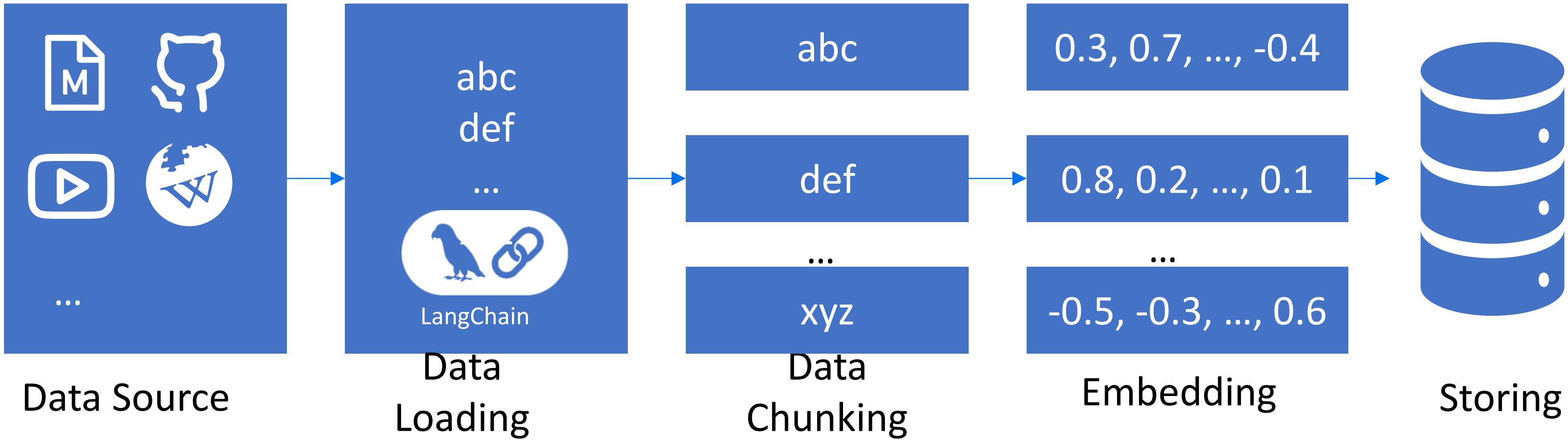
Vector Databases:

