Neural Information Retrieval

PRI 23/24 · Information Processing and Retrieval
M.EIC · Master in Informatics Engineering and Computation

Sérgio Nunes
Dept. Informatics Engineering
FEUP · U.Porto

Based on An Introduction to Neural Information Retrieval, B. Mitra and N. Craswell (2018).
Based on Lecture Notes on Neural Information Retrieval, N. Tonellotto (2022).

Concepts and Terminology (1)

- ➔ **Neural Networks**, computational model consisting of layers of interconnected nodes that process and transform input data to produce an output.
  - ➔ **Shallow NN**, uses only one or two layers; used in “shallow learning” tasks.
  - ➔ **Deep NN**, uses multiple layers; used in “deep learning” tasks.
- ➔ **Deep Learning**, subset of machine learning that uses neural networks with multiple layers to learn complex patterns from data.
- ➔ **Dense Vectors**, numerical vector representations where most elements are non-zero; used to encode dense data.
- ➔ **Sparse Vectors**, numeral vector representations where most elements are zero; used to encode sparse data.
- ➔ **Embeddings**, low-dimensional numerical vector representations of objects (e.g., queries, text, images); able to capture relationships and similarities.
  - ➔ **Word Embeddings**, specific type of embedding used in NLP to represent words.

Concepts and Terminology (2)

- ➔ **Transformers**, neural network architecture with high impact in NLP problems; efficiently captures relationships between words.
    - ➔ **Encoder**: component in neural networks that processes input data to an encoded representation (e.g., dense vector).
    - ➔ **Decoder**: component in neural networks that generates output based on the encoded representation.
- ➔ **BERT** (Bidirectional Encoder Representations from Transformers), pre-trained language model built on transformer architecture.
- ➔ **Neural IR**, application of neural networks in the context of information retrieval tasks.
- ➔ **Semantic Search**, application of search techniques that leverage on understanding the meaning (semantics) of user queries and documents to improve relevance estimation.
- ➔ **Dense Retrieval**, use of dense vector representations for document retrieval and ranking.
- ➔ **Language Model**, learned computational model representing text; used in many NLP tasks.
- ➔ **Large Language Model**, language models that use a large number of parameters (i.e., millions or billions).

Deep Learning

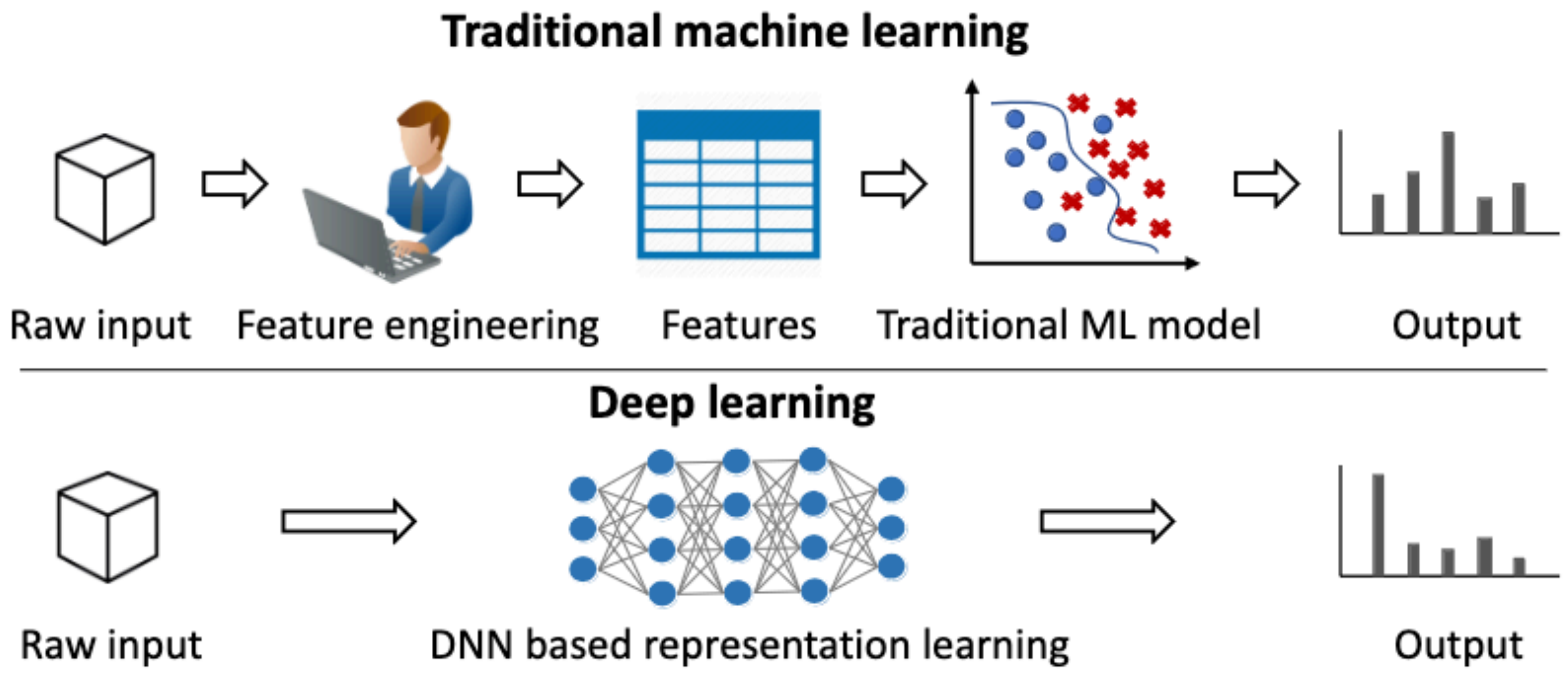


Figure 2: A traditional machine learning pipeline using feature engineering, and a deep learning pipeline using DNN based representation learning.

Figure from: Techniques for Interpretable Machine Learning, M. Du, N. Liu, X. Hu (2020).

