O'REILLY

Open Source Large Language Model Bootcamp

Learn How to Answer Questions, Use SBERT, Llama & Co and Tailor Them to Your Needs

Learning Objectives

By the end of this course, you will understand:

- Text retrieval (extractive question answering)
- Working with Open Source LLMs
- Choosing the correct base LLM
- Text generation (generative question answering)
- Deployment in limited environment

Agenda

- Introduction
- Syntactic and Semantic Similarity
- From Word Embeddings to Sentence Embeddings
- SBERT for Calculating Similarity
- Retrieving Content
- Optimizing with Cross Encoders
- Dense Passage Retrieval

Introduction

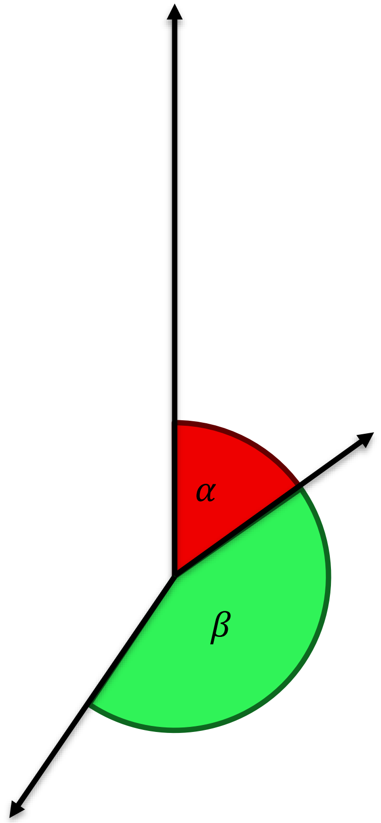
- About me (Christian Winkler)
  - Programming for more than 40 years
  - PhD in physics, working as a professor at a university of applied science
- About the course
  - LLMs and the technology can be intimidating
  - Let's try to clarify the myths
  - Hands on experience

Recap: the document-term matrix

	looking	cheap	flight	where	should	stay	thanks	answer	nearest	train	station	car	airport
Looking for cheap flight?	1	1	1	0	0	0	0	0	0	0	0	0	0
Where should I stay?	0	0	0	1	1	1	0	0	0	0	0	0	0
Thanks for your answer	0	0	0	0	0	0	1	1	0	0	0	0	0
Nearest train station	0	0	0	0	0	0	0	0	1	1	1	0	0
Looking for a car	1	0	0	0	0	0	0	0	0	0	0	1	0
Train to airport	0	0	0	0	0	0	0	0	0	1	0	0	1

Syntactic vs. semantic similarity

- Similarity: measure between 0 and 1, often cosine similarity (angle between vectors)
- Syntactic similarity uses tokens (without flections)
  - Remove flections via lemmatization
  - Cannot capture synonyms etc.
- *Semantic* similarity uses concepts
  - More complex representation needed
- Solution: Word embeddings
  - Most prominent version: word2vec



Word embeddings

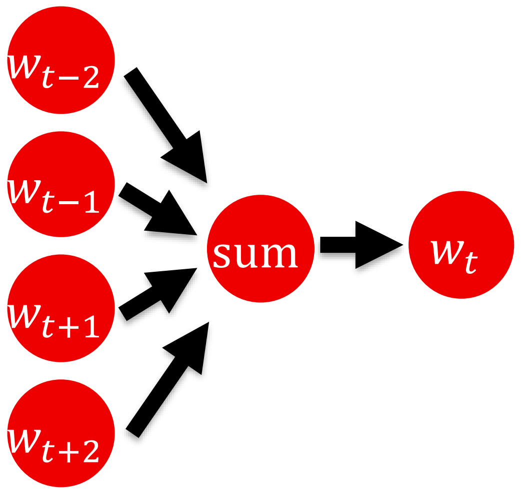
- "You shall know a word by the company it keeps."
- Example
  - What is "tezgüino"?
  - What is similar to "tezgüino"?

A bottle of _____ is on the table.
Everybody likes _____.
Don't have _____ before you drive.
We make _____ out of corn.

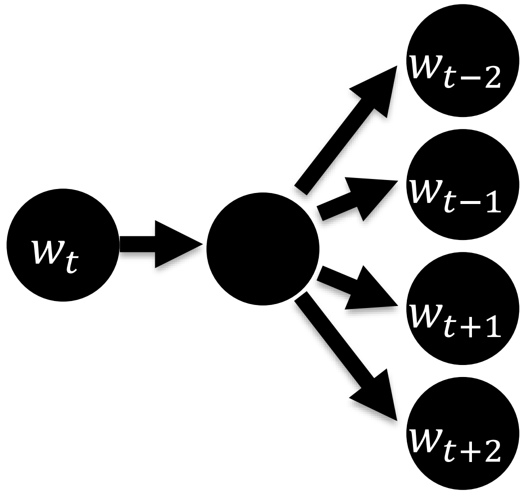
Schematic idea of word embeddings

	"Royalty"	"Masculinity"	"Femininity"	"Age"	
King	0,96	0,99	0,03	0,64	....
Queen	0,99	0,05	0,97	0,72	....
Woman	0,08	0,03	0,98	0,51	....
Princess	0,93	0,01	0,93	0,12	....