Data Ingestion Pipeline

Data Ingestion Pipeline:
Introduction

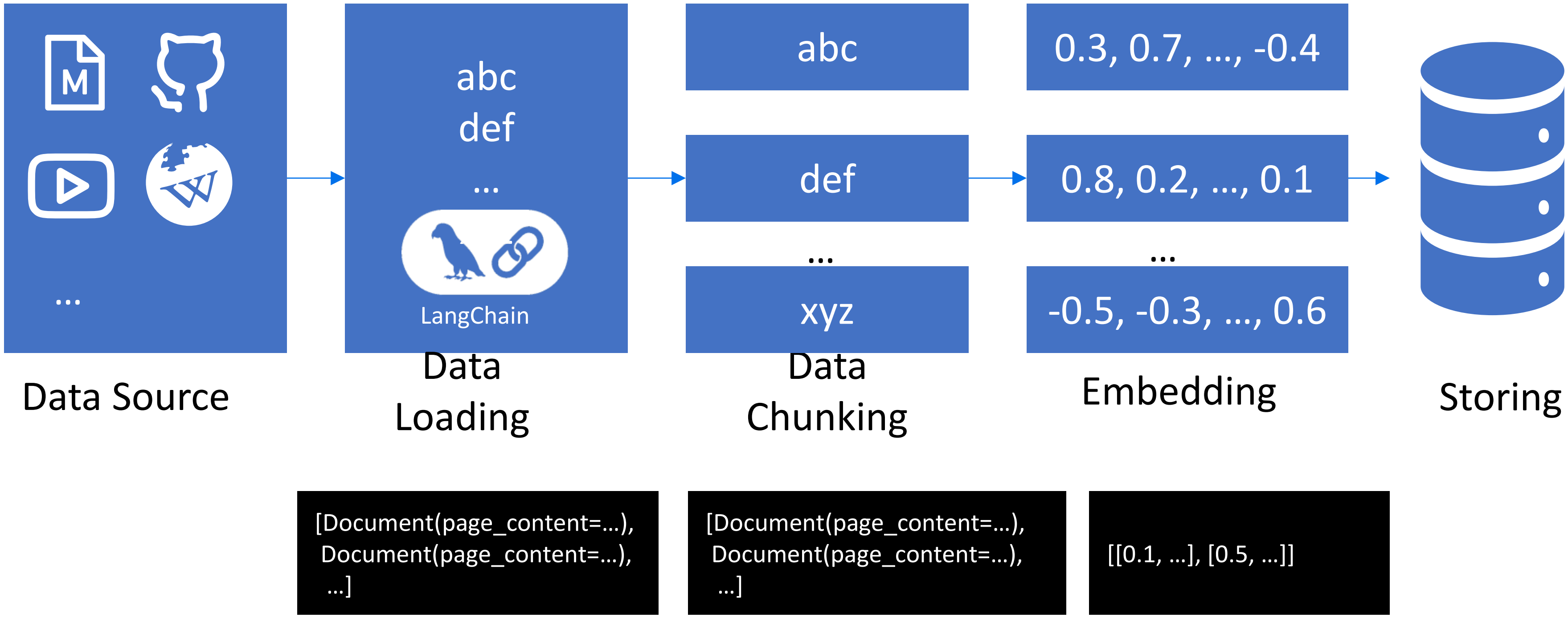
Vector Database

Introduction



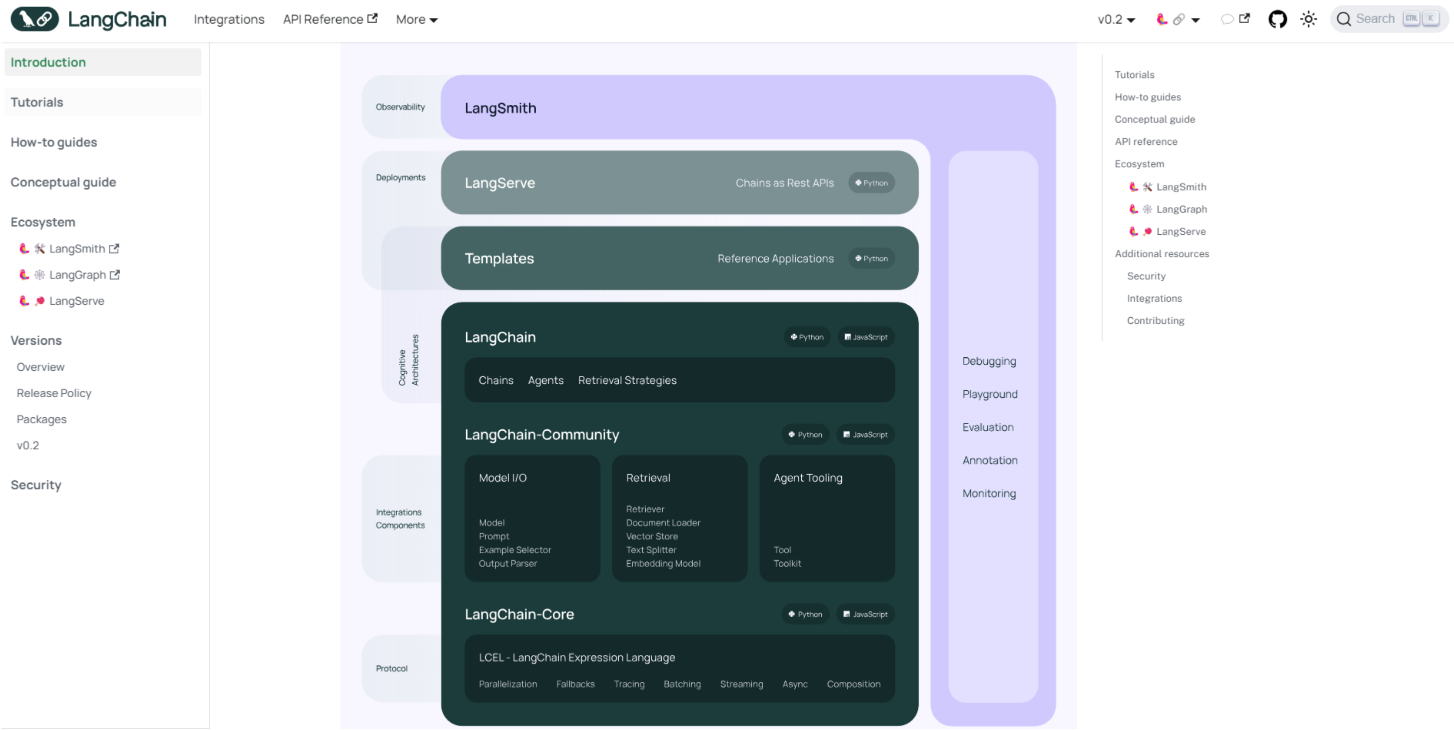
Vector Database

Data Types



Vector Database

Additional Resources

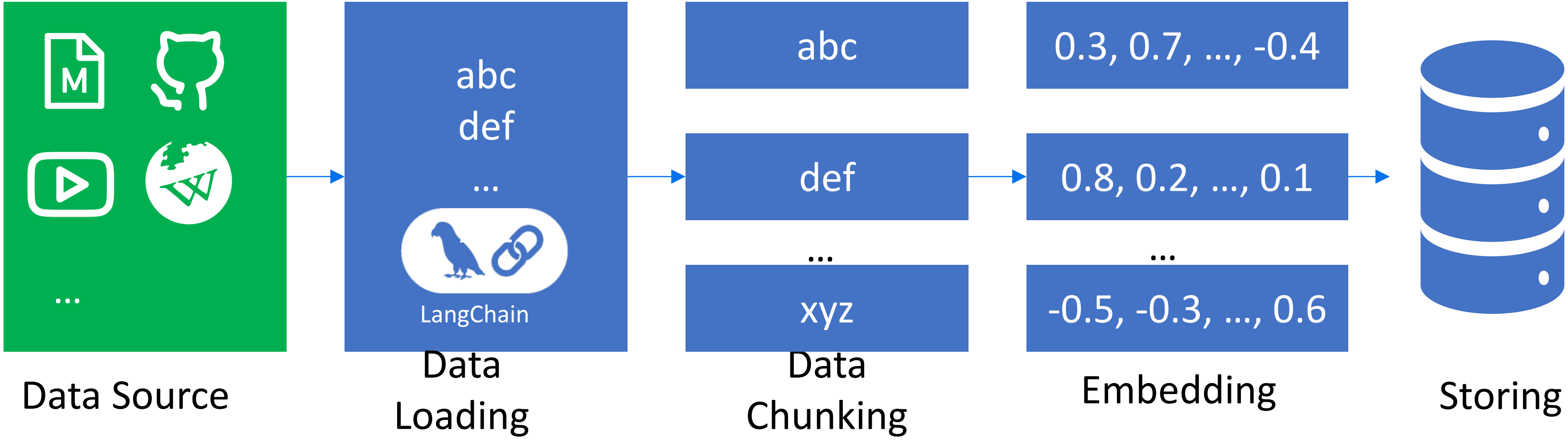


Source: https://python.langchain.com/

Data Ingestion Pipeline:
Data Source and -Loading

Vector Database

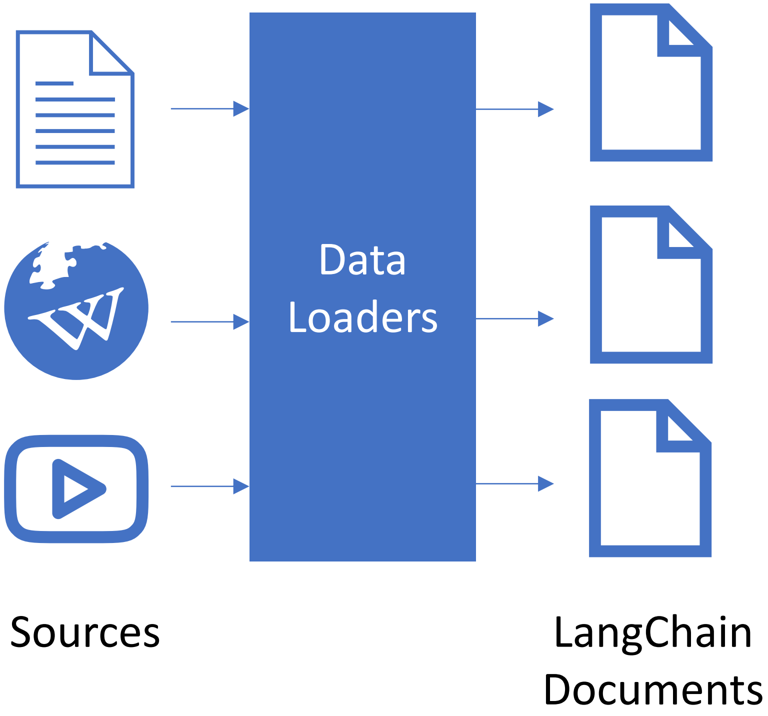
Data Source



Vector Database

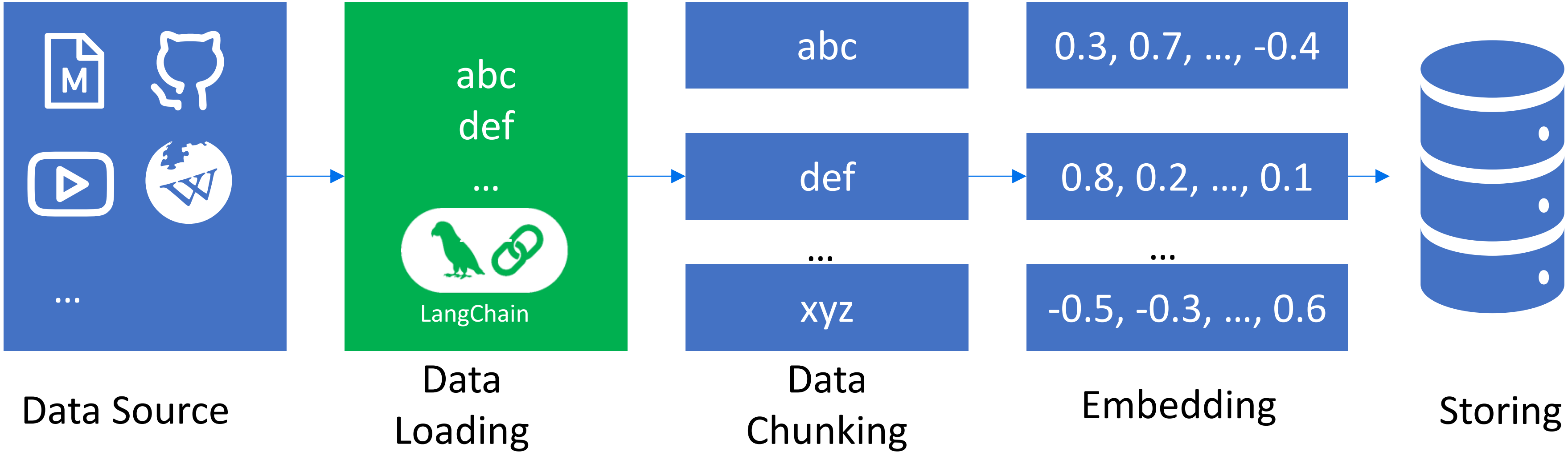
Data Loading

- Hundreds of different data sources are supported by LangChain
- DataLoader returns list of LangChain documents
- Documents have two attributes
  - Metadata
  - page_content



Vector Database

Data Loading

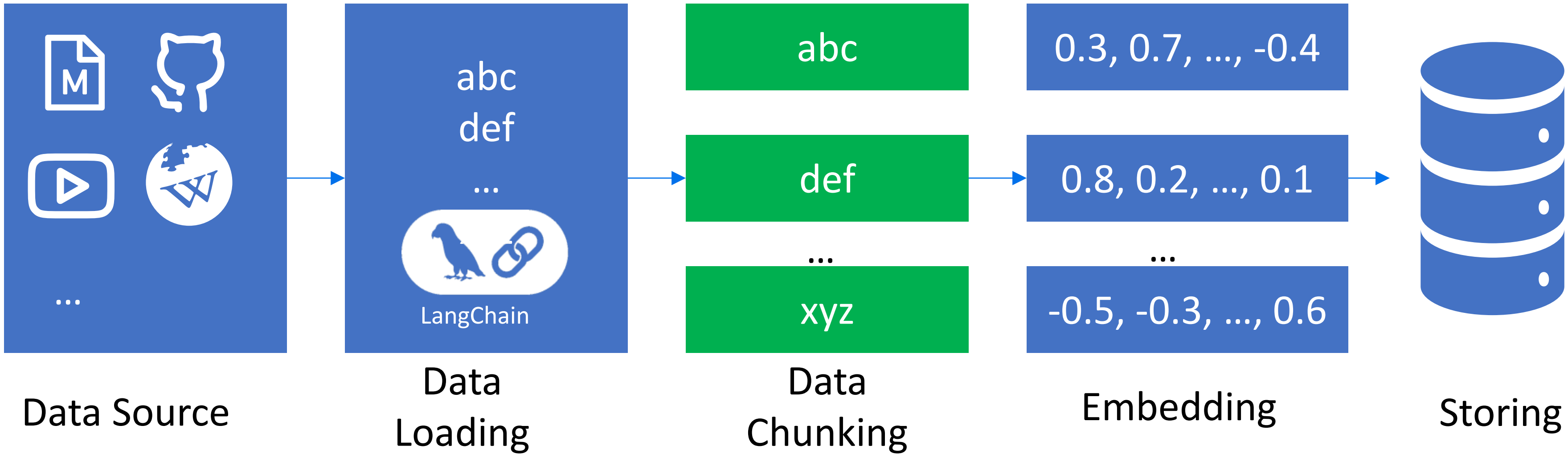


Data Ingestion Pipeline:

Data Chunking

Vector Database

Data Chunking

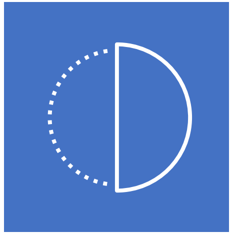


Vector Database

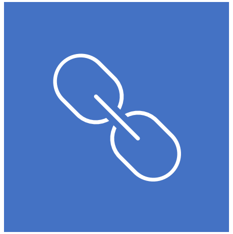
Data Chunking

What is Data Chunking?

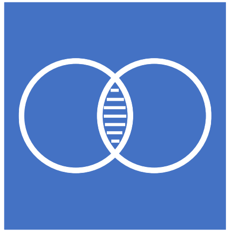
- Dividing larger pieces of information into smaller, manageable units
- These units called „chunks“
- Required to fit model context window
- Chunks should be:
  - Small
  - Semantically meaningful



Split



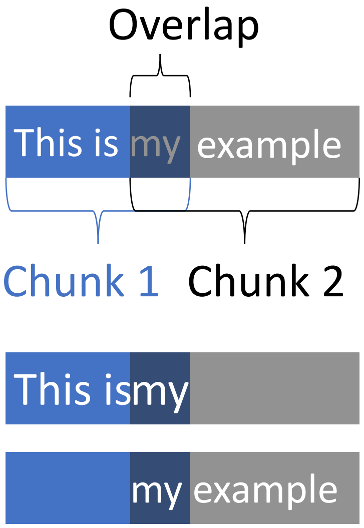
Combine

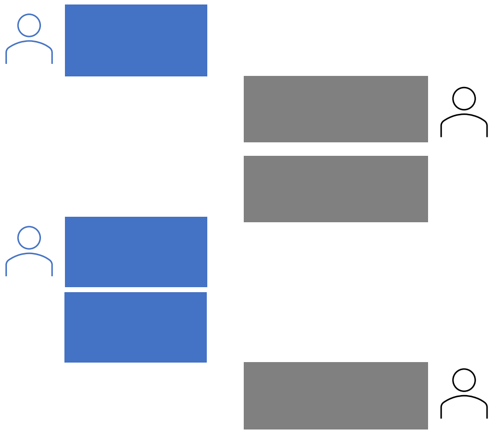


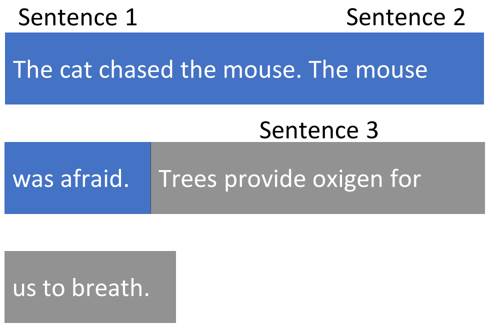
Overlap

Vector Database

Data Chunking: Chunking Approaches







- Sentence 1 and 2 are very similar → same chunk
- Sentence 3 different → new chunk

Fixed Chunk-Sizes
Identical
pre-defined

Structure-Based Chunk-Sizes
- e.g. chat messages should be consistent, no mix of users and chunks

Semantic Chunking
- based on semantic similarity
- e.g. when semantic break is observed

Vector Database

Data Chunking: Splitter Types

Text

JSON

HTML

Code

Vector Database

Data Chunking: Splitter Types

- chunk_size…defines maximum size of chunks [characters]
- chunk_overlap…possible overlap of max 5 characters

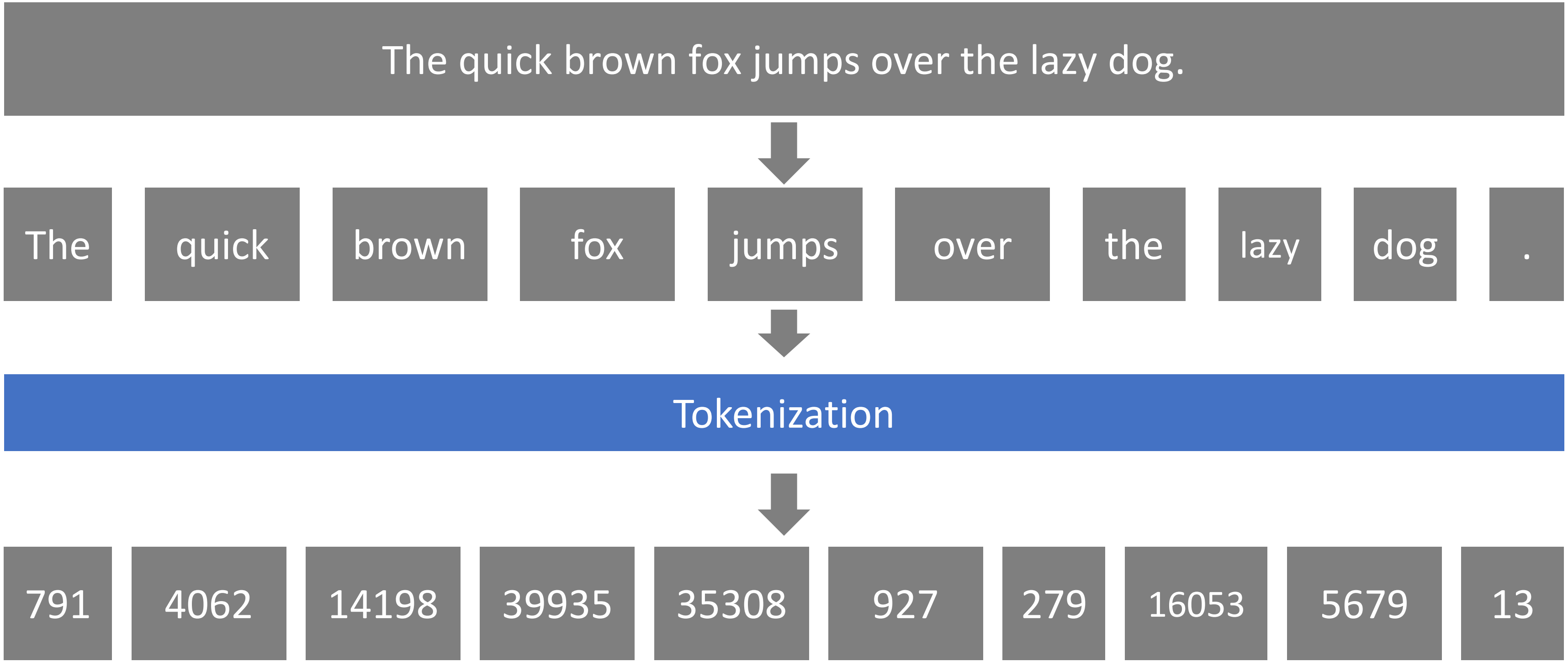
The quick brown fox jumps over the lazy dog.\n This is a simple example to show text splitting.\n.

```
RecursiveCharacterTextSplitter(
  chunk_size=20,
  chunk_overlap=5
  separators=[“\n”, “ “, “”]
)
```

- The quick brown fox
- brown fox jumps
- jumps over the lazy
- the lazy dog.
- This is a simple
- simple example to
- to show text
- text splitting.