Industry Adoption: Bing and Google

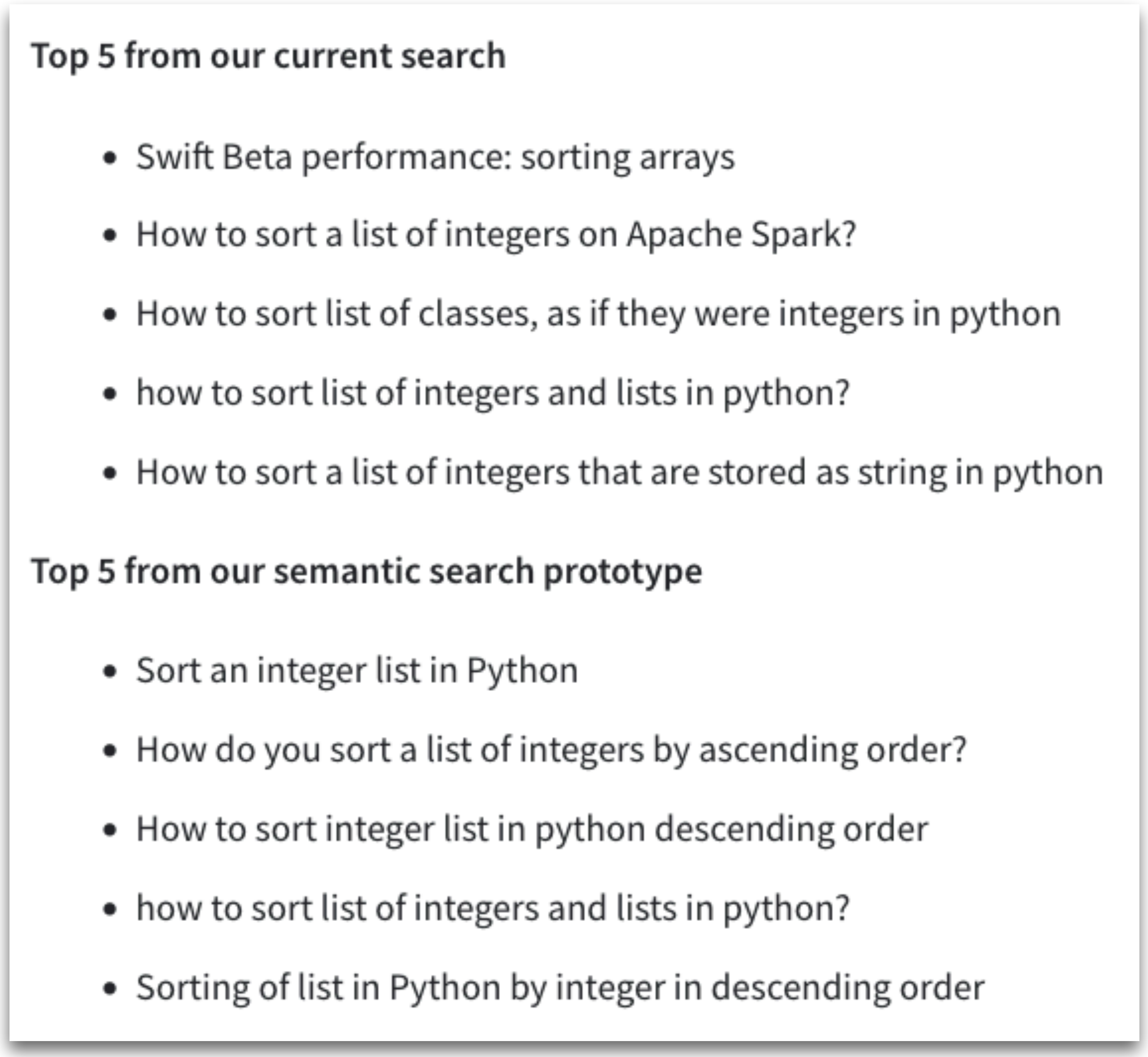
- ➔ In 2019, both Google and Bing announced the use of BERT models in search.
- ➔ Bing delivers its largest improvement in search experience using Azure GPUs [source]
  - ➔ "*All these enhancements lead toward more relevant, contextual results for web search queries.*"
  - ➔ "*Deep learning at web-search scale can be prohibitively expensive*"
- ➔ Understanding searches better than ever before (Google) [source]
  - ➔ "*With BERT, Search is able to grasp this nuance and know that the very common word "to" actually matters a lot here, and we can provide a much more relevant result for this query.*"
  - ➔ "*A powerful characteristic of these systems is that they can take learnings from one language and apply them to others. So we can take models that learn from improvements in English (a language where the vast majority of web content exists) and apply them to other languages.*"

Industry Adoption (1): StackOverflow

- ➔ In 2023, StackOverflow announced the adoption of “semantic search” using BERT.
- ➔ Ask like a human: Implementing semantic search on Stack Overflow (2023) [source]
  - ➔ *“When Stack Overflow started in 2008, we used Microsoft SQL’s full-text search capabilities. As our website grew to one of the most heavily trafficked domains on the internet, we needed to expand. We replaced it with Elasticsearch, which is the industry standard, and have been using that ever since.”*
  - ➔ *“[…] lexical search requires a domain-specific language to get results for anything more than a stack of keywords. […] To get good results, you shouldn’t need to know any magic words. With semantic search, you don’t.”*
  - ➔ *“Semantic search is a completely different paradigm than lexical search. Imagine the search space as a 3D cube where all the documents are given a numerical score based on their meaning and embedded into vectors with coordinates. Vectors are stored in proximity to each other according to the meanings of their terms.”*
- ➔ Challenges identified:
  - ➔ *“The first challenge is deciding what model to use. We could choose between a pre-tuned open-source model, a proprietary model, or fine-tuning our own model. Luckily, our data dumps have been used in many of these embedding models, so they are already optimized for our data to some extent.”*
  - ➔ *“The second challenge is deciding how to chunk the text; that is, how to break the text up into tokens for embedding.”*

Industry Adoption (2): StackOverflow example

➔ When searching for [how to sort list of integers in python].