Construction

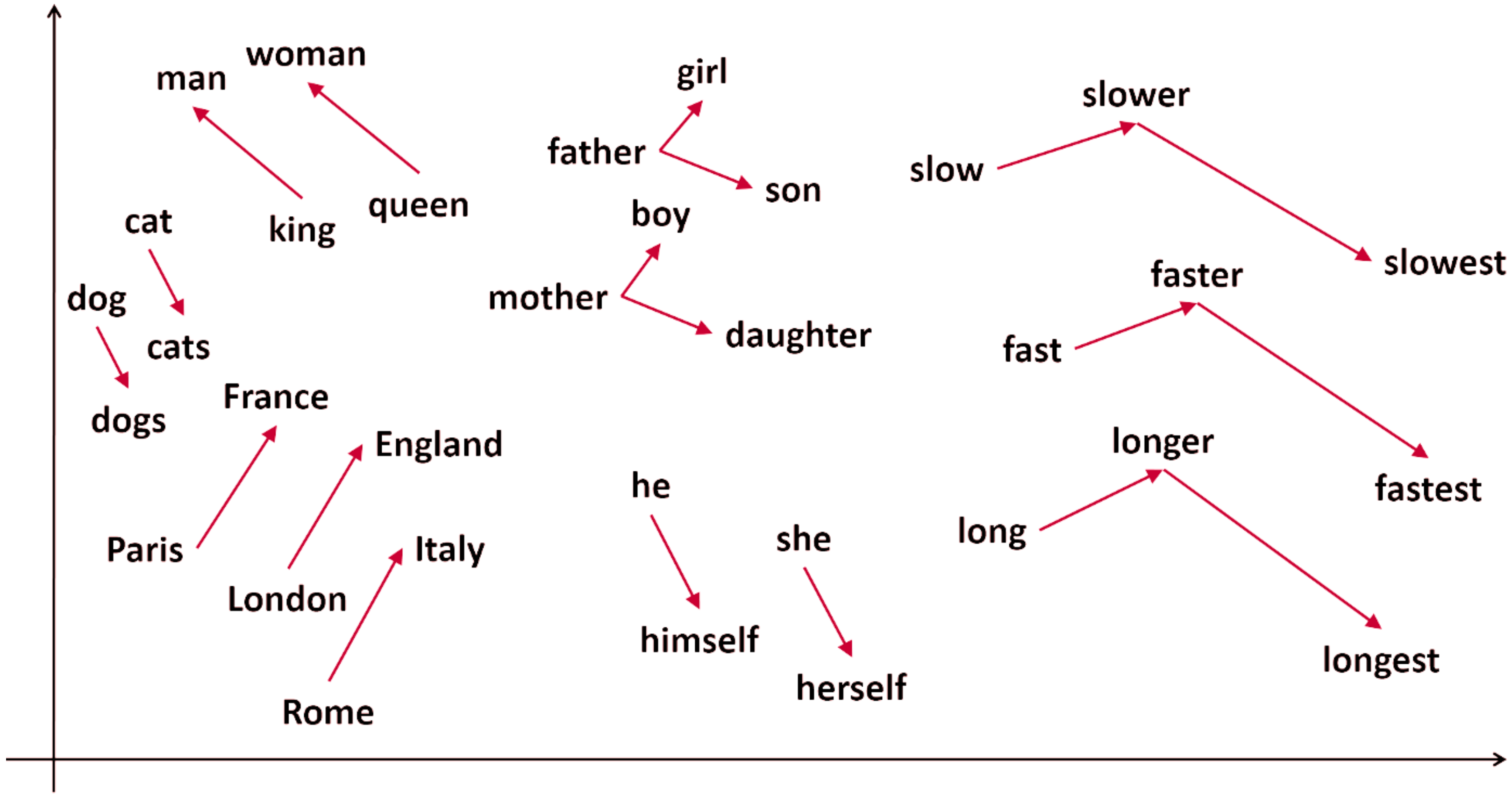


continuous bag-of-words



Skip-gram

Similarities and relationships



Shortcomings of word embeddings

Missing contextualization

- Important for meaning
- Irony and sarcasm

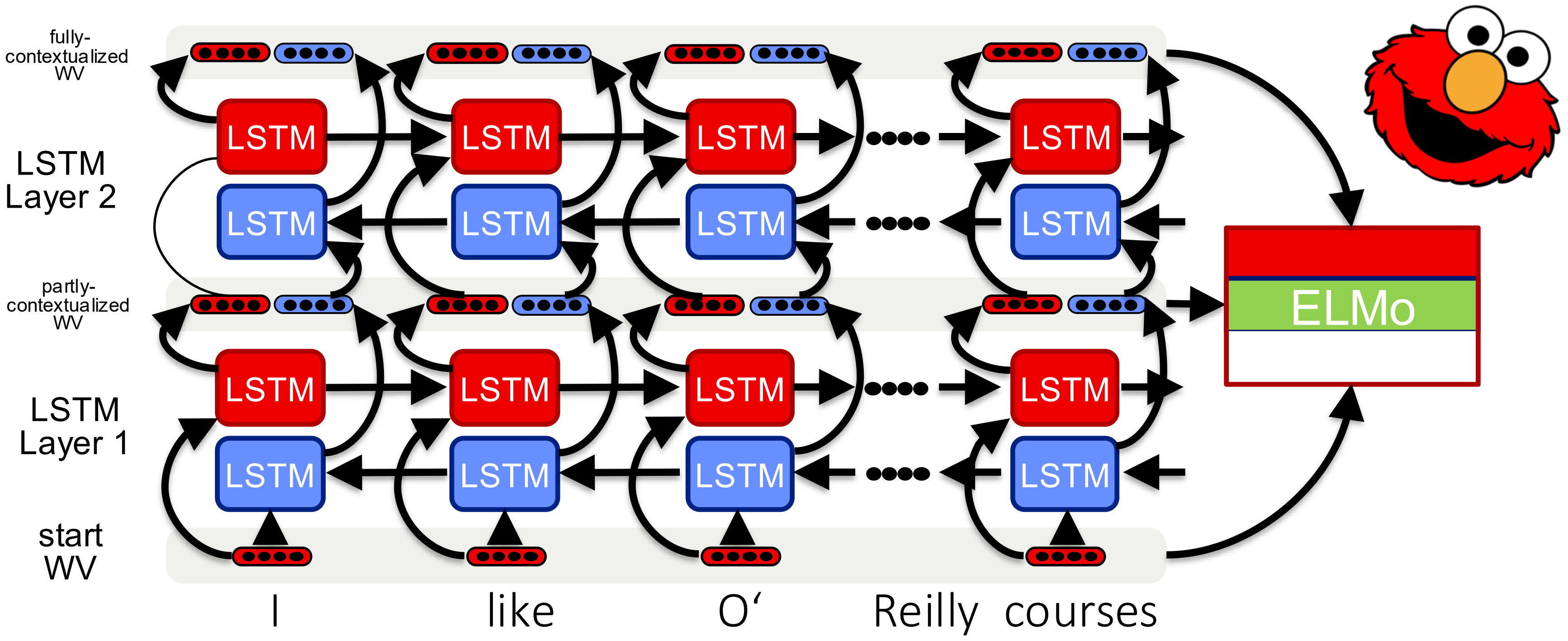
Homonyms cannot be captured

- Meaning of word depends on context
- Read a *book* or *book* a flight

Each model has to be trained separately

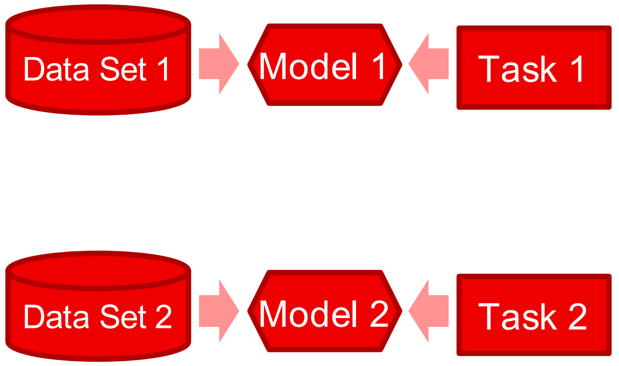
- No transfer learning

Basic idea of contextualized language models



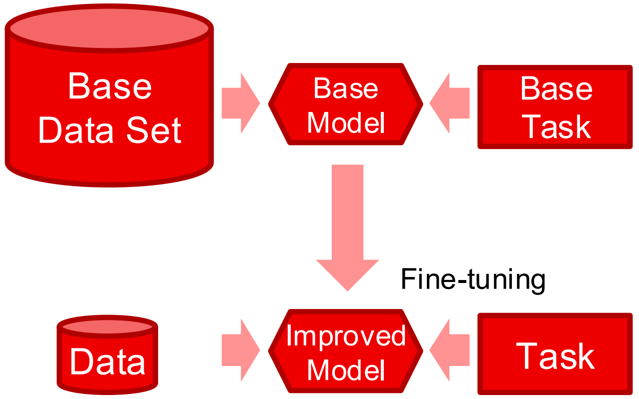
Transfer learning

Classical ML



Each model is trained for one specific task. Start without prior knowledge. Need large labeled training data set.

Transfer Learning



A base model is trained with a large unlabeled dataset. With much less data, it is finetuned for a specific task. Effort is almost negligible compared to base task.