Vector Database

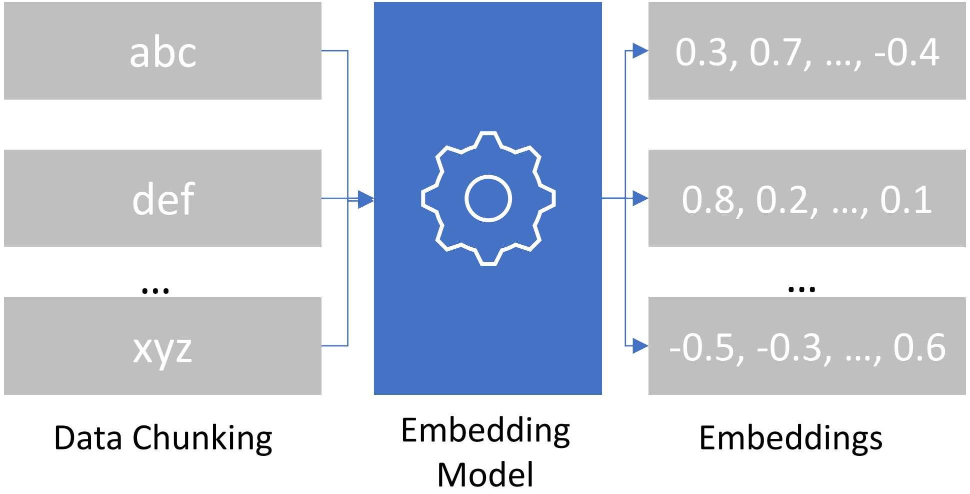
Data Chunking: Tokenization



Vector Database

Data Chunking: Context Window

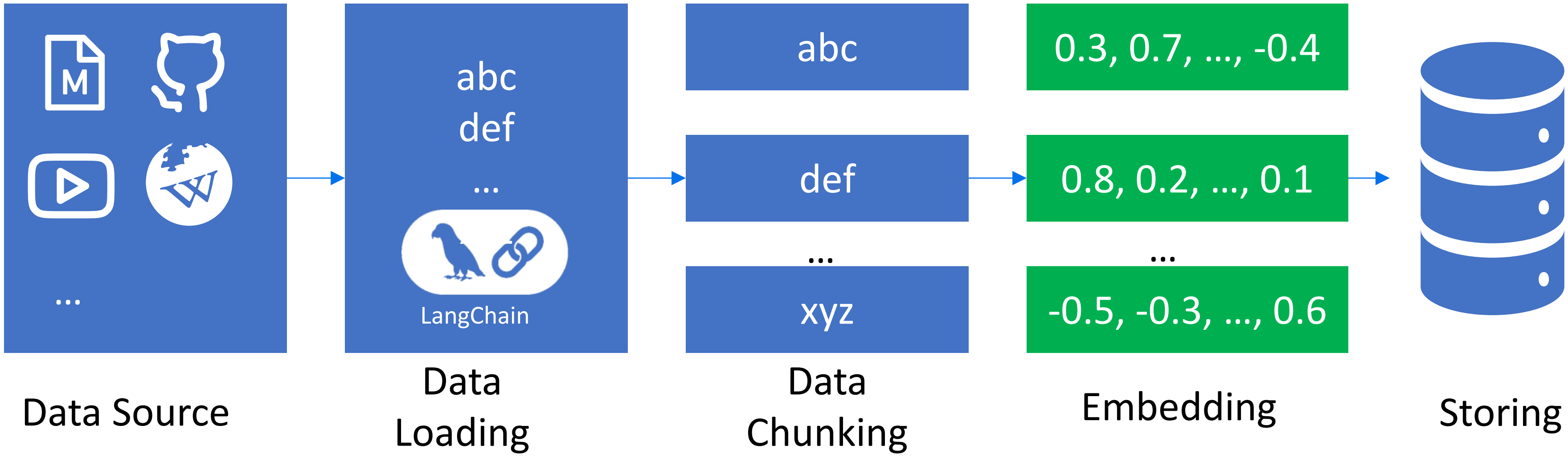
- Embedding model works with tokens, NOT words
- Model can cover only specific sequence lengths
- Too long text (longer than context window) will be truncated



Data Ingestion Pipeline: Embeddings

Vector Database

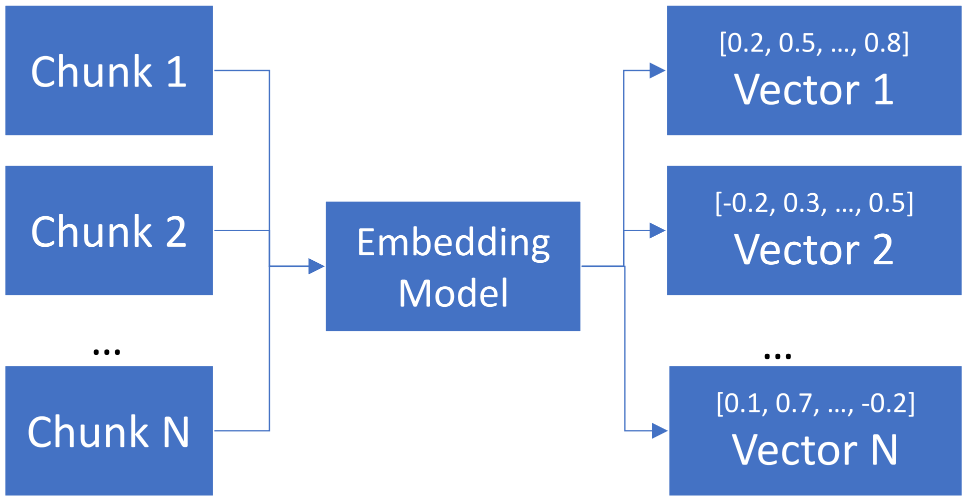
Embeddings: Introduction



Vector Database

Embeddings: What?

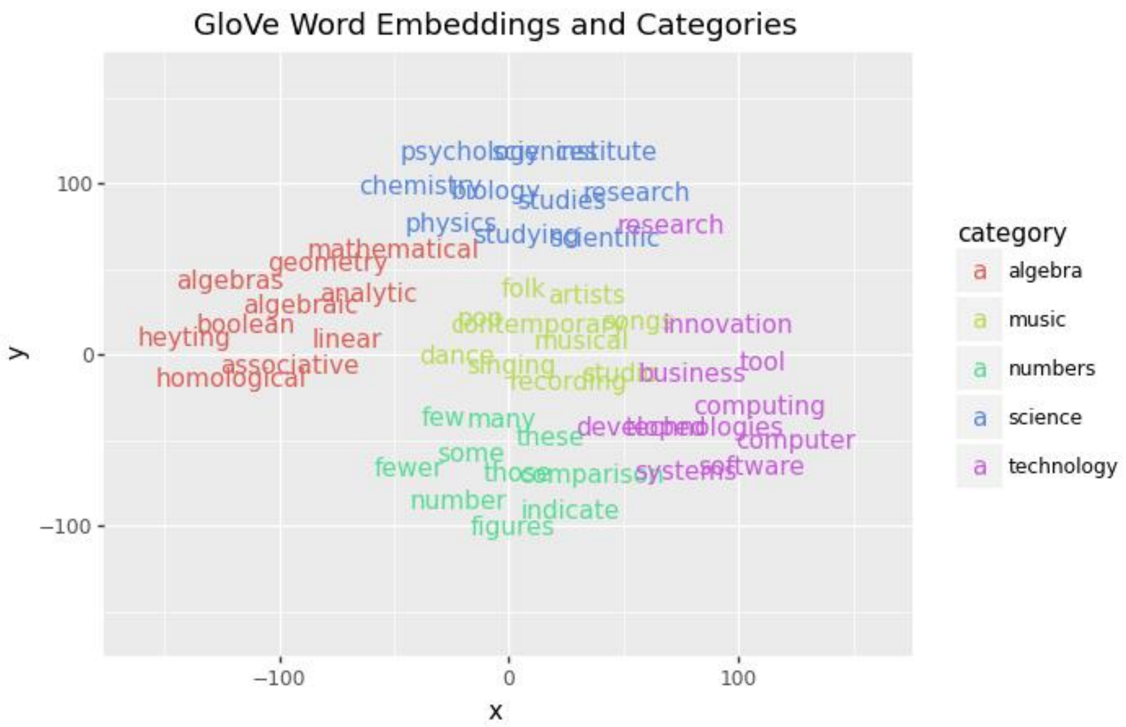
- Conversion of text data into numeric vectors
- Each word / sentence is represented as vectors
- Vector has „low“ number of dimensions



Vector Database

Word Embeddings: What is it?

- Convert words to numbers
- Representation of words as unique tensors in high-dimensional space
- Relationships to other words are captured
- Ideally similar words are close
- Usually Deep Learning applied to get embeddings
- Embeddings represent meaning

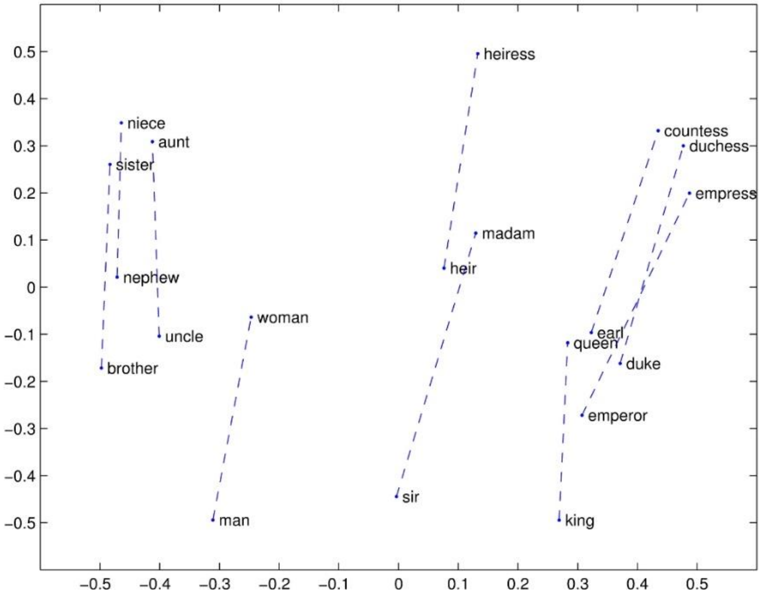


Word Embeddings represent words as **low-dimensional vectors** in mathematical space and **capture** their semantic and syntactic **meaning**.

Vector Database

Embeddings: Why?