Top 5 from our current search

- Swift Beta performance: sorting arrays
- How to sort a list of integers on Apache Spark?
- How to sort list of classes, as if they were integers in python
- how to sort list of integers and lists in python?
- How to sort a list of integers that are stored as string in python

Top 5 from our semantic search prototype

- Sort an integer list in Python
- How do you sort a list of integers by ascending order?
- How to sort integer list in python descending order
- how to sort list of integers and lists in python?
- Sorting of list in Python by integer in descending order

Source: Ask like a human: Implementing semantic search on Stack Overflow (2023)
https://stackoverflow.blog/2023/07/31/ask-like-a-human-implementing-semantic-search-on-stack-overflow

Neural Information Retrieval

- ➔ Neural IR is the application of shallow or deep neural networks to IR tasks.
  - ➔ Neural models have been employed in many IR scenarios—including ad-hoc retrieval, recommender systems, multimedia search, and even conversational systems that generate answers in response to natural language questions.
- ➔ Unlike Learning to Rank approaches that train machine learning models over a set of hand-crafted features, neural models accept the raw text of a query and document as input.
- ➔ Neural networks are typically used in two different ways:
  - ➔ Learn the ranking functions combining the relevance signals to produce an ordering of documents.
  - ➔ Learn the abstract representations of documents and queries to capture their relevance information.

Data in Neural Information Retrieval

- ➔ Neural models for IR use vector representations of text, which usually contain a large number of parameters that need to be tuned.
- ➔ ML models with large set of parameters typically benefit from large quantity of training data.
- ➔ Learning suitable representations of text demands large-scale datasets for training.

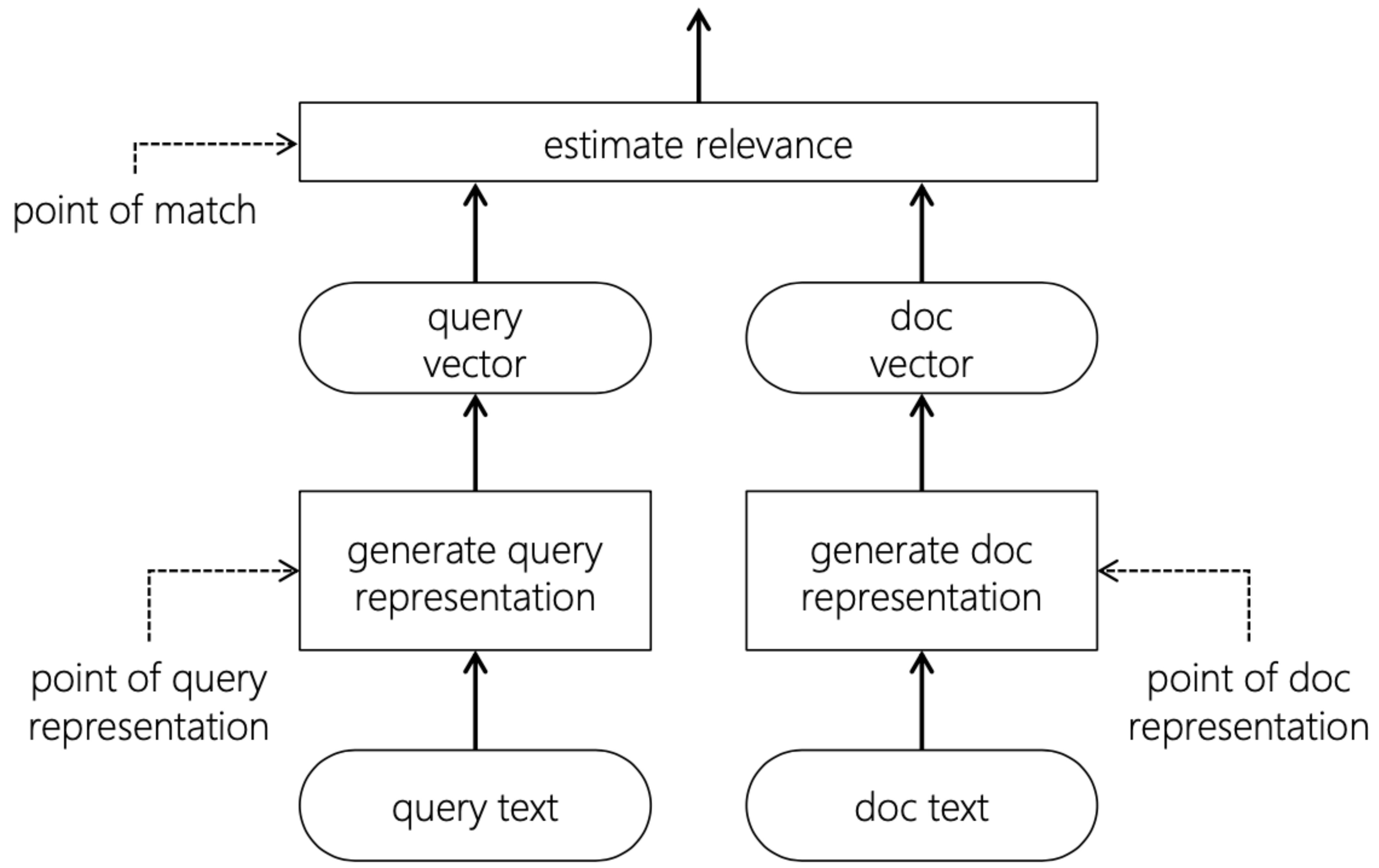
- ➔ Therefore, unlike classical IR models, these neural approaches tend to be data hungry, with performance that improves with more training data.

Document Ranking

- ➔ Document ranking comprises three steps (recall “IR models” topic):
  - ➔ Generate a representation of the query that specified the information need.
  - ➔ Generate a representation of the document.
  - ➔ Match the query and the document representations to estimate their mutual relevance.

- ➔ Neural approaches can influence one or more of these three steps.
- ➔ Unlike traditional learning to rank models, neural architectures depend less on manual feature engineering on more on automatically detecting regularities in matching patterns.

