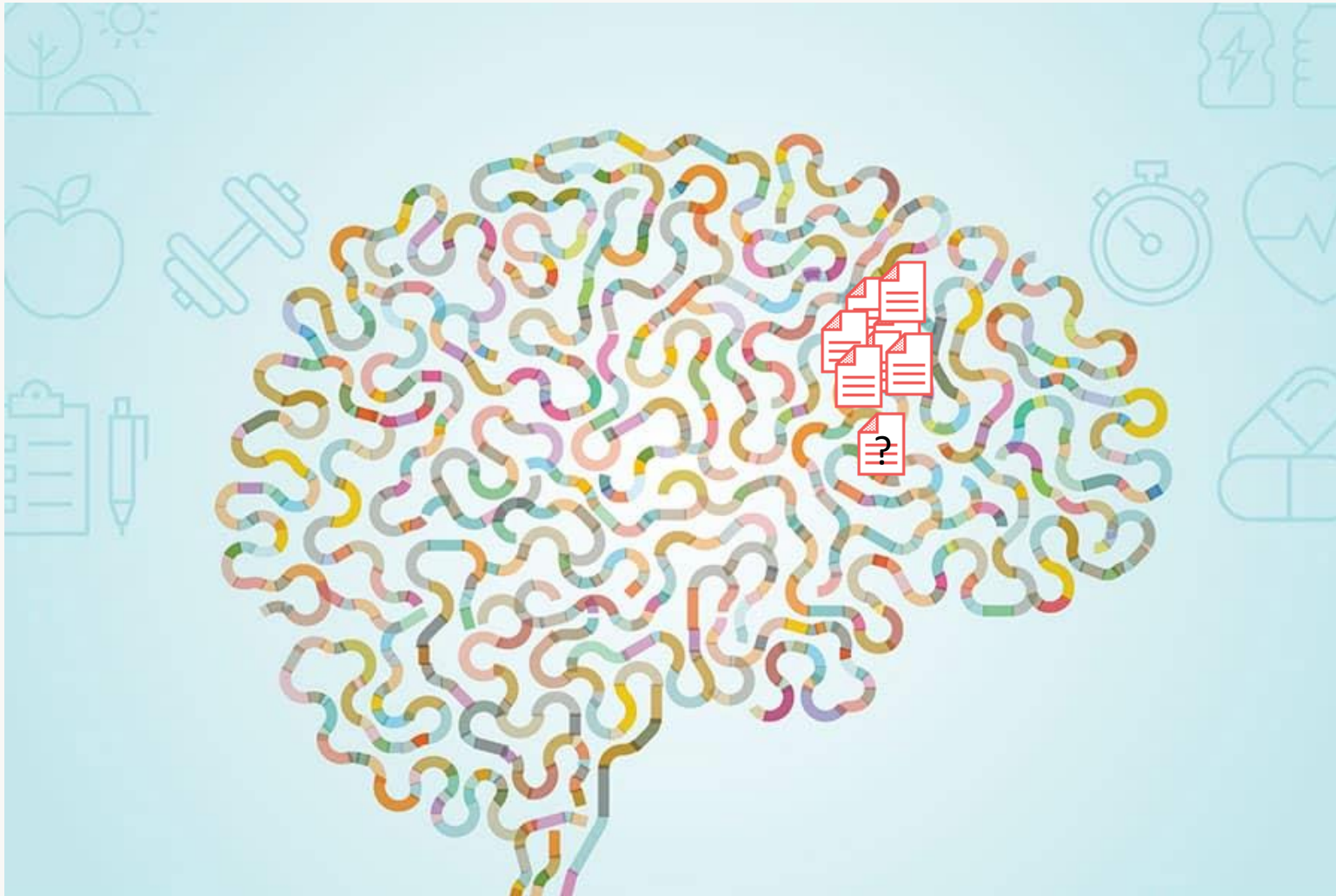
# Retrieval



# Indexing II

## Not good enough?

## Not good enough?

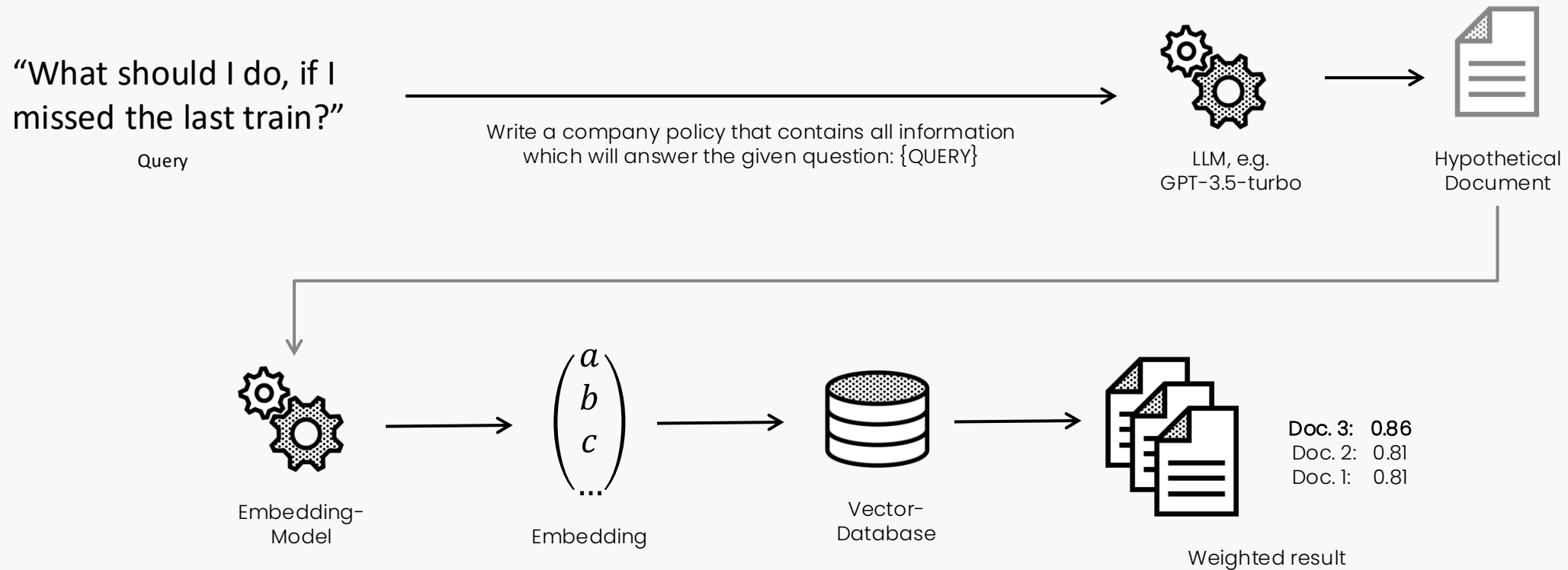


# Not good enough?

- Semantic search still only uses your data
- It's just as good as your embeddings
  - All chunks need to be sized correctly and distinguishable enough
- Garbage in, garbage out