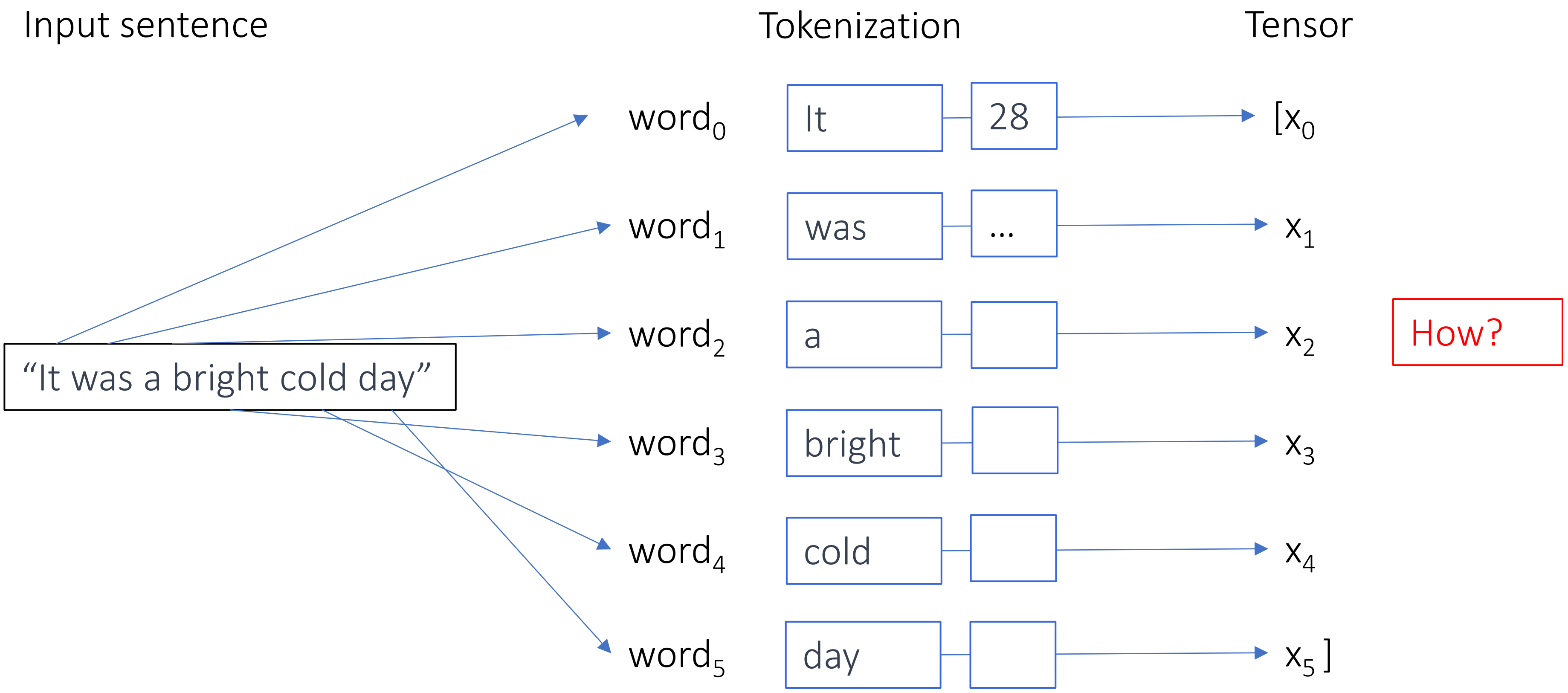
- Semantic representation
  - capture meaning of data
  - enable comparison and analysis
- Lower dimensionality
  - computational complexity is reduced
  - high-dimensional data can be represented in lower dimensions
- Reusability
  - usable across different applications



Source: https://nlp.stanford.edu/projects/glove/

Vector Database

From Words to Tensors



Vector Database

Word Embedding Approaches

- One-Hot Encoding
- Frequency-Based
- Neural Network

Vector Database

One-Hot Encoding

Index:	0	1	2	3	4	5
Word:	It	was	a	bright	cold	day

	0	1	2	3	4	5
It	1	0	0	0	0	0
was	0	1	0	0	0	0
a	0	0	1	0	0	0
bright	0	0	0	1	0	0
cold	0	0	0	0	1	0
day	0	0	0	0	0	1

Vector Database

One-Hot Encoding - Problems

Problems

- Curse of dimensionality → memory issues
- Matrix very sparse

- Words are isolated from each other

- All words have the same distance to each other

Vector Database

Word Embedding Approaches

One-Hot Encoding

Frequency-Based

Neural Network

Count

- Very similar to OHE
- Gets count of words in document

TF-IDF

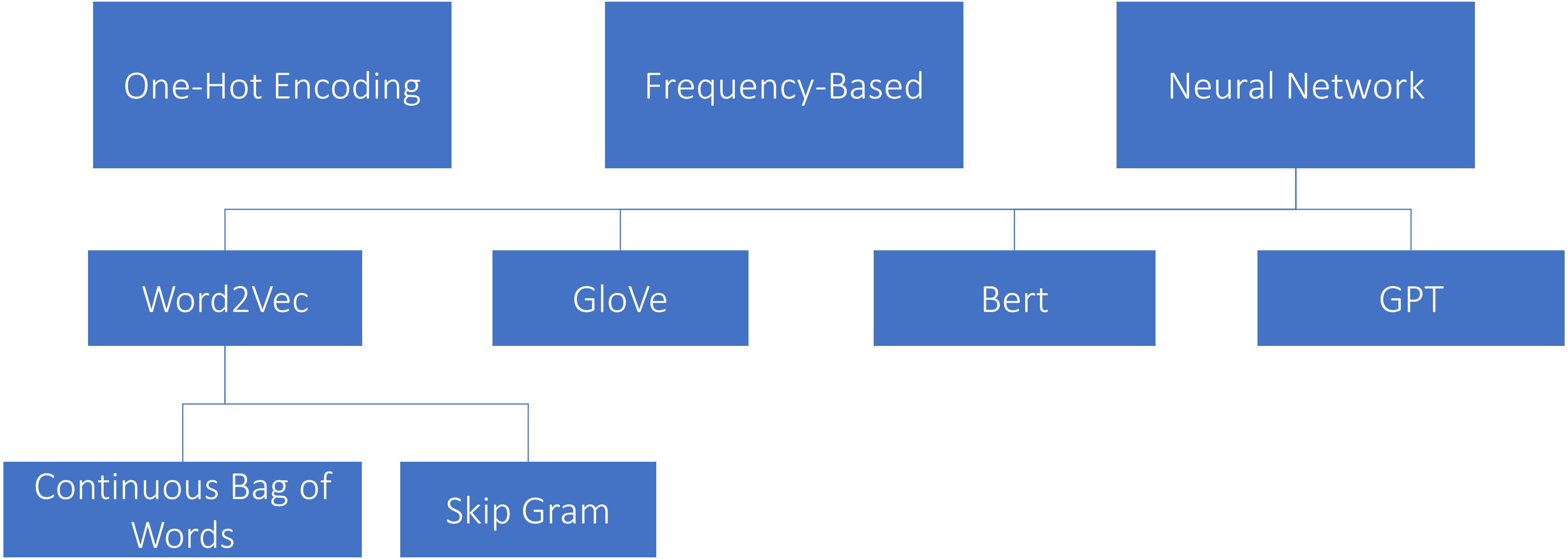
- Term-Frequency/Inverse Term Freq.
- Gets count of words in document AND corpus
- Words frequent in a doc → important
- Words frequent in corpus → not important

Co-Occurrence

- Gets similarity of words

Vector Database

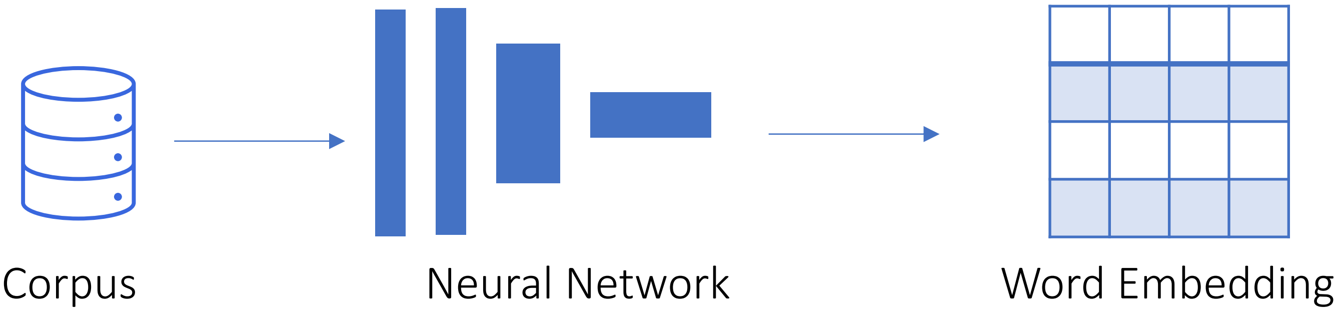
Word Embedding Approaches



Vector Database

Neural Network based Embeddings

- Aim to
    - Capture context / meaning
    - Capture similarity to other words
    - Reduce dimension
    - Avoid memory issues
- Developed based on Neural Networks



Vector Database

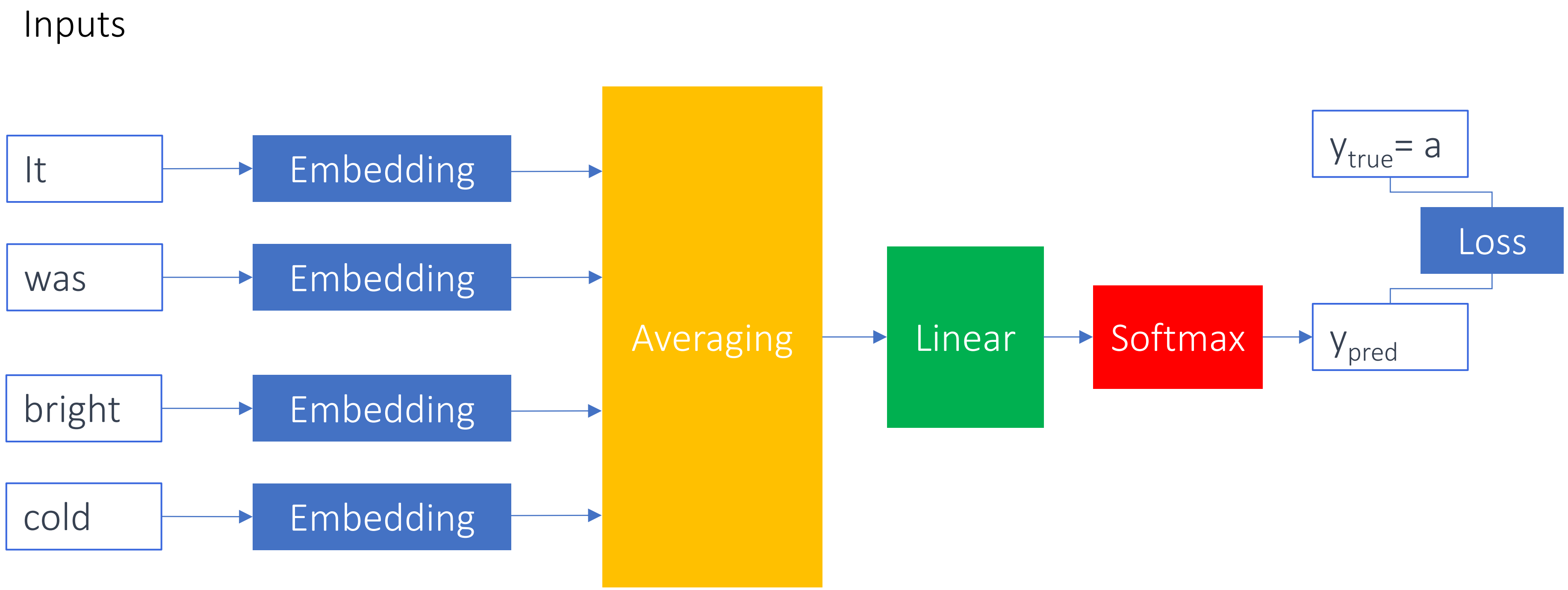
Word2Vec: Continuous Bag of Words

$\underbrace{\text{word}_{n-2} \quad \text{word}_{n-1}}_{\text{Independent features}} \quad \underbrace{\text{word}_{n}}_{\text{dependent feature}} \quad \underbrace{\text{word}_{n+1} \quad \text{word}_{n+2}}_{\text{Independent features}}$

Independent Features	Dependent Feature“
[“It”, “was”, “bright”, “cold”]	“a”
[“was“, “a”, “cold”, “day”]	„bright“
…	