Document Ranking



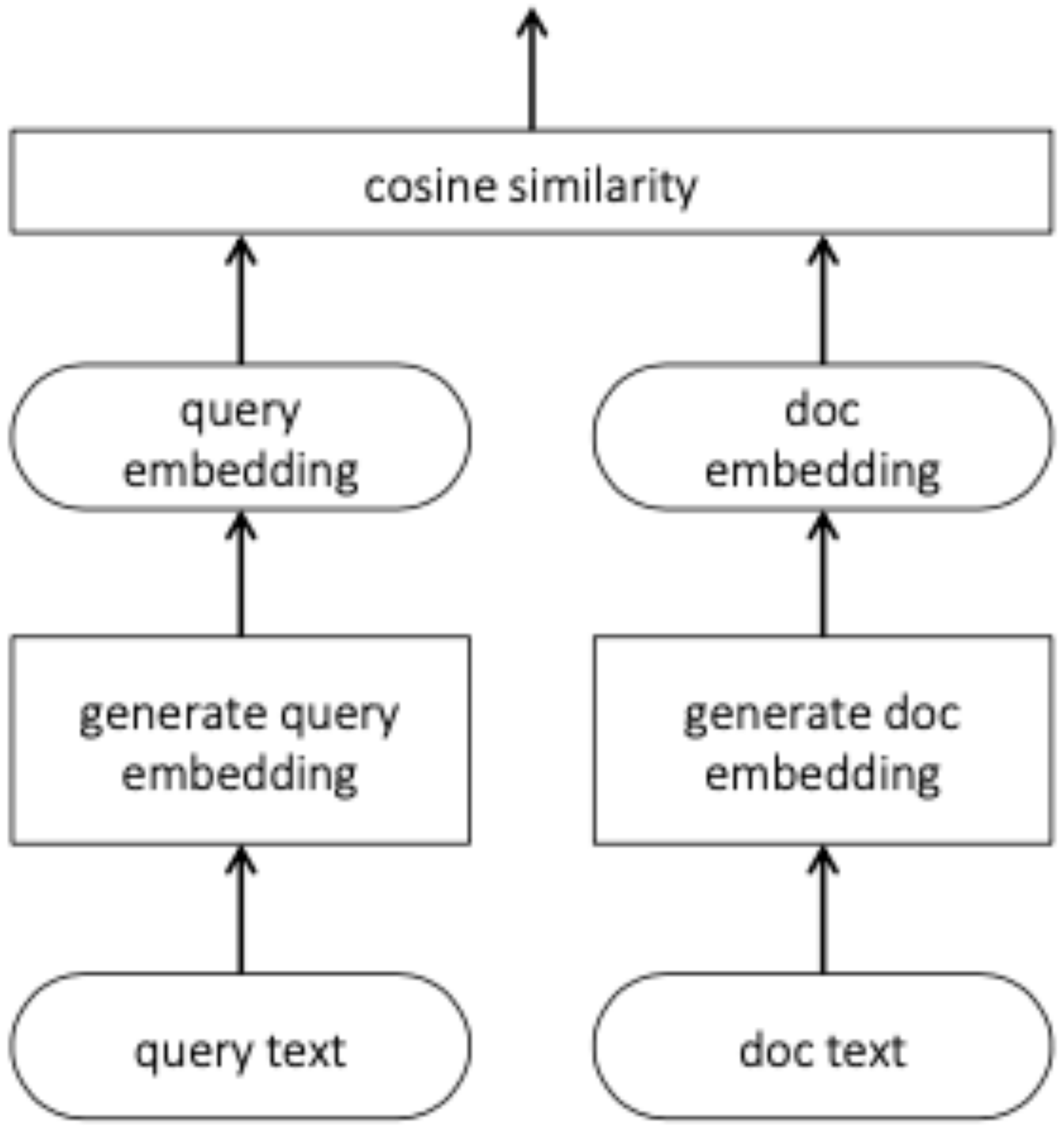
estimate relevance
point of match
query vector
doc vector
generate query representation
generate doc representation
point of query representation
point of doc representation
query text
doc text

Figure 2.3: Document ranking typically involves a query and a document representation steps, followed by a matching stage. Neural models can be useful either for generating good representations or in estimating relevance, or both.

Document Ranking

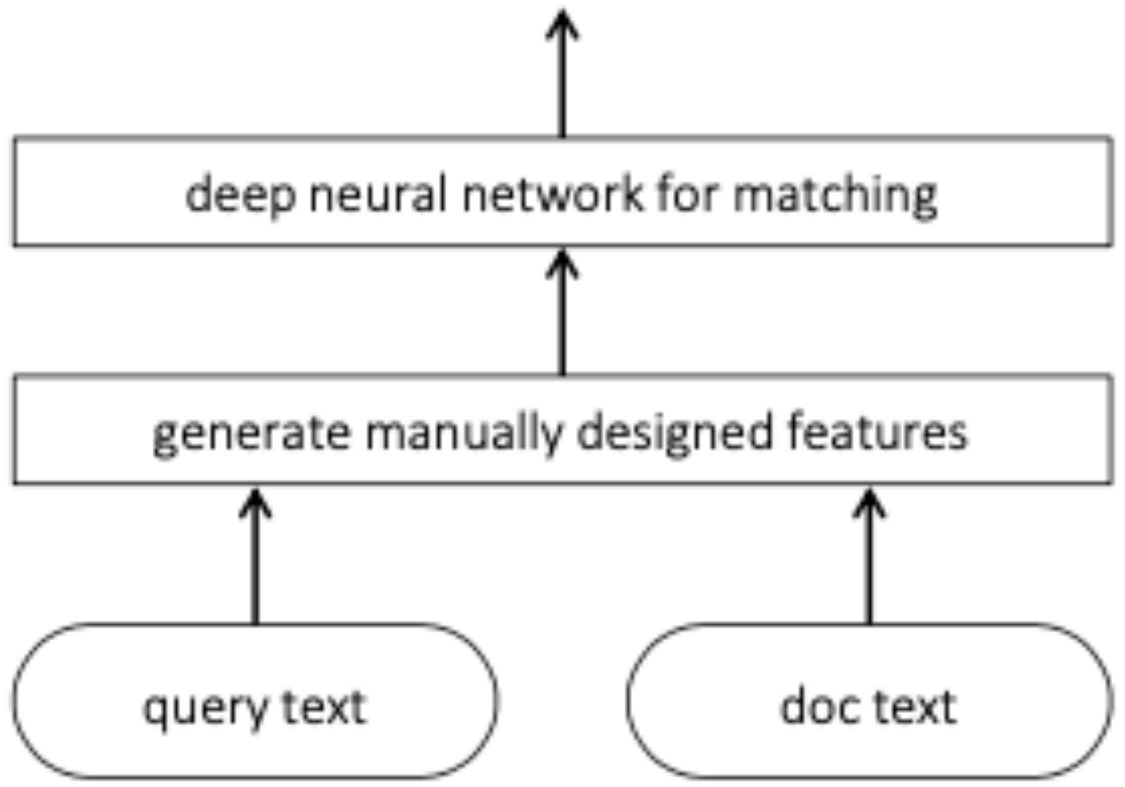
➔ Many neural IR models depend on learning low-dimensional vector representations (or embeddings) of query and document text.