Transformer architecture

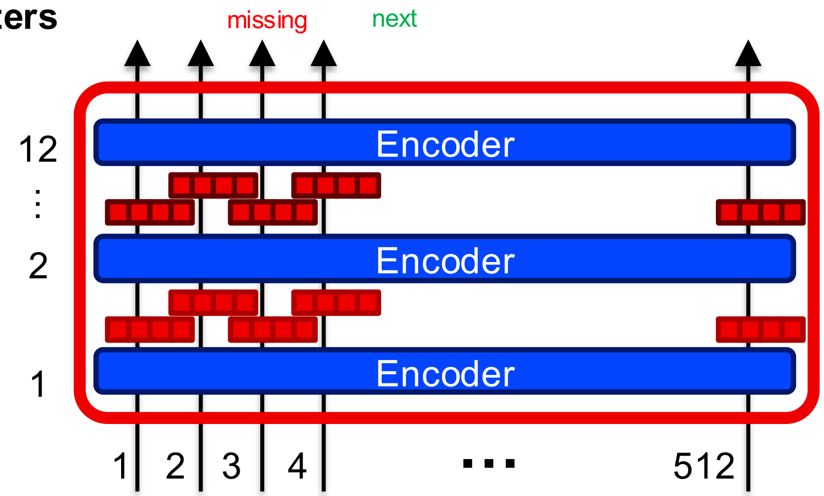
Language models are complex with billions of parameters

Base: Transformer architecture with self-attention

- Attention to which words?
- Many layers with contextualization
- Complex and a bit confusing

BERT: encoder, predict *missing* word

GPT: decoder, predict *next* word



Q&A

Example: sentiment detection

Long standing challenge for NLP

Difficult problem, long time unsolved

Separate words are not enough (e.g. “not”)

Context is important for determining sentiment

Incorporate irony, sarcasm etc.

Could finetune a model based on BERT (part of the finetuning course!)

Here: use existing model (search for sentiment on Hugging Face)

Alternative: zero shot

Work interactively in Jupyter notebook:
Sentiment detection

Challenge: find similar sentences

Is BERT already enough for this?

Almost, but not completely

Model is tuned for guessing **missing** words

Model can be finetuned for **similarity**

- Start with mean pooling: create averages of individual (contextualized) word embeddings
- Use supervised learning with a pre-labeled dataset (part of the finetuning course!)
- ➔ Optimized model understanding similarity

Fortunately, these algorithms have already been implemented

Model is called SBERT (https://sbert.org)

Challenge

- Use existing corpus (= large amount of text)
  - Our example: UN general debates (free)

1. Prepare data
   a) Segment sentences
   b) Calculate SBERT encoding of each sentence (due to time only for 2023)
   c) Save encoding
2. Encode statement
3. Find most similar vector

Recipe

Load data

Calculate sentence fragments

Calculate embeddings for each sentence

Save embeddings

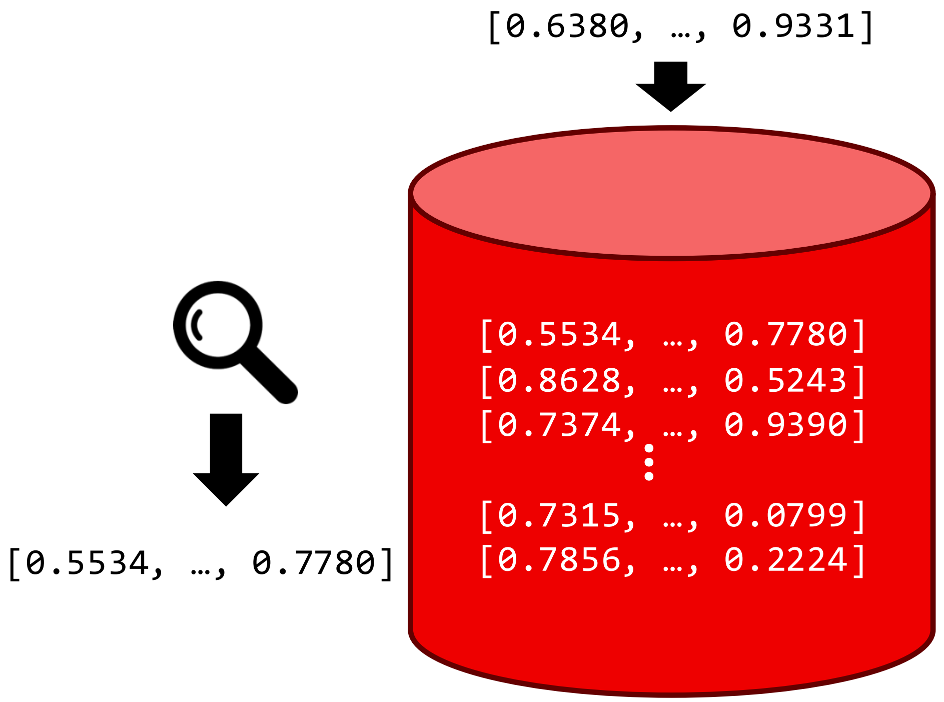
Work interactively in Jupyter notebook:
Semantic retrieval