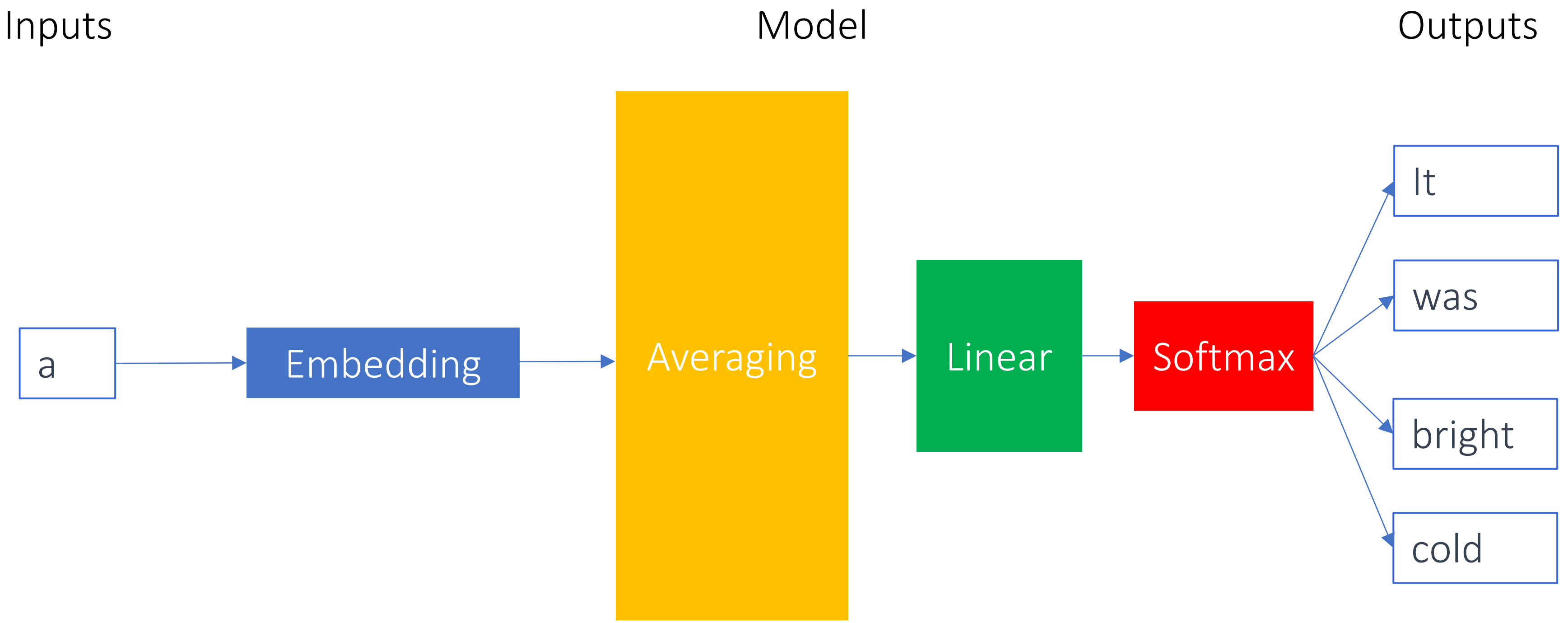
Vector Database

Word2Vec: Continuous Bag of Words Model



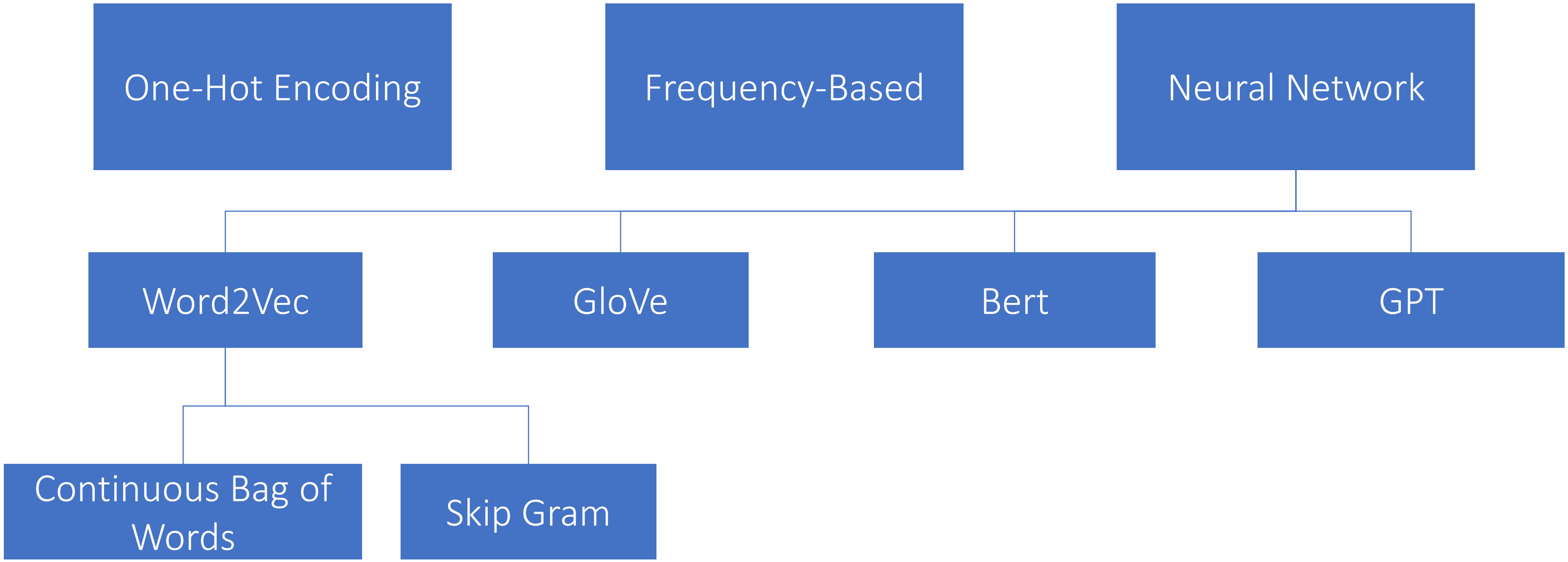
Vector Database

Word2Vec: Skip Gram



Vector Database

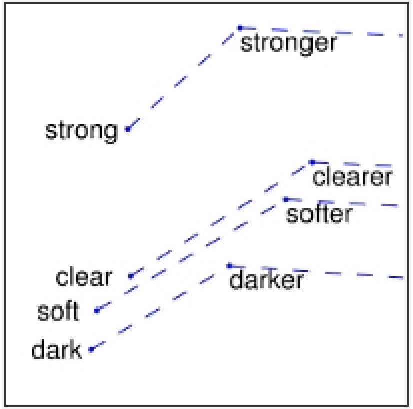
Word Embedding Approaches



Vector Database

GloVe

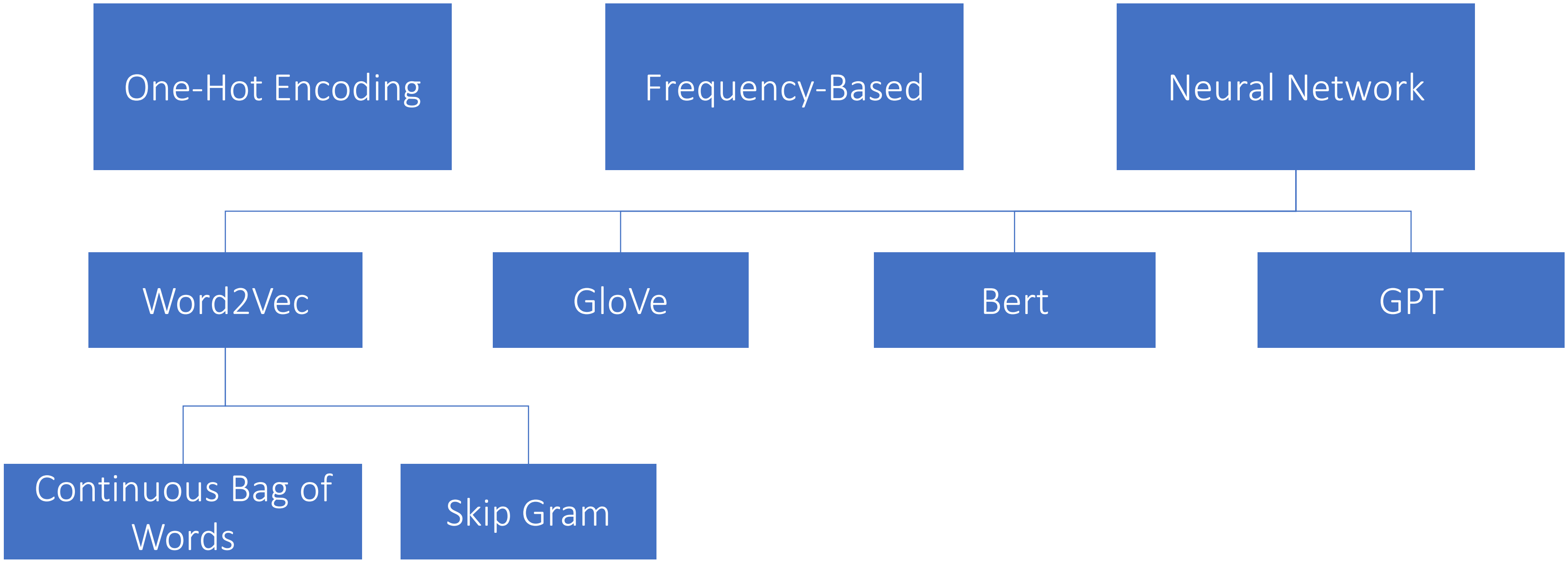
- Global Vectors for Word Representations
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. [GloVe: Global Vectors for Word Representation]
- based on co-occurrence matrix of words in a corpus, which counts how often words appear together in the same context.
- constructs a matrix of word co-occurrence counts and then factorizes this matrix to obtain word embeddings
- factorization based on singular value decomposition (SVD)
- resulting embeddings are dense, low-dimensional vectors
- Encode words as vector of other words



Source: https://nlp.stanford.edu/projects/glove/

Vector Database

Word Embedding Approaches



Vector Database

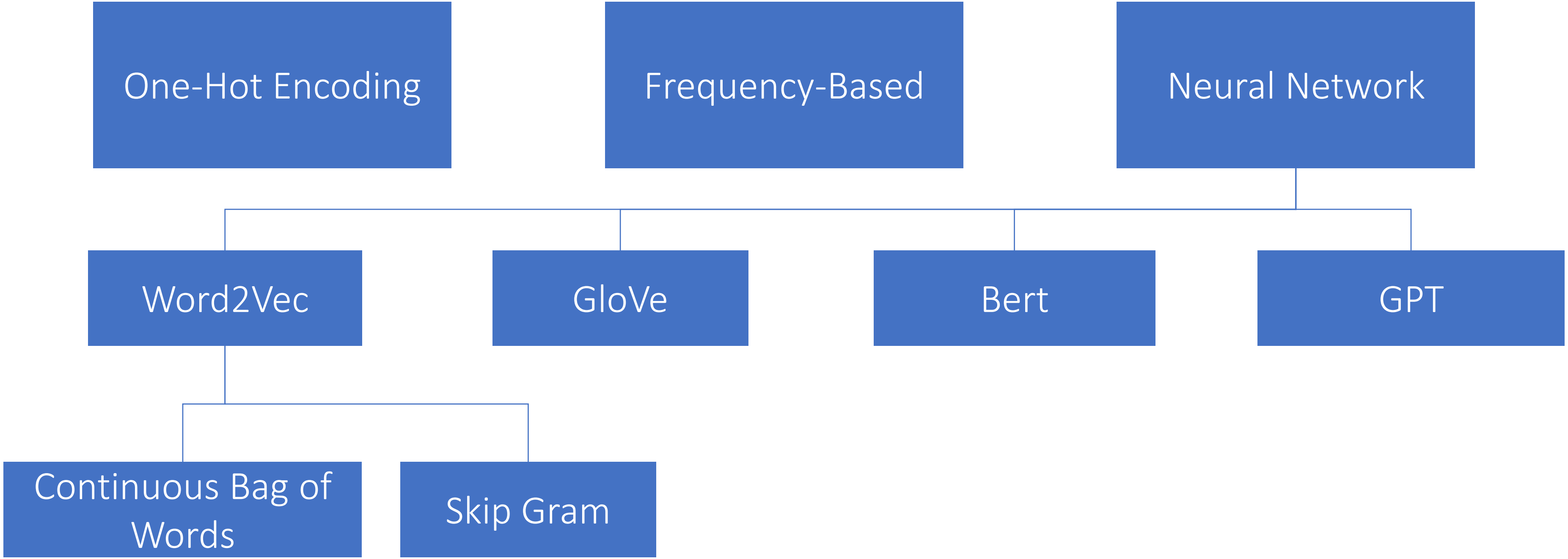
BERT

- **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
- Developed by Google in 2018
- Pre-trained word embedding
- Based on Transformers
- Applies „masked language modeling“ – masking some words in sentence and learn to predict them
- Applies „next sentence prediction“ – model predicts whether two sentences are similar in a text

- Original variants: BERT-base (110m parameters, 440MB) and BERT-large (340m parameters, 1.3GB)
- Other variants: RoBERTa, ALBERT, ELECTRA, …