➔ And then use them with traditional IR models in conjunction with simple similarity metrics (e.g., cosine similarity).

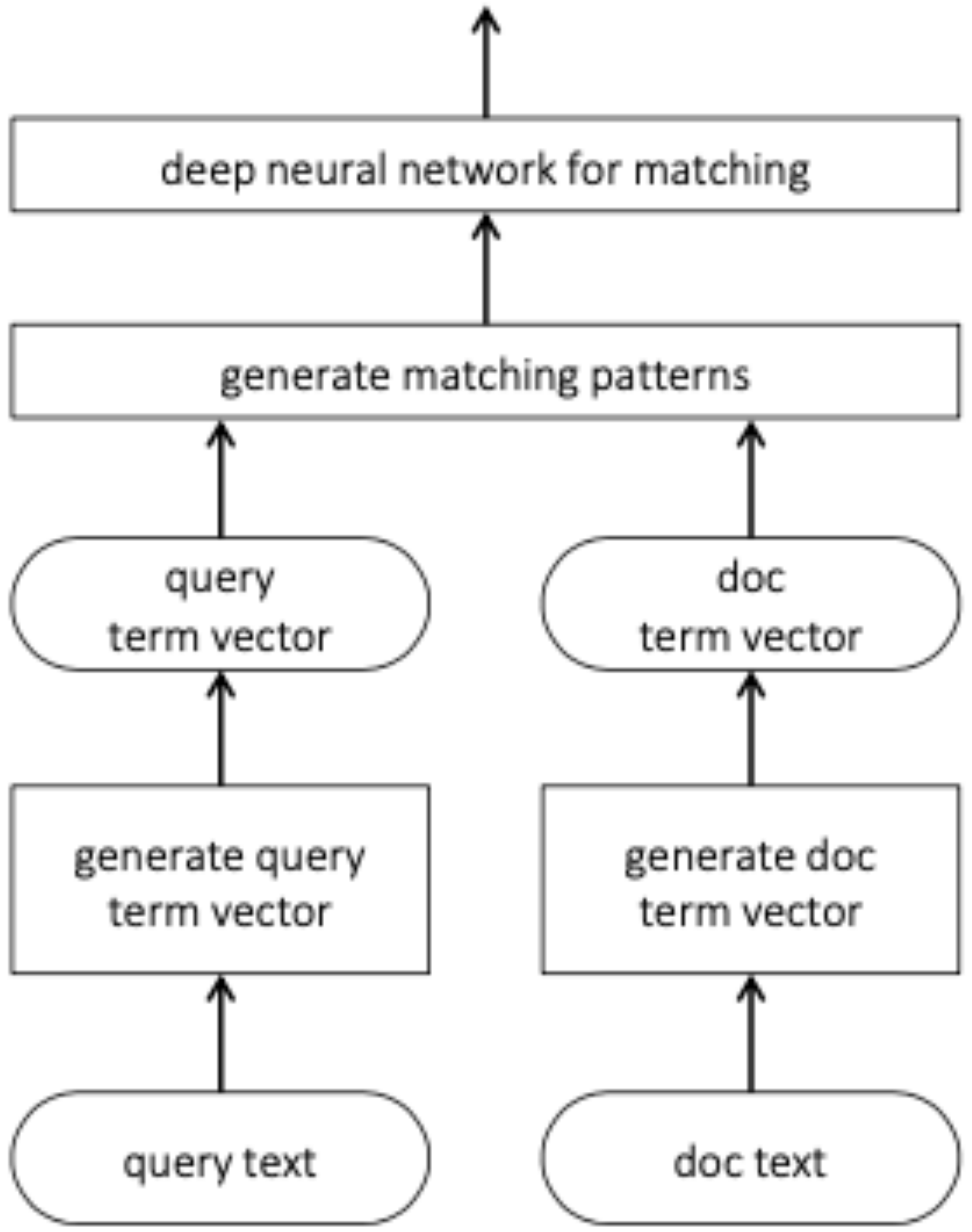


(c) Learning query and document representations for matching (*e.g.*, (Huang *et al.*, 2013; Mitra *et al.*, 2016a))