Q&A

Vector databases

Introduction to vector databases

- Challenge
  - Calculating all distances does not scale
  - Only interested in “best matches”
- Database index?
  - Not possible because of cosine similarity (Euclidean is the same)
- Solution: vector database
  - Specialized store
  - Uses sophisticated methods to find best matches (HNSW)



Options for vector databases

- Milvus
  - Dedicated or integrated
  - Lite version easily accessible from Python
  - Mananged version available
- Weaviate
  - Open source product
  - Cloud-based server
  - Separate instances possible

- ChromaDB
  - Integrated library
  - Popular for Python
  - Calculates embeddings
- qdrant
  - Dedicated vector database
  - Sparse features
  - Powerful and standalone
  - Written in Rust
- PostgreSQL (pgvector)

Work interactively in Jupyter notebook:
Use vector database

Q&A

Lexical databases and rank fusion

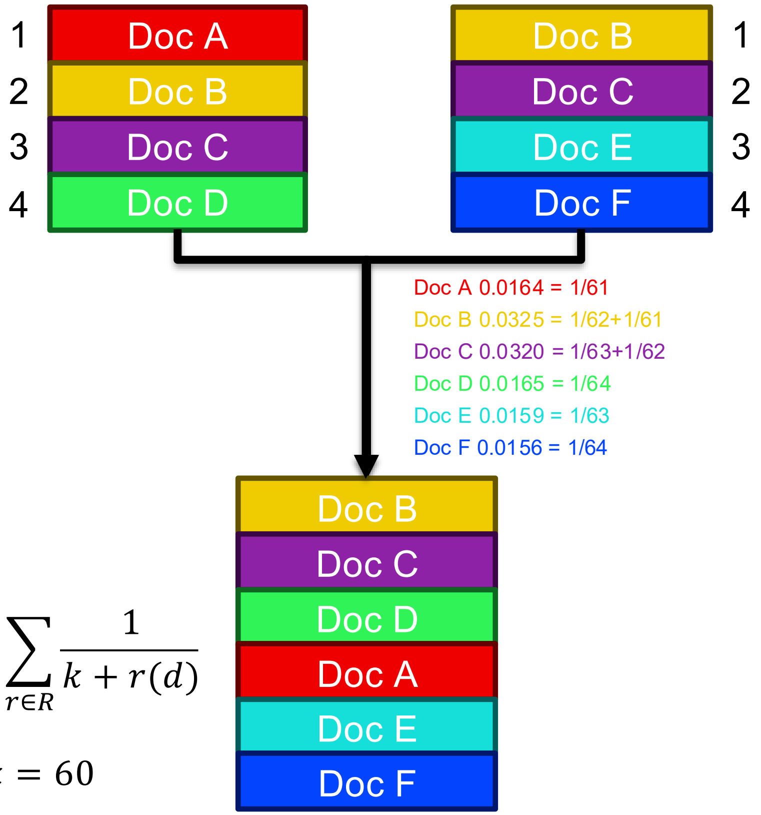
Lexical retrieval

- General idea
  - Look for word (or combinations)
  - Ranking with TF/IDF or Okapi BM25
  - Retrieval via inverted index
- Software options
  - ElasticSearch
  - Apache Solr
  - Tantivy

	looking	cheap	flight	where	should	stay	thanks	answer	nearest	train	station	car	airport
Looking for cheap flight?	1	1	1	0	0	0	0	0	0	0	0	0	0
Where should I stay?	0	0	0	1	1	1	0	0	0	0	0	0	0
Thanks for your answer	0	0	0	0	0	0	1	1	0	0	0	0	0
Nearest train station	0	0	0	0	0	0	0	0	1	1	1	0	0
Looking for a car	1	0	0	0	0	0	0	0	0	0	0	1	0
Train to airport	0	0	0	0	0	0	0	0	0	1	0	0	1

Rank fusion

- Problem
  - Scores for semantic matches (cosine)
  - Scores for lexical matches (BM25)
  - Different metrics (incommensurable)
- Solution
  - Consider order of results (ranking)
  - “Merge” result lists
  - Popular algorithm:
    Reciprocal Rank Fusion (RRF)
  - Many more available



Doc A 0.0164 = 1/61
Doc B 0.0325 = 1/62+1/61
Doc C 0.0320 = 1/63+1/62
Doc D 0.0165 = 1/64
Doc E 0.0159 = 1/63
Doc F 0.0156 = 1/64

$$RRF(d) = \sum_{r \in R} \frac{1}{k + r(d)}$$

$$k = 60$$

Work interactively in Jupyter notebook:

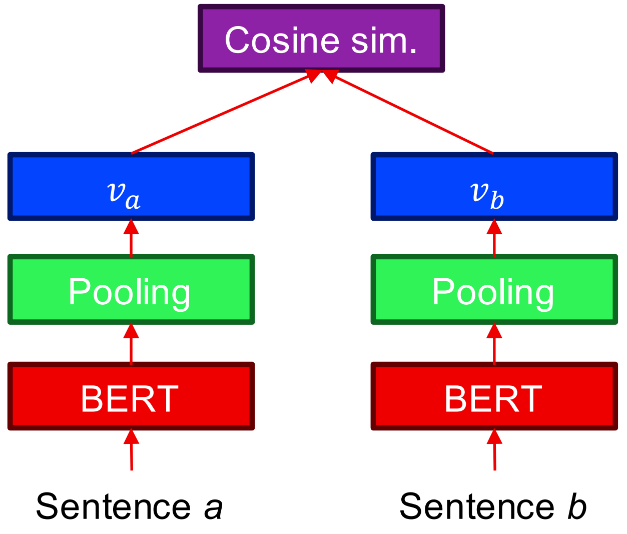
Implement RRF

Q&A

Cross encoders

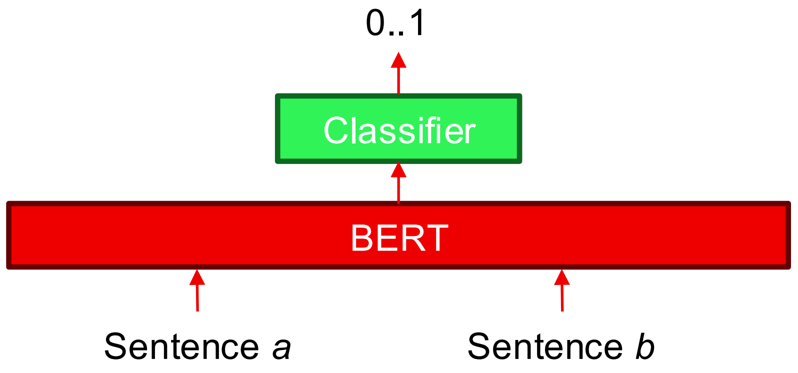
How do cross-encoders work

So far…



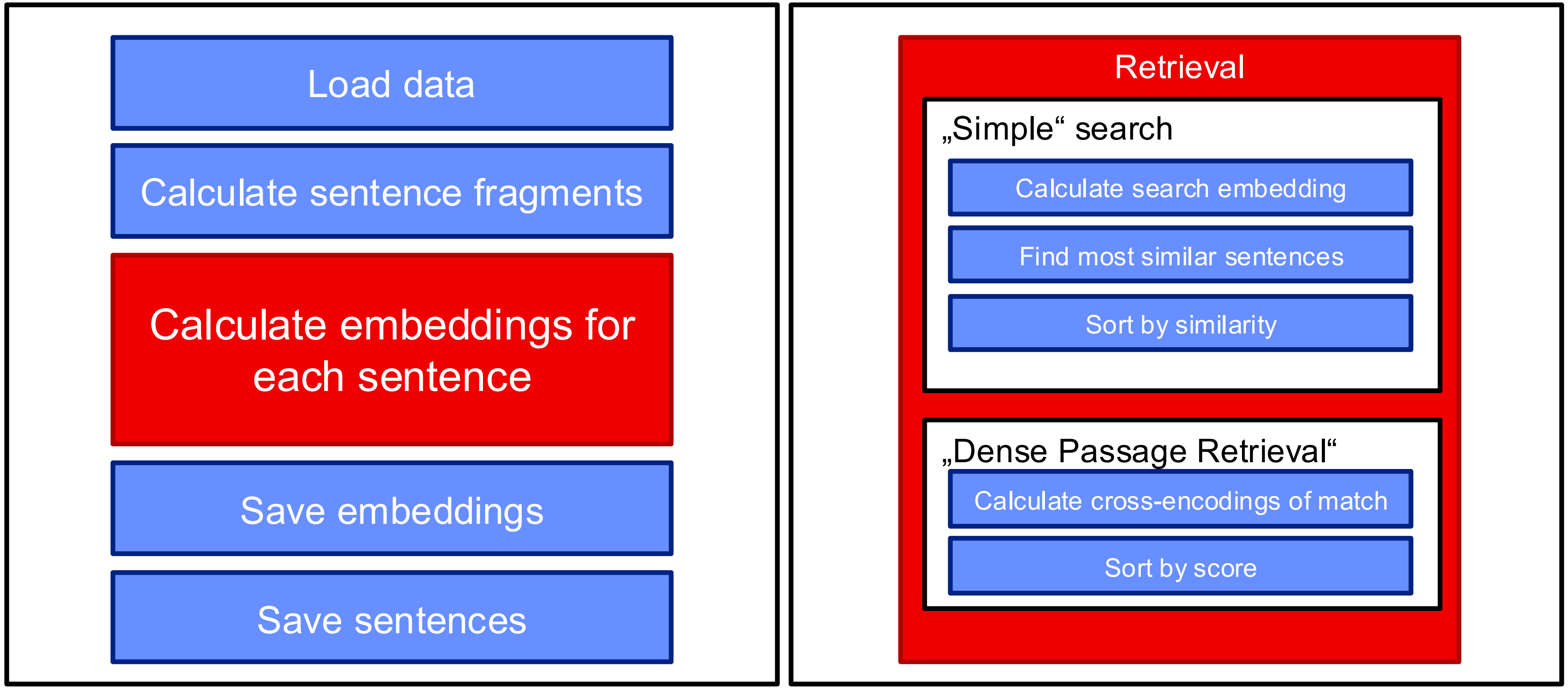
Bi-encoder

New



Cross-encoder

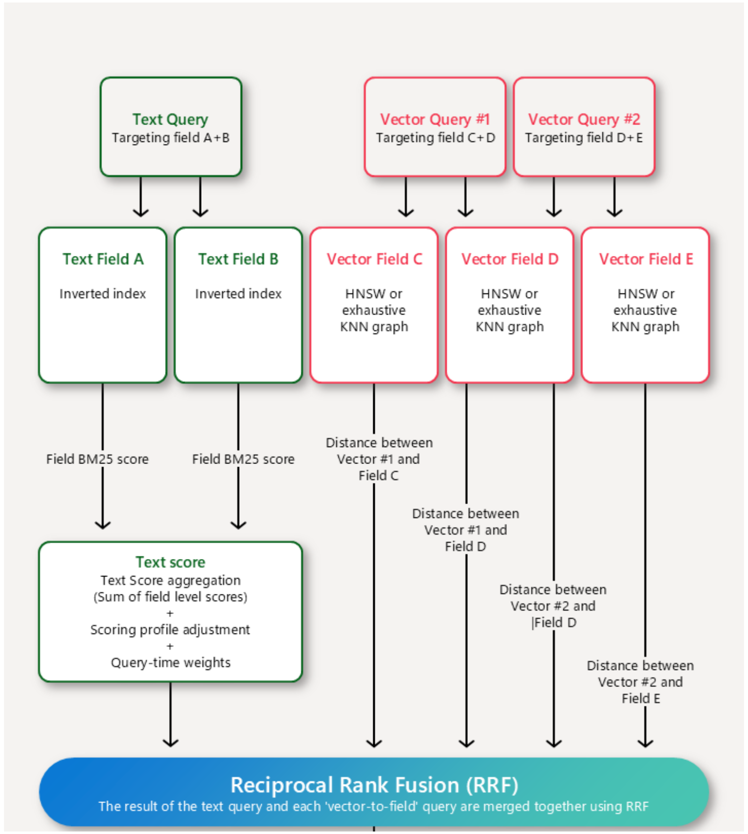
Improving the prior solution

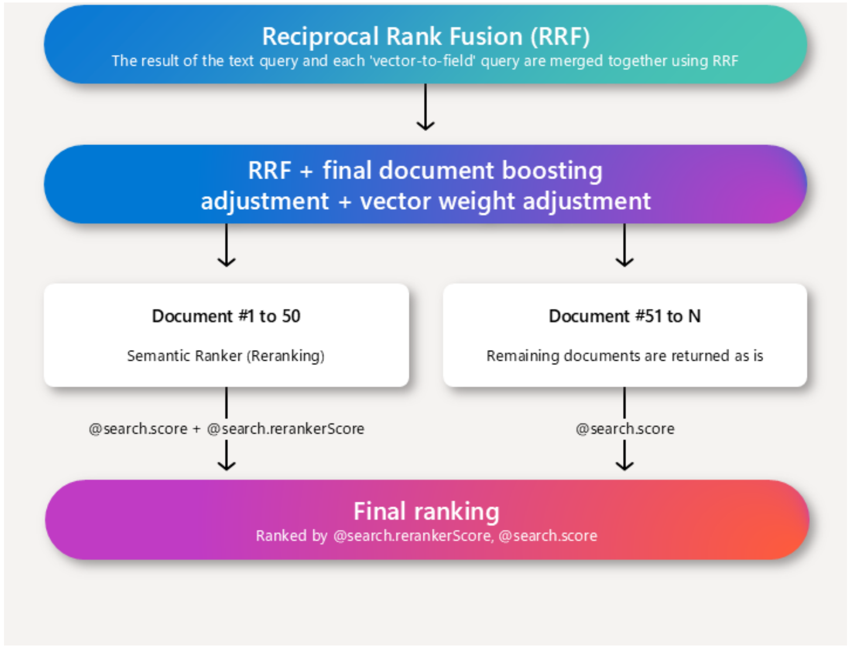

Load data
Calculate sentence fragments
Calculate embeddings for each sentence
Save embeddings
Save sentences
Retrieval
„Simple“ search
Calculate search embedding
Find most similar sentences
Sort by similarity
„Dense Passage Retrieval“
Calculate cross-encodings of match
Sort by score

Work interactively in Jupyter notebook:

Dense passage retrieval

Combinations are possible!