Vector Database

Word Embedding Approaches



Vector Database

GPT

- **G**enerative **P**re-trained **T**ransformers
- Developed by OpenAI
- Not strictly a word embedding, but contextualized word embedding
- Unique embedding for each occurrence of a word based on surrounding words in text
- Applies Transformer architecture
- GPT-3 has 175 billion parameters

Vector Database

Difference Embedding Model vs. Large-Language Model

Parameter	Embedding Model	LLM
Base architecture	transformers	transformers
Process	texts, words, … → numerical vectors	predict next words
Target	find semantic similarities of texts	generate outputs depending on context, e.g. next words
Applications	semantic search, clustering, representations for ML	text generation, QA systems, chatbots, translations, code generation

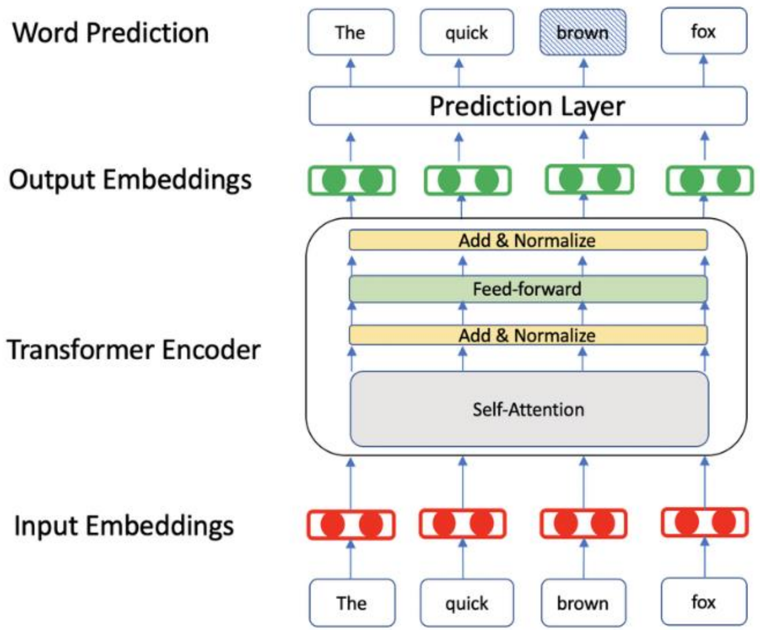
Vector Database

Difference Embedding Model vs. Large-Language Model

Parameter	Embedding Model	LLM, LMM
Inputs	Text, words, sentences, images, …	Text, words, sentences, images, …
Output	vector	human-readable text/code
Focus	representation of data	processing and generation of data
Model Size	smaller, more specific (narrow AI, e.g. sentence transformers)	larger, e.g. GTP, Llama, …
based on	pre-trained language models, uses architecture only for vector creation	uses transformers

Vector Database

Difference Embedding Model vs. Large-Language Model



BERT embedding model

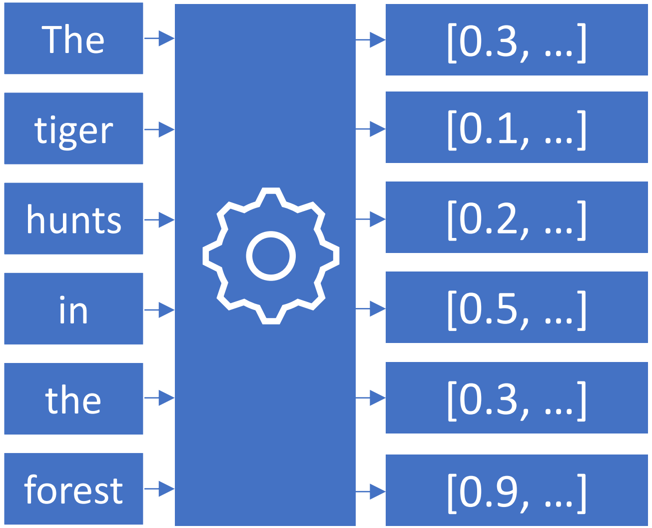
BERT model for fill-mask

Source: https://www.researchgate.net/figure/An-illustration-of-the-BERT-model-The-model-is-predicting-the-masked-word-brown_fig5_347822270

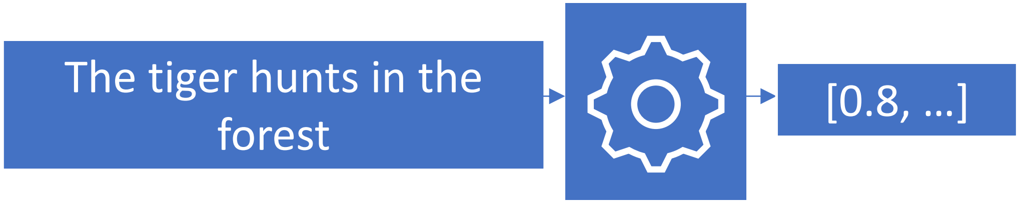
Vector Database

Embeddings: How?

Word Embeddings



Sentence Embeddings



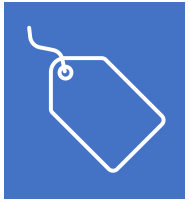
Vector Database

Embeddings: Which types are available?

Type	Model	Provider	Price	Vector Size
Online	text-embedding-3-small	OpenAI	0.02$ / 1M tokens	1536
Online	text-embedding-3-large	OpenAI	0.13$ / 1M tokens	3072
Online	mistral-embed	MistralAI	0.10$ /1M tokens	1024
Offline	all-MiniLM-L6-v2	Open Source	---	384
Benchmark: https://huggingface.co/spaces/mteb/leaderboard ...				

Vector Database

Embeddings: Factors to consider



Price

Speed

Off-/Online

Benchmark Performance

Vector Database

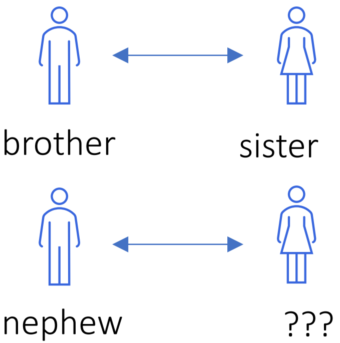
Coding: Embedding GloVe closest words

- Find closest words

```
get_closest_words_from_word('chess')
✓ 7.5s

[('chess', 0.0),
 ('backgammon', 4.379469394683838),
 ('grandmasters', 4.5636835098266),
 ('grandmaster', 4.613785743713379),
 ('scrabble', 4.677640438079834)]
```

- Find word analogies



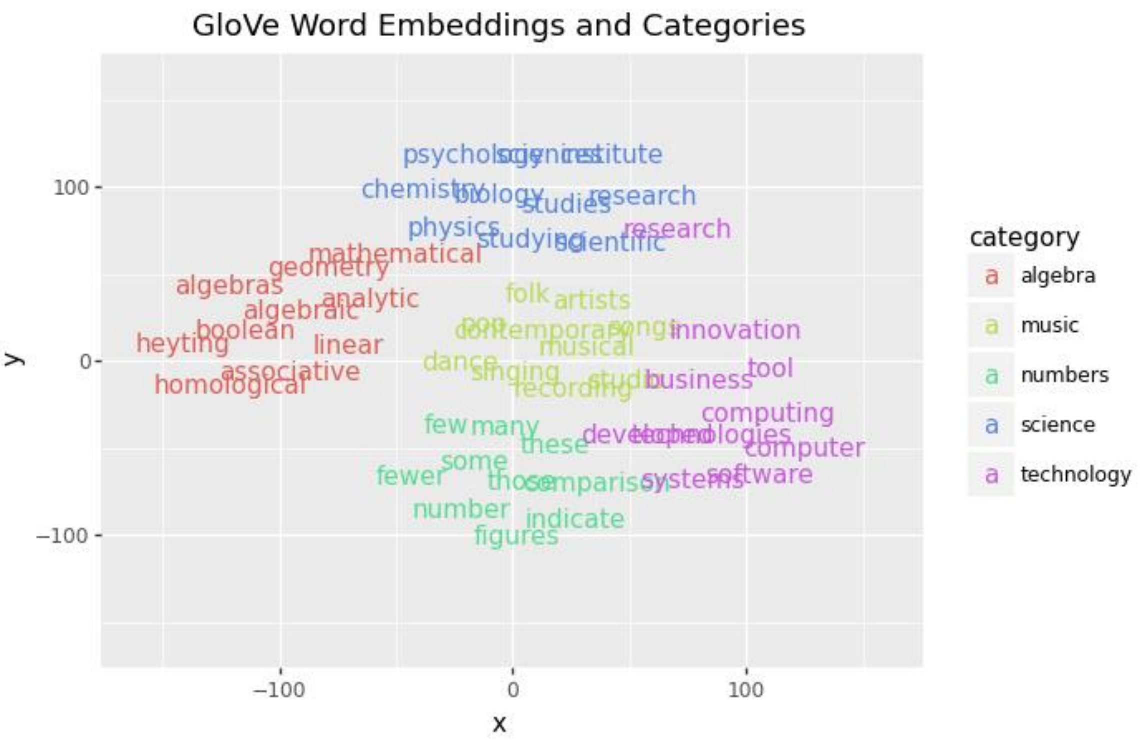

```
analogy = get_word_analogy(word1='sister',
                           word2='brother',
                           word3='nephew')
analogy
✓ 7.3s

'niece'
```

Vector Database

Coding: Word Cluster

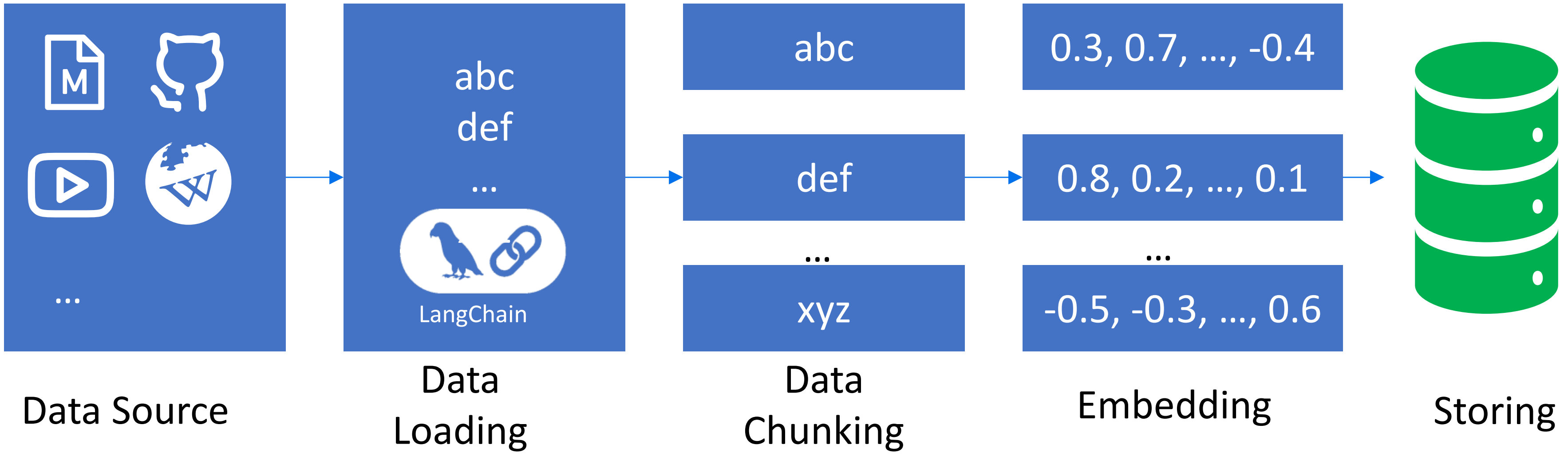
- Given some categories
- Find words for the categories
- Check if they are „close“ (similar)



Data Ingestion Pipeline:
Data Storing

Vector Database

Data Storing



Vector Database

Data Storing: What is a vector database?