# HyDE (Hypothetical Document Embeddings)

## ■ Search for a hypothetical Document



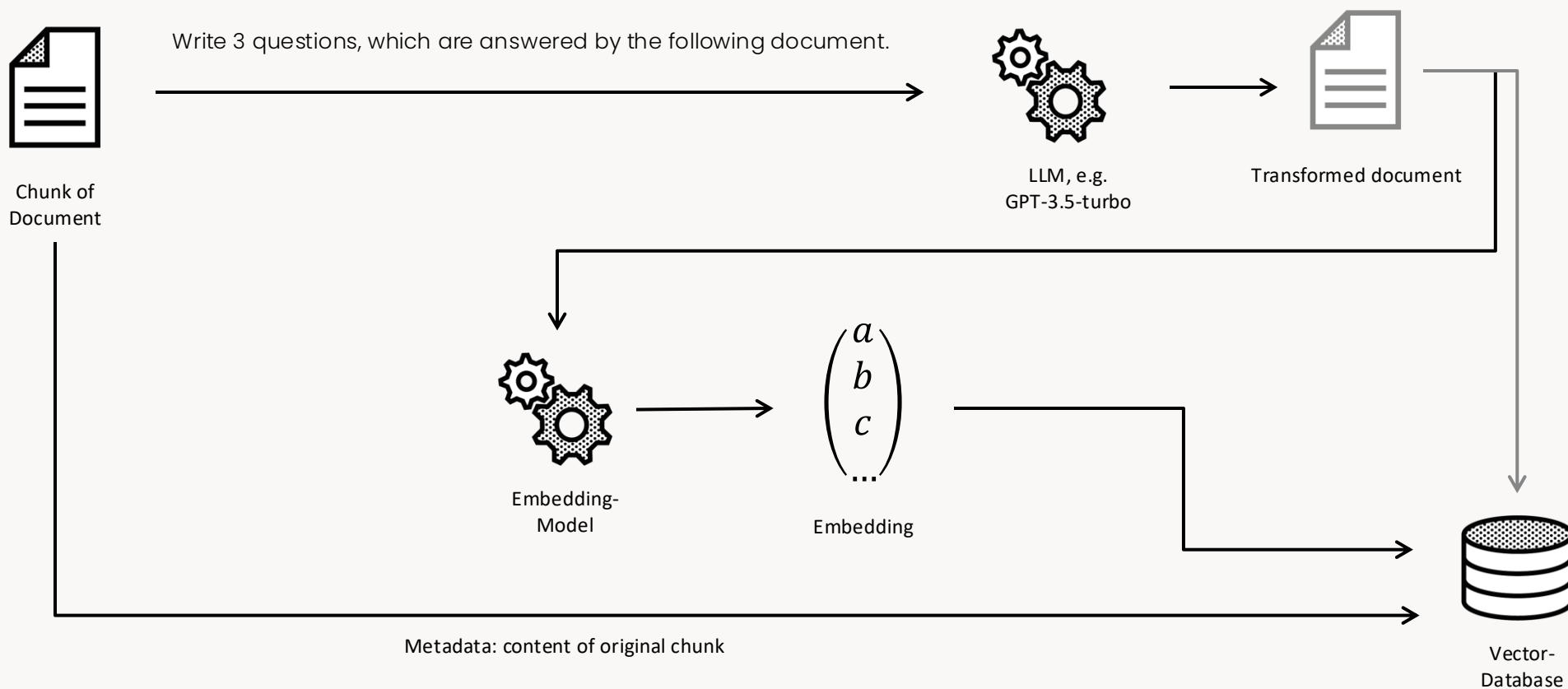


# What else?

- 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 hypothetical 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