Text Query
Targeting field A+B
Vector Query #1
Targeting field C+D
Vector Query #2
Targeting field D+E
Text Field A
Inverted index
Text Field B
Inverted index
Vector Field C
HNSW or exhaustive KNN graph
Vector Field D
HNSW or exhaustive KNN graph
Vector Field E
HNSW or exhaustive KNN graph
Field BM25 score
Field BM25 score
Distance between Vector #1 and Field C
Distance between Vector #1 and Field D
Distance between Vector #2 and Field D
Distance between Vector #2 and Field E
Text score
Text Score aggregation
(Sum of field level scores)
+
Scoring profile adjustment
+
Query-time weights
Reciprocal Rank Fusion (RRF)
The result of the text query and each 'vector-to-field' query are merged together using RRF


Reciprocal Rank Fusion (RRF)
The result of the text query and each 'vector-to-field' query are merged together using RRF
RRF + final document boosting adjustment + vector weight adjustment
Document #1 to 50
Semantic Ranker (Reranking)
Document #51 to N
Remaining documents are returned as is
@search.score + @search.rerankerScore
@search.score
Final ranking
Ranked by @search.rerankerScore, @search.score

Source: https://learn.microsoft.com/en-us/azure/search/hybrid-search-ranking

Q&A

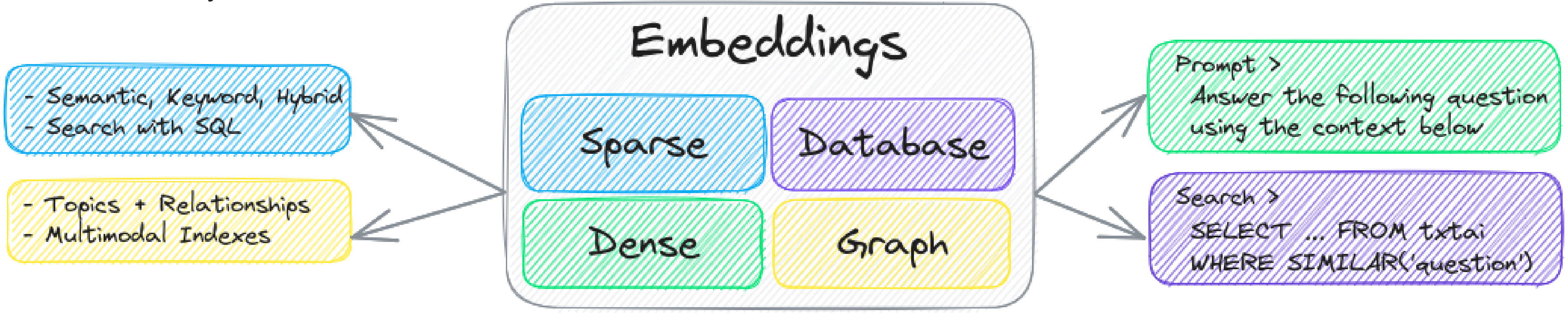
Using existing software

LangChain

- Open source framework
- Tries to unify LLM handling via APIs
- Pros
  - Very easy to change backends etc.
  - Working with pipelines
  - Easier handling of “chained” method call
  - Very popular and many examples
- Cons
  - Code quality sometimes doubtful
  - Documentation hard to read

txtai

- “txtai is an all-in-one embeddings database for semantic search, LLM orchestration and language model workflows.”
- Everything is integrated
- Very nice API


- Semantic, Keyword, Hybrid
- Search with SQL
- Topics + Relationships
- Multimodal Indexes
Embeddings
Sparse
Database
Dense
Graph
Prompt >
Answer the following question using the context below
Search >
SELECT ... FROM txtai WHERE SIMILAR('question')

LlamaIndex

- Open source framework
- Integrated cloud available
- Uses OpenAI as standard, but configurable
- Pros
    - Easy to change backends etc.
    - Very active and loved by developers
    - Good documentation
- Cons
    - Still a bit new
    - Extractive and generative models entangled

Work interactively in Jupyter notebook: use LangChain and txtai

Summary & discussion

Group Discussion

- Any questions left?
- What would you like to achieve with this technology?

Review Course Outcomes

By the end of this course, you should be able to:

- Calculate sentence embeddings with SBERT
- Understand and use cross-encoders
- Use existing software like LangChain and txtai to build a semantic search engine

Resources

GitHub repository

- https://github.com/datanizing/oreilly-open-source-llm
- Continuously updated

Qwen models

- https://huggingface.co/Qwen

General Discussion

- https://www.reddit.com/r/LocalLLaMA/