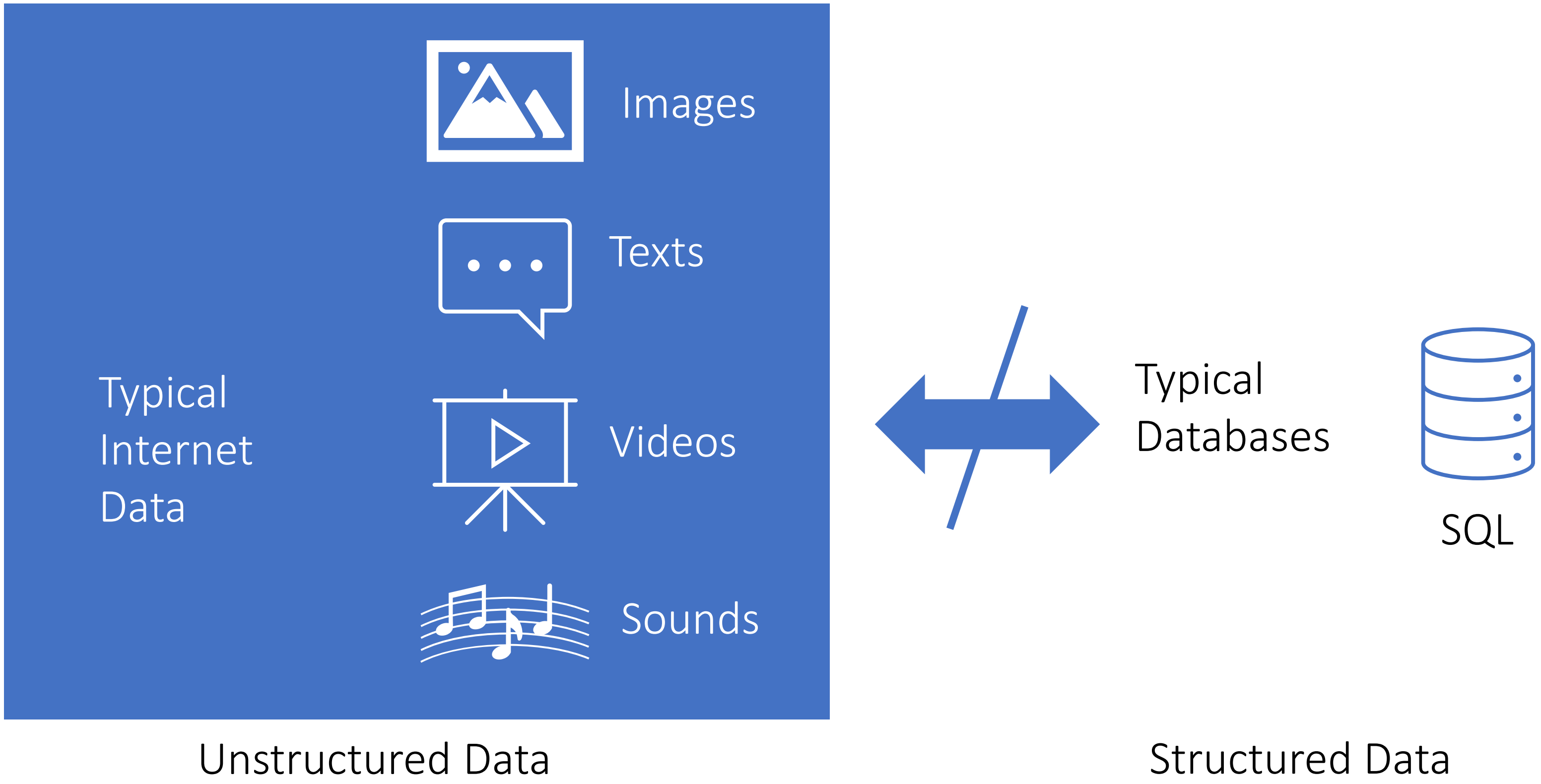
A vector database stores high-dimensional data (embeddings) for fast querying and similarity analysis.

Features

- Special type of database
- Allows to store, manage, and query data which is represented in geometric formats
- Enables similarity search, clustering, real-time analytics

Vector Database

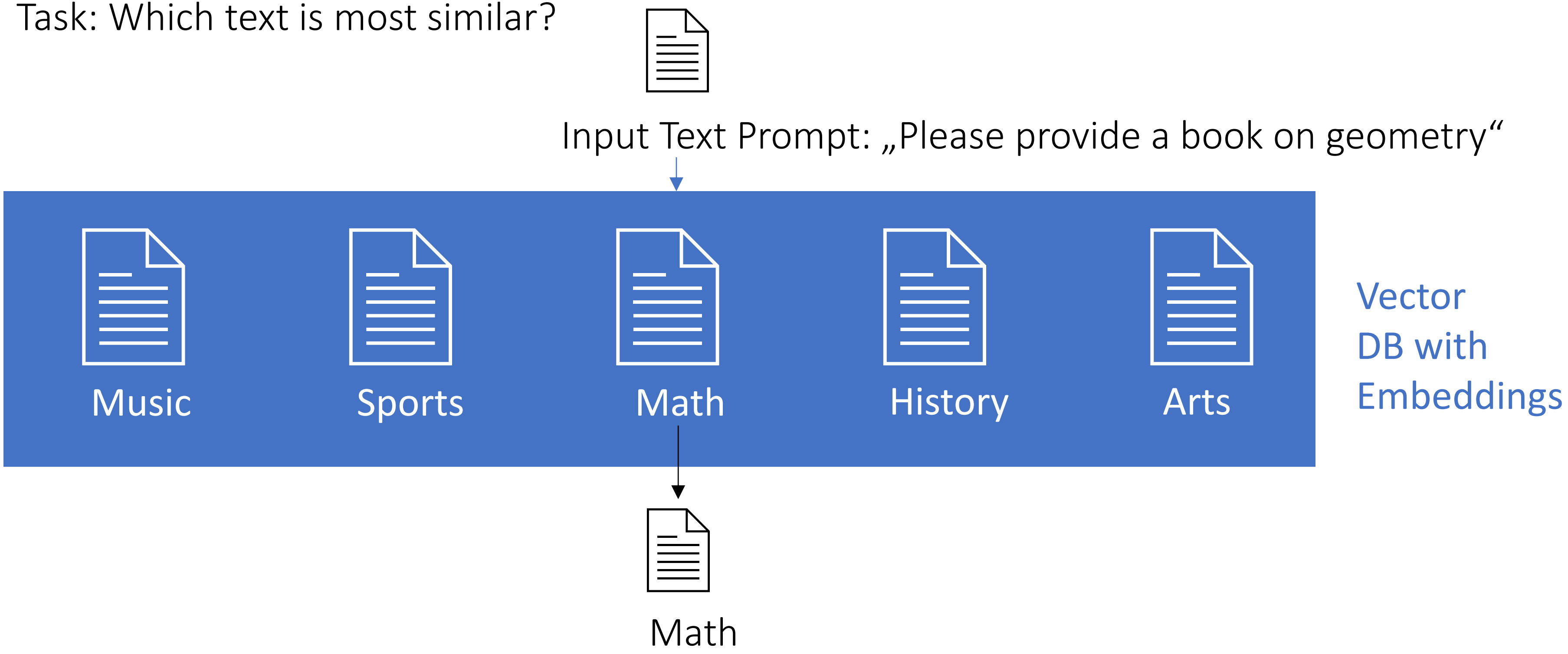
Data Storing: Why is a vector database needed?



Vector Database

Data Storing: Text Querying

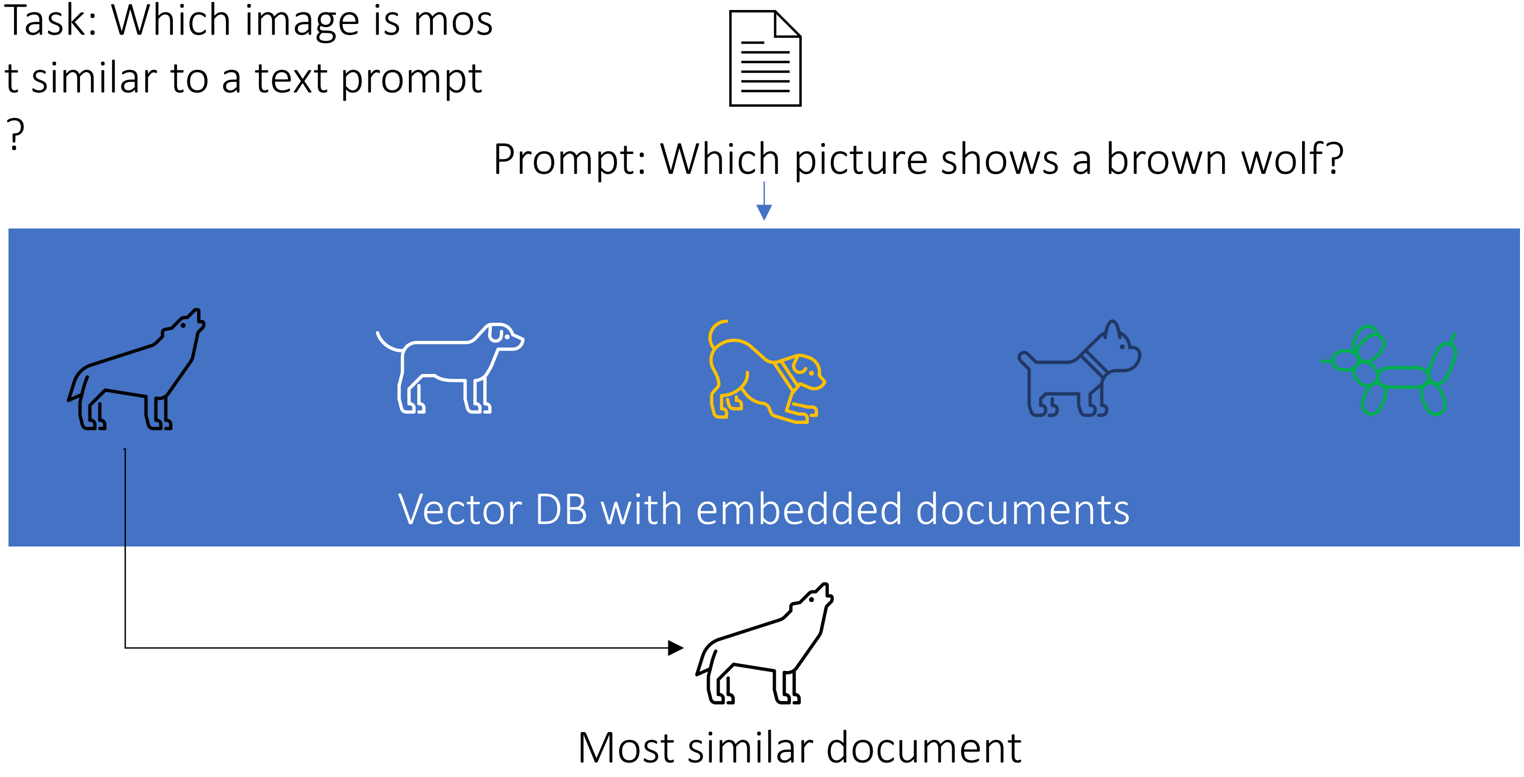
Task: Which text is most similar?



Vector Database

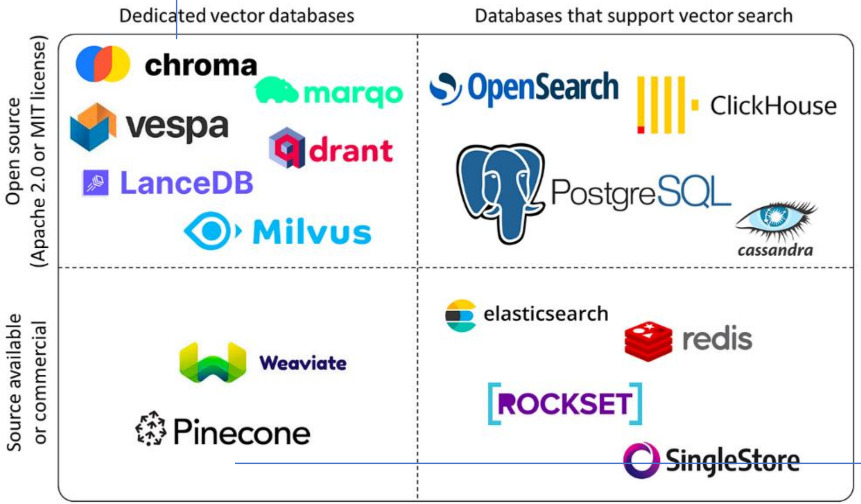
Data Storing: Image Querying

Task: Which image is most similar to a text prompt?