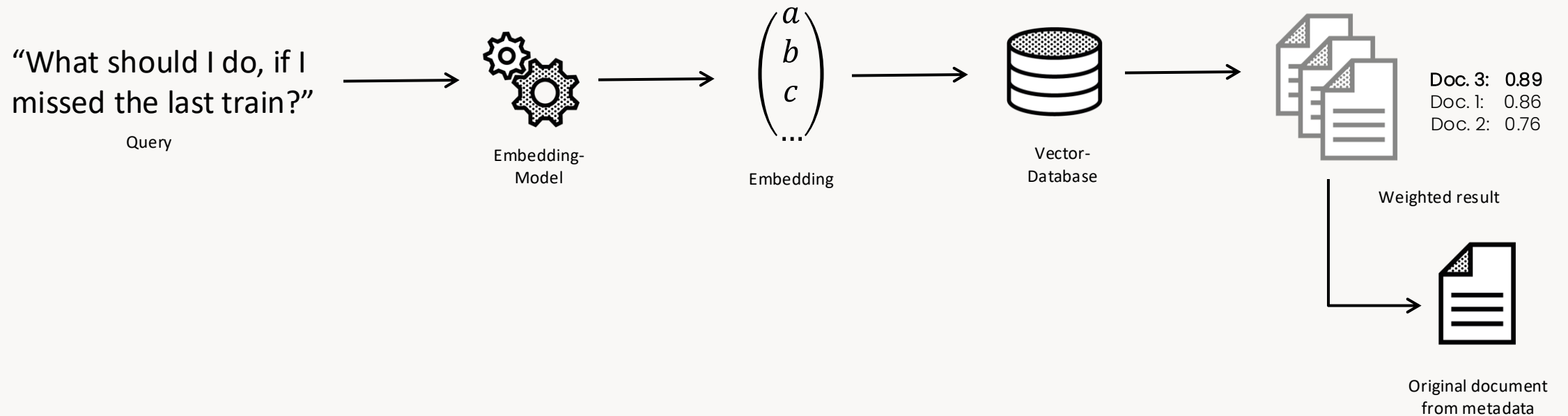
# Alternative Indexing

## HyQE: Hypothetical Question Embedding



# Alternative Indexing

- Retrieval



# DEMO

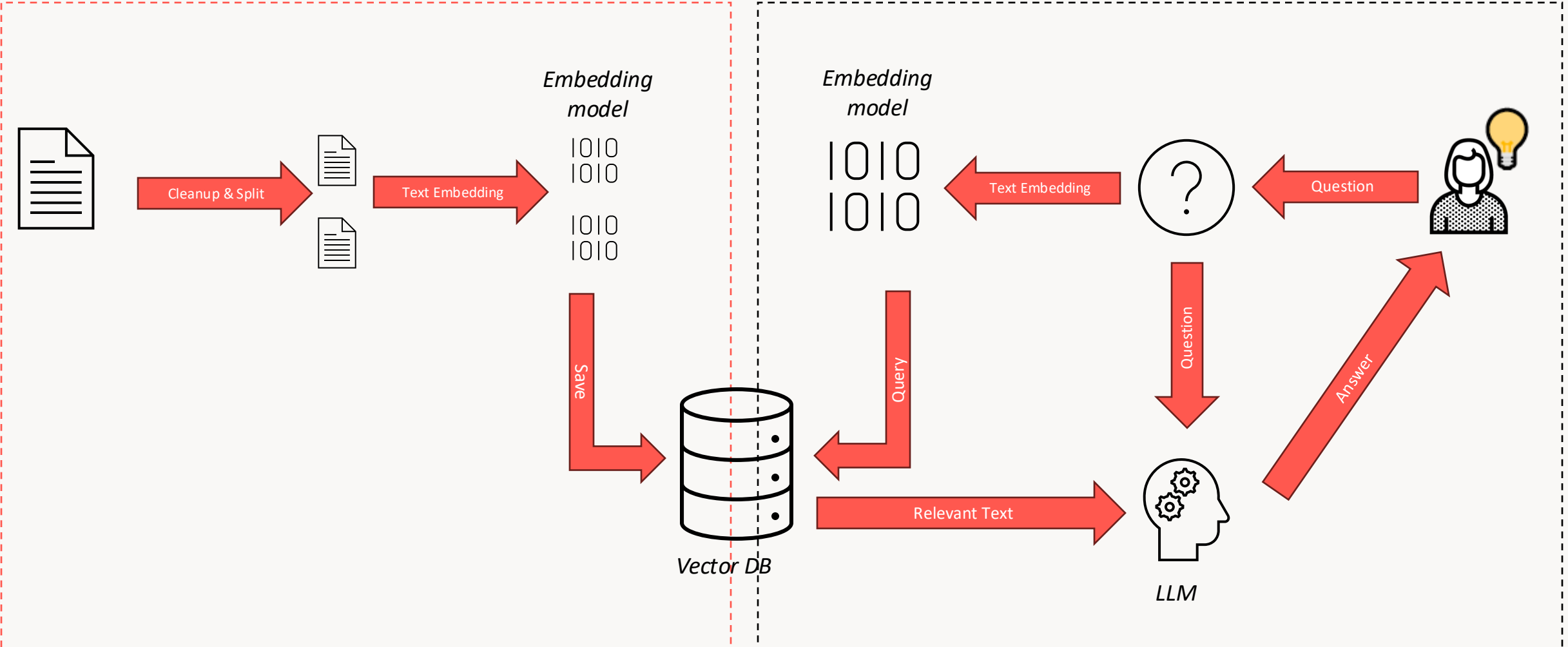
Compare embeddings

LangChain, Qdrant, OpenAI GPT

# Conclusion

# Retrieval-augmented generation (RAG)

## Indexing & (Semantic) search

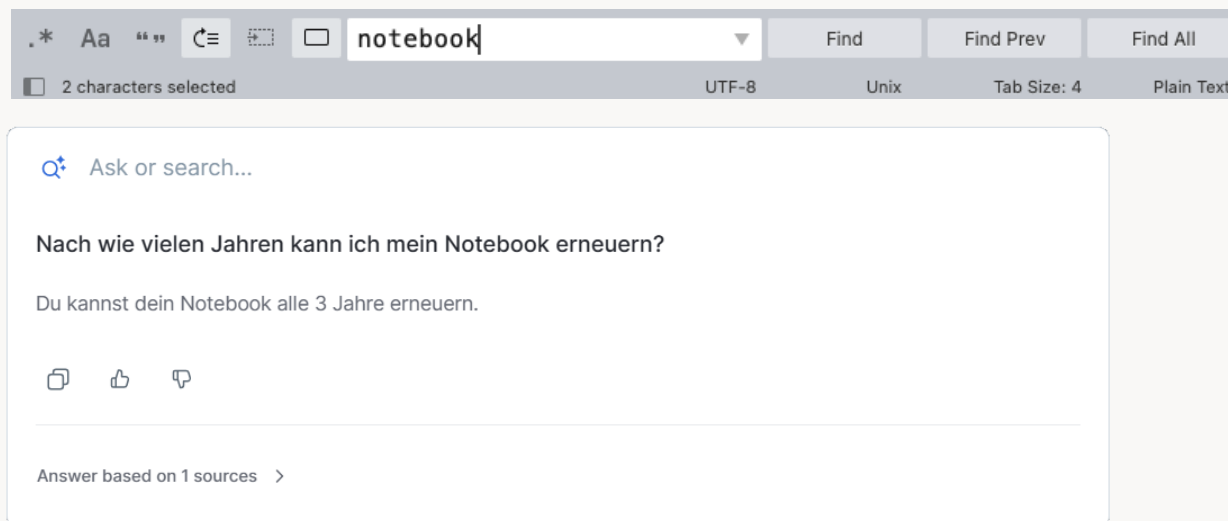


## Recap: Not good enough?

- 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

# 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!

think  
tecture

Demos:

<https://github.com/thinktecture-labs/infoday-genai-2024-talk-to-your-data>

Sebastian Gingter

<https://thinktecture.com/sebastian-gingter>

# “Talk to your Data”: Improving RAG solutions based on real-world experiences



## Slides & Code

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

Sebastian Gingter  
sebastian.gingter@thinktecture.com  
Developer Consultant