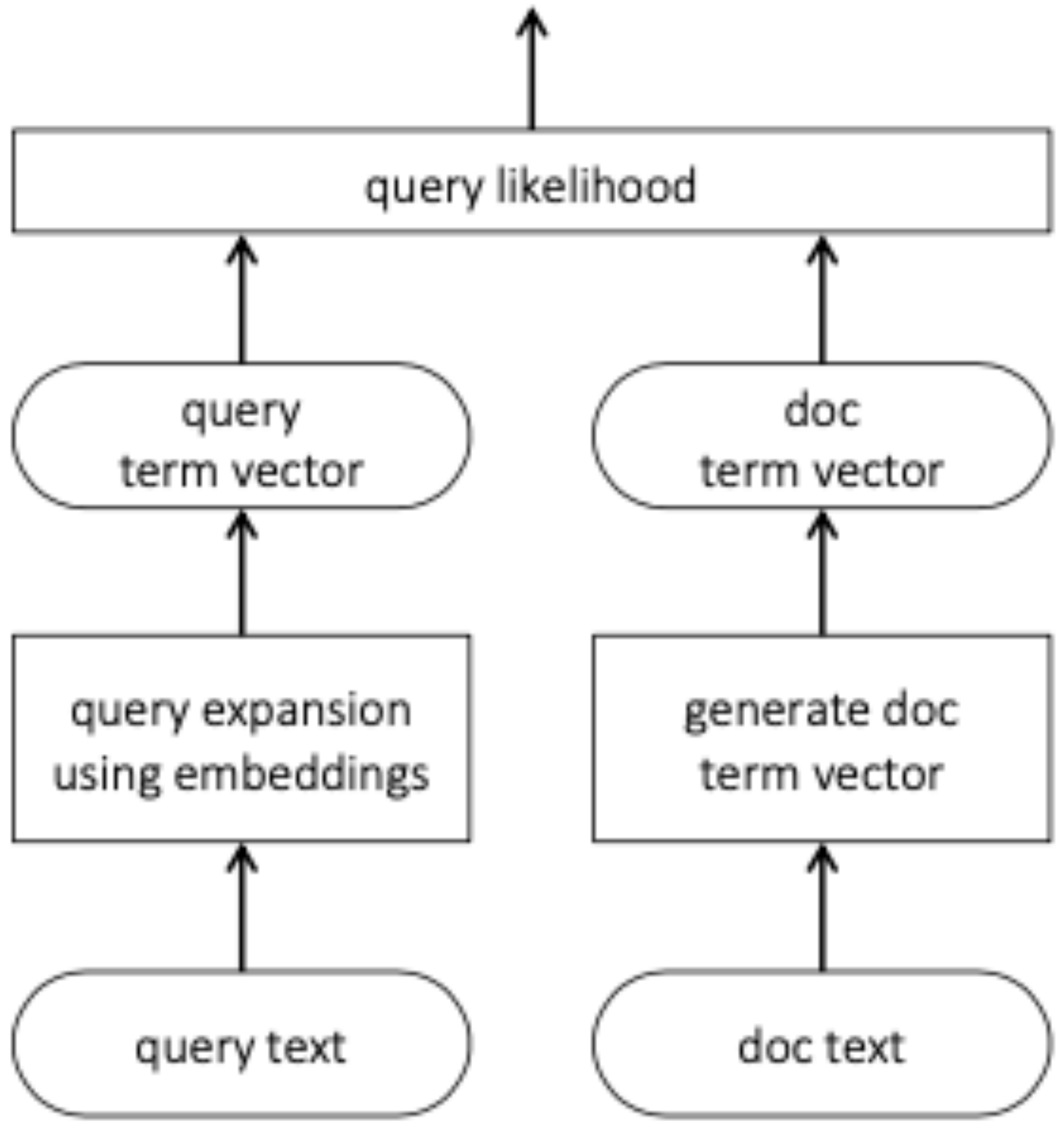
Document Ranking


deep neural network for matching
generate manually designed features
query text
doc text

(a) Learning to rank using manually designed features (*e.g.*, Liu (2009))


deep neural network for matching
generate matching patterns
query term vector
doc term vector
generate query term vector
generate doc term vector
query text
doc text

(b) Estimating relevance from patterns of exact matches (*e.g.*, (Guo *et al.*, 2016a; Mitra *et al.*, 2017a))



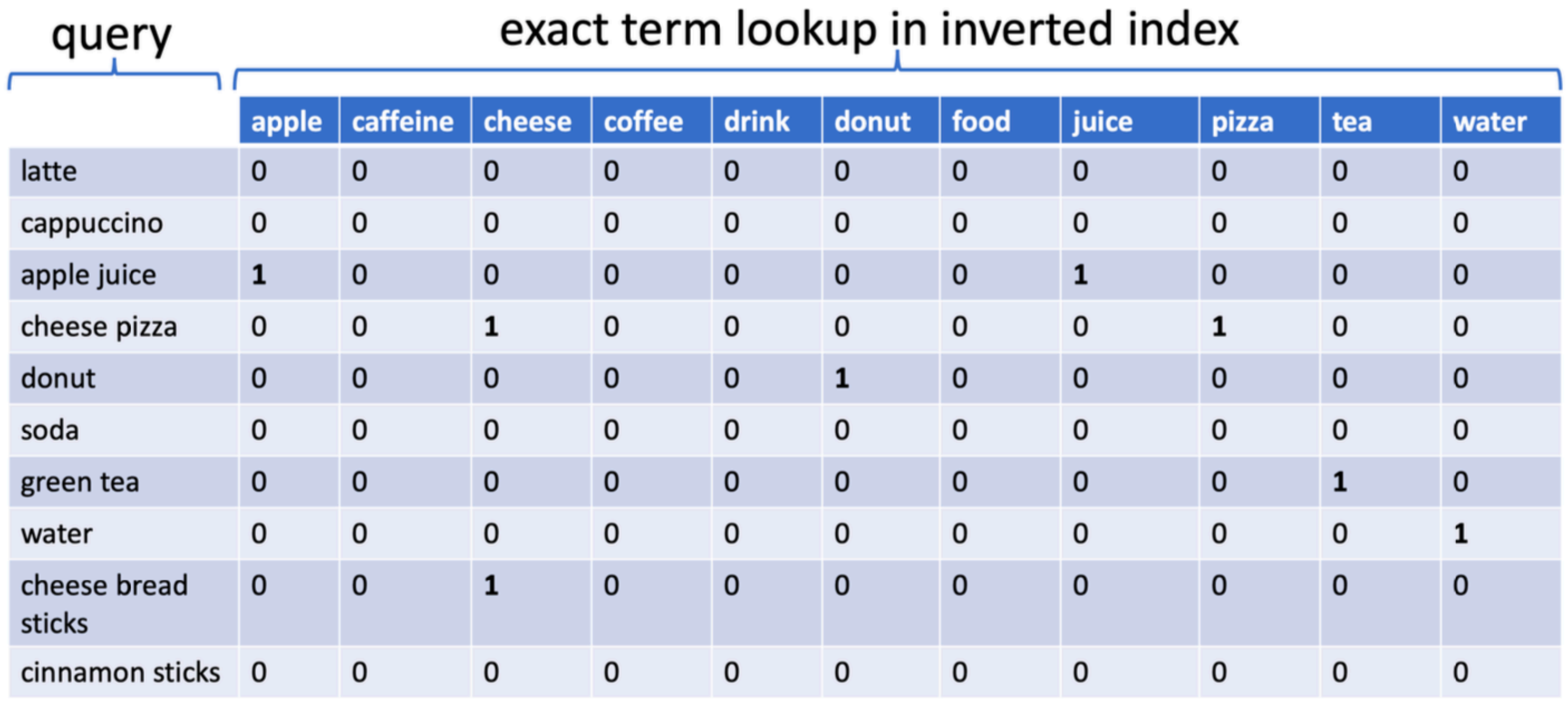
(d) Query expansion using neural embeddings (*e.g.*, (Roy *et al.*, 2016; Diaz *et al.*, 2016))

Image from An Introduction to Neural Information Retrieval, B. Mitra and N. Craswell (2018).