Vector Database

Data Storing: Vector DB Providers



Source: https://www.datacamp.com/blog/the-top-5-vector-databases

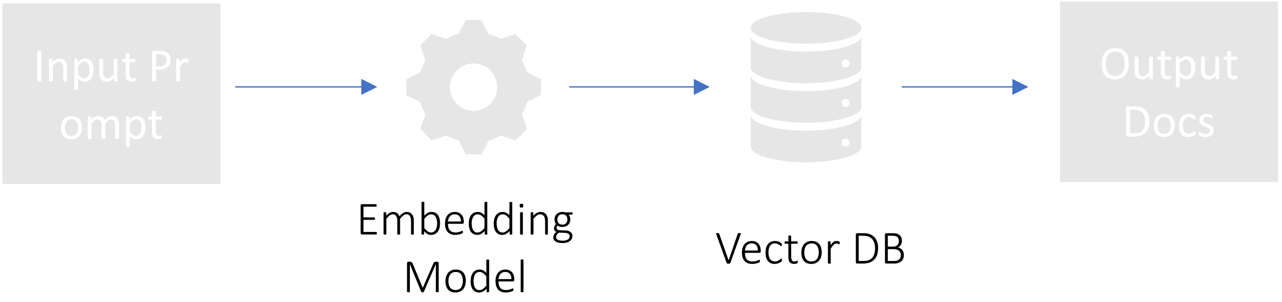
Pinecone

Pinecone	Chroma
pinecone.io	trychroma.com
web-based	local
freemium	free

Data Ingestion Pipeline:
Data Querying

Vector Database

Data Querying: Text Querying

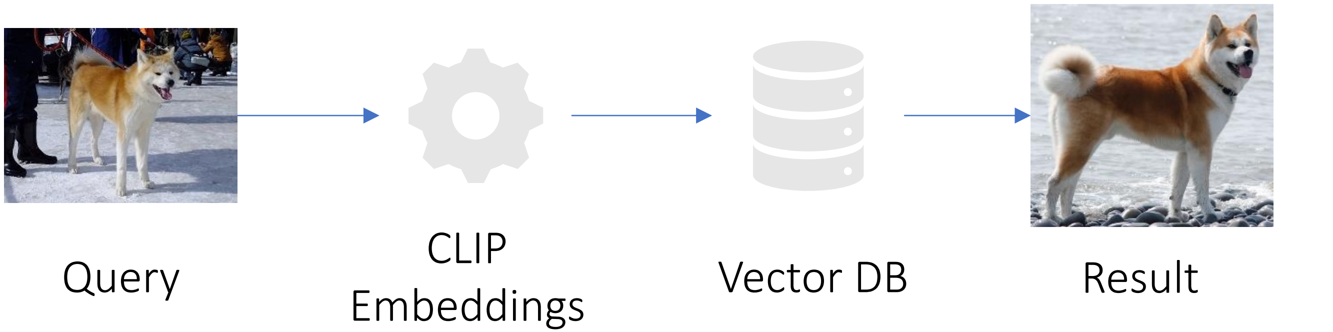


Practical Implementation

```
collection.query(query_texts=[“This is my input text”])
```

Vector Database

Data Querying: Image Querying




```
query_list = ["../data/dogs/akita_1.jpg"]
query_result = chroma_collection.query(
         query_images = query_list,
         n_results=3,
         include=['documents’,
                  'distances’,
                  'metadatas', 'data’,
                  'uris'],)
```

```
Result 1: ../data/dogs/akita_3.jpg
          with distance: 0.17
```

Vector Database

Data Querying: Image Querying 2

„dog in grassland“

Query

CLIP Embeddings

Vector DB

Result

```
query_list = [“dog in grassland"]
query_result = chroma_collection.query(
         query_texts = query_list,
         n_results=3,
         include=['documents’,
                  'distances’,
                  'metadatas', 'data’,
                  'uris'],)
```

```
Query: dog in grassland Result 0:
../data/dogs/mastiff_1.jpg
with distance: 0.85
```

Data Ingestion Pipeline:
Similarity Search

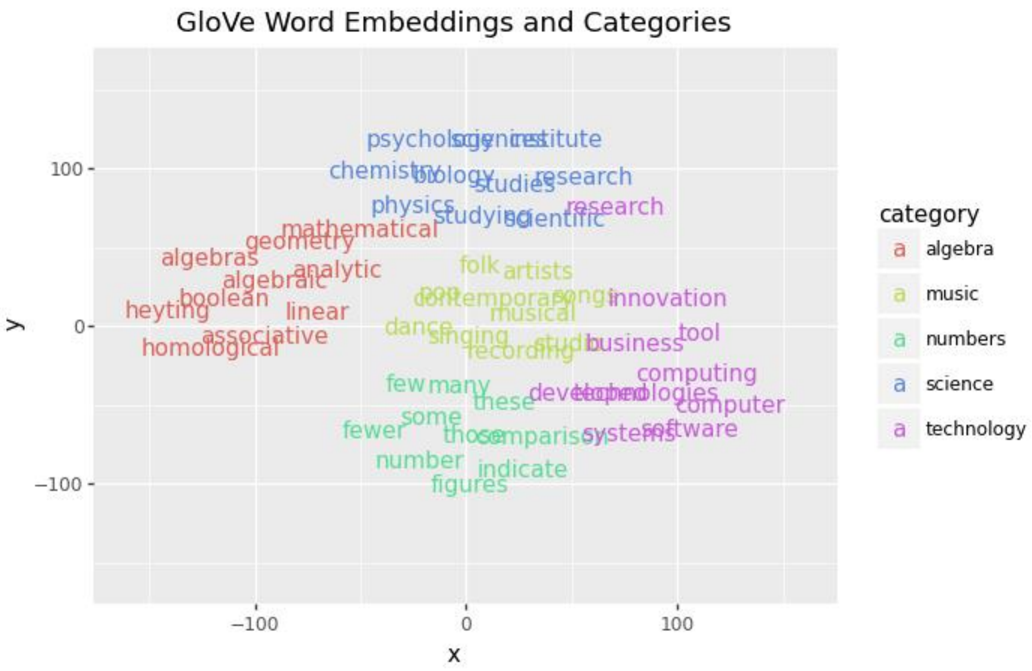
Vector Database

Similarity Search

- Vector DB needs to analyze similarity of query-embedding compared to document embeddings.
- Approaches:
  - Cosine Similarity
  - Maximum Margin Relevance

Vector Database

Similarity Search



Example: word embeddings reduced to 2 dimensions

$$dist = \sqrt{\underbrace{(x_1 - y_1)^2} + (x_n - y_n)^2}$$

For an embedding vector of 768 embeddings, there are 768 dist ance terms

Vector Database

Similarity Search

Imagename	Embedding					
dog1	0.3	0.02	0.8	0.6	...	0.4
dog2	0.1	0.52	0.7	0.6	...	0.4
...						
dogN	0.3	0.62	0.9	0.2	...	0.3

Vector Database

dogTest

0.3	0.02	0.8	0.6	...	0.4

$$dist = \sqrt{\sum (x_i - y_i)^2}$$

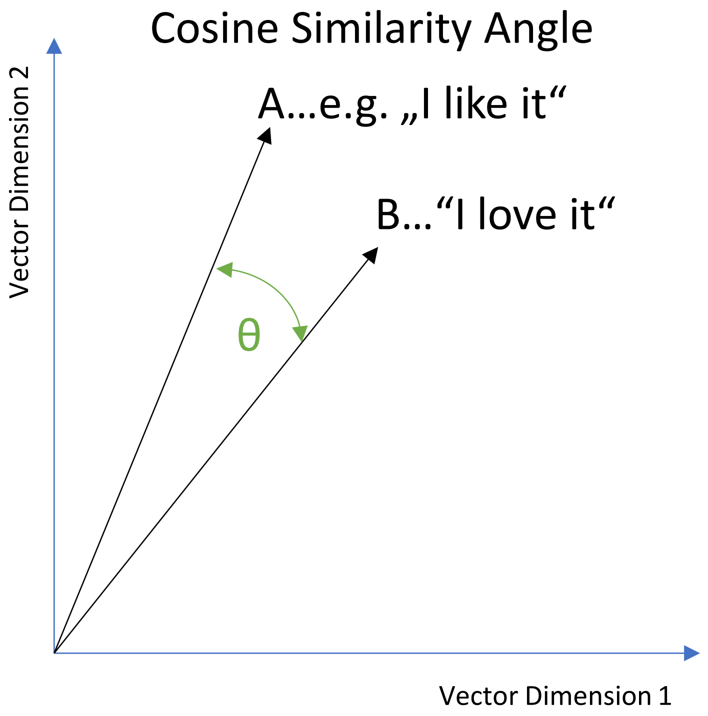
For small data, go with np.array().

For large dataset not feasible!

Vector Database

Similarity Search: Cosine Similarity

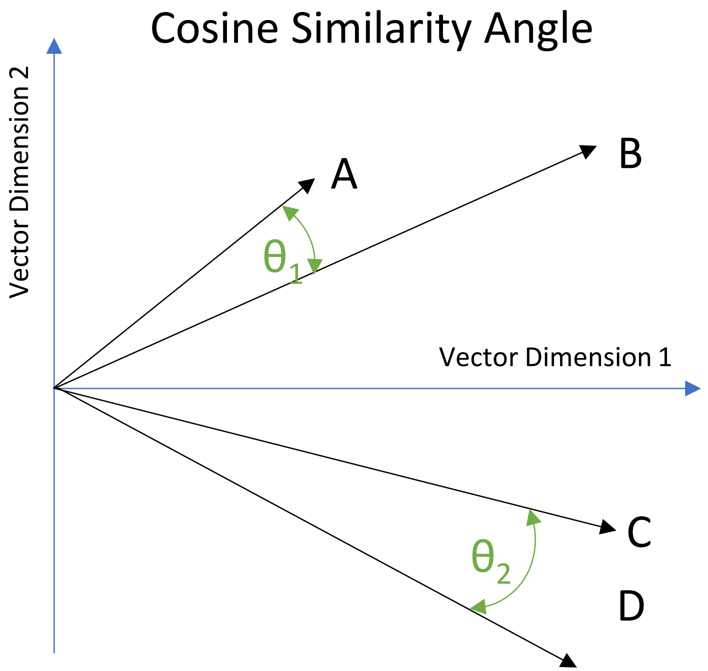
- Measures similarity between Embedding-Vectors based on angle θ.
  - Vectors maximally dissimilar
    → vectors perpendicular (θ = 90°)
  - Vectors completely similar
    → vectors parallel (θ = 0°)



Vector Database

Similarity Search: Cosine Similarity

- Only the angle defines the similarity
- NOT the euclidean distance or magnitude of a vector
- Example
  - A: "The cat sleeps."
  - B: "The feline slumbers peacefully on the soft cushion."
  - C: "Trees grow leaves in spring.“
  - D: "Fish swim in the ocean."



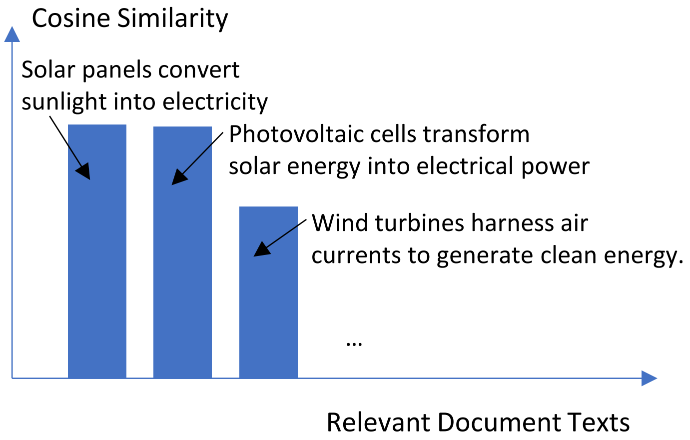
Vector Database

Similarity Search: Maximum Margin Relevance

- Aproach: reduce redundance while maintaining relevance and diversity
- Redundancy…similar vectors
- Relevance…how closely do query and documents match
- Avoid clustering effect

Topic: Renewable Energies

What are the main types of rene wable energy sources and how d o they work?



Data Ingestion Pipeline:
Retrieval-Augmented Generation

Data Ingestion Pipeline

Retrieval-Augmented Generation