Word Embeddings

- ➔ In the 1950s, many linguists formulated the distributional hypothesis: words that occur in the same contexts tend to have similar meanings.
- ➔ According to this hypothesis, the meaning of words can be inferred by their usage together with other words in existing texts.
- ➔ "*You shall know a word by the company it keeps.*", John Firth (1957)
- ➔ In recent years, the distributional hypothesis has been applied to create semantic understandings of terms and term sequences through word embeddings.
- ➔ A word embedding is a numerical vector that represents the semantic meaning of a given term sequence.

Recall Sparse Vectors Representation

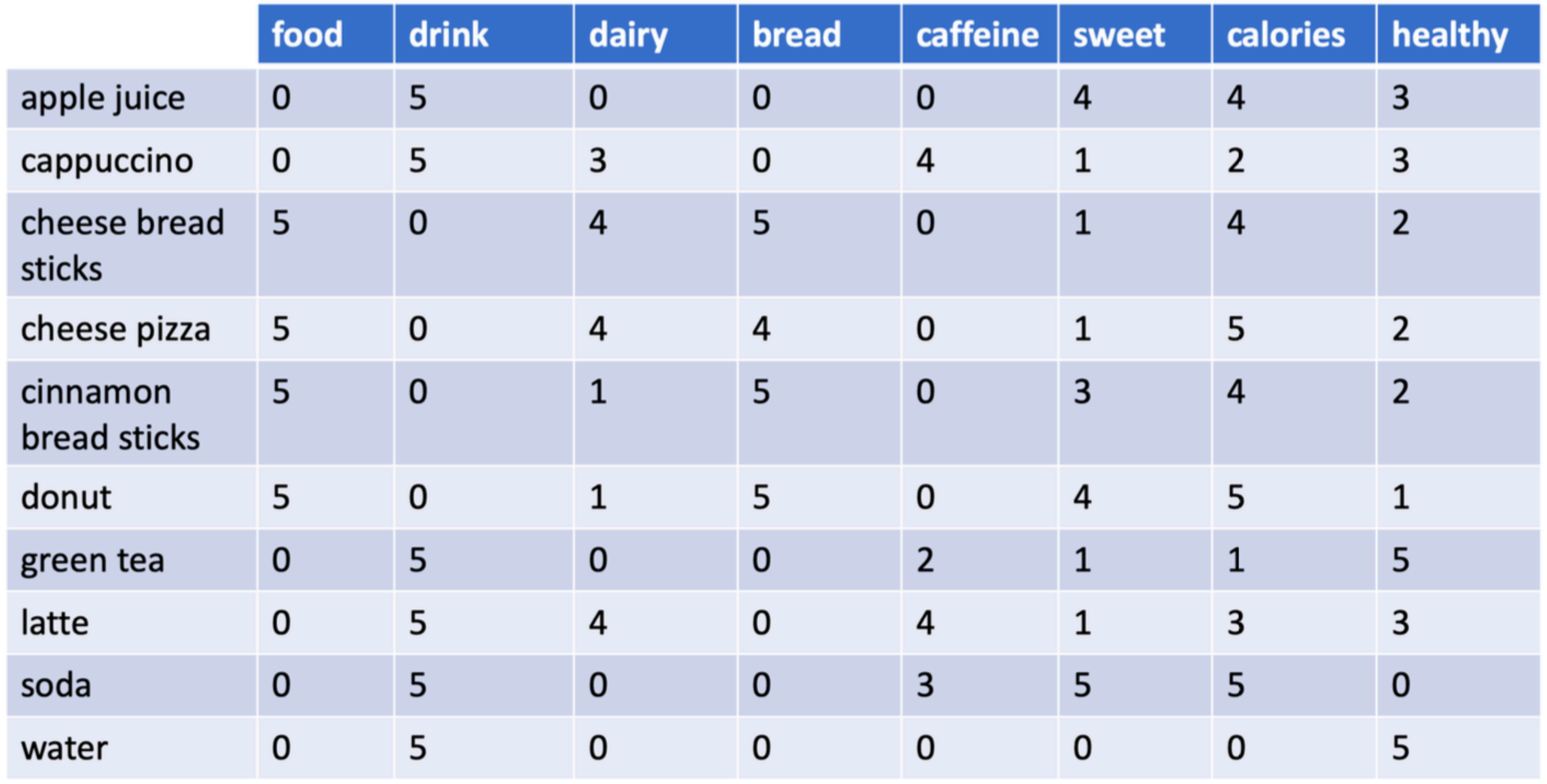

query
exact term lookup in inverted index

	apple	caffeine	cheese	coffee	drink	donut	food	juice	pizza	tea	water
latte	0	0	0	0	0	0	0	0	0	0	0
cappuccino	0	0	0	0	0	0	0	0	0	0	0
apple juice	**1**	0	0	0	0	0	0	**1**	0	0	0
cheese pizza	0	0	**1**	0	0	0	0	0	**1**	0	0
donut	0	0	0	0	0	**1**	0	0	0	0	0
soda	0	0	0	0	0	0	0	0	0	0	0
green tea	0	0	0	0	0	0	0	0	0	**1**	0
water	0	0	0	0	0	0	0	0	0	0	**1**
cheese bread sticks	0	0	**1**	0	0	0	0	0	0	0	0
cinnamon sticks	0	0	0	0	0	0	0	0	0	0	0

Figure 2.10 Vectors with one dimension per term in the inverted index. Every query on the left maps to a vector on the right, with a value of "1" for any term in the index that is also in the query, and a "0" for any term in the index that is not in the query.

Image from AI Powered Search, T. Grainger et al. (2022).

Word Embeddings / Dense Vectors



	food	drink	dairy	bread	caffeine	sweet	calories	healthy
apple juice	0	5	0	0	0	4	4	3
cappuccino	0	5	3	0	4	1	2	3
cheese bread sticks	5	0	4	5	0	1	4	2
cheese pizza	5	0	4	4	0	1	5	2
cinnamon bread sticks	5	0	1	5	0	3	4	2
donut	5	0	1	5	0	4	5	1
green tea	0	5	0	0	2	1	1	5
latte	0	5	4	0	4	1	3	3
soda	0	5	0	0	3	5	5	0
water	0	5	0	0	0	0	0	5

Figure 2.11 Word embeddings with reduced dimensions. In this case, instead of one dimension per term (exists or missing), now higher-level dimensions exist that score shared attributes across items such as "healthy", contains "caffeine" or "bread" or "dairy", or whether the item is "food" or a "drink".

Image from AI Powered Search, T. Grainger et al. (2022).

Word Embeddings

- ➔ Word embeddings allow for the representation of words
  - ➔ as low-dimensional
  - ➔ numerical dense vectors
  - ➔ that are learnt from data,
  - ➔ resulting in a projection in a new space
  - ➔ where similarities and relationships between the original words are kept.

Transformers

- ➔ Static word embeddings map words with multiple senses into an average or most common-sense representation based on the training data used to compute the vectors.
    - ➔ The vector of a word does not change with the other words used in a sentence around it.
- ➔ A transformers is a neural network designed to explicitly take into account the context of arbitrary long sequences of text.
    - ➔ Transformer compute contextualized word embeddings, where the representation of each input token is conditioned by the whole input text.
    - ➔ The most popular contextualized word embeddings are learned with deep neural networks such as the Bidirectional Encoder Representations from Transformers (BERT).
- ➔ With fine-tuning, the parameters of a pre-trained language model can be updated for the domain data and target task.
    - ➔ Pre-training typically requires a huge general-purpose training corpus, and long and expensive computation resources.
    - ➔ On the other side, fine-tuning requires a small domain-specific corpus focused on the downstream task, affordable computational resources and few hours or days of additional training.

From “exact match” to “soft match”

- ➔ **Key point**: neural methods replace exact matching with soft matching.

- ➔ With traditional methods, soft matching is possible:
  - ➔ Stemming handles singular/plural and different tenses.
  - ➔ Synonyms expand soft match possibilities.
  - ➔ Query expansion techniques can add context to the query by adding new terms.

- ➔ With neural methods, soft match is central:
  - ➔ Word embeddings capture semantic similarities, allowing nuanced understanding of related terms.

Retrieval Architectures

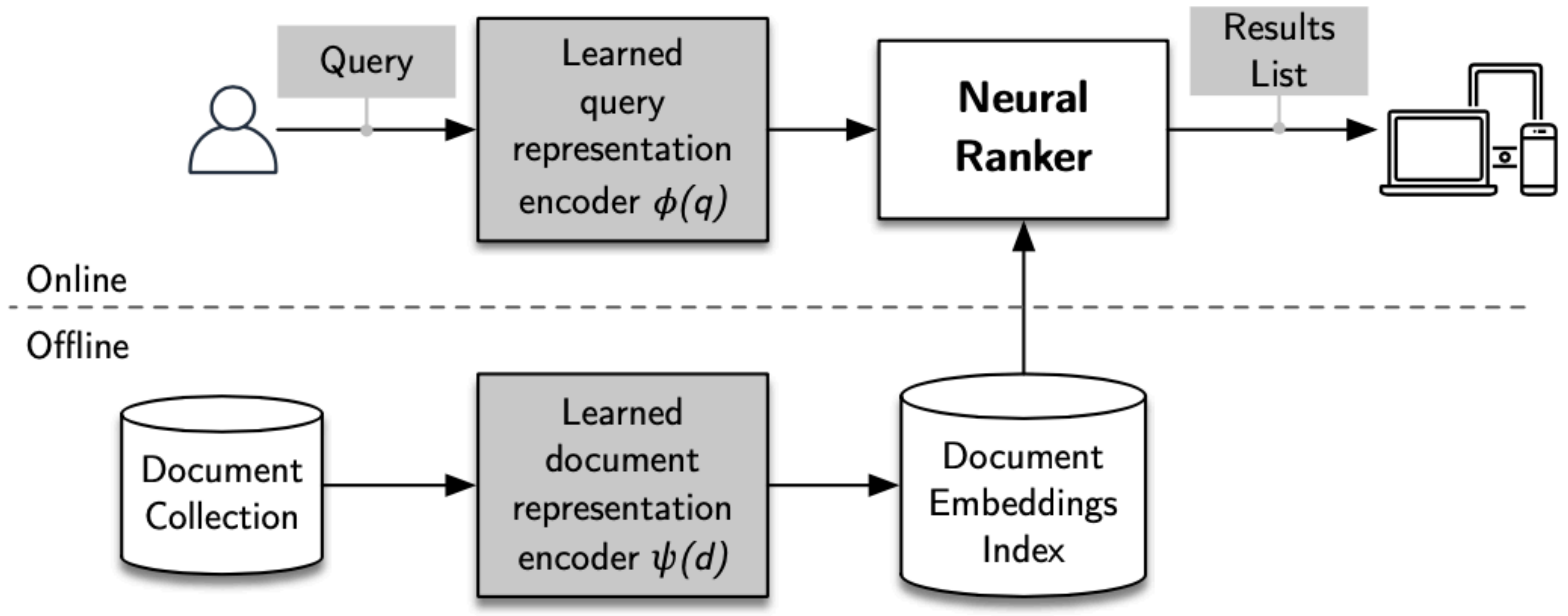

Query
Learned query representation encoder $\phi(q)$
Neural Ranker
Results List
Online
Offline
Document Collection
Learned document representation encoder $\psi(d)$
Document Embeddings Index

Figure 8: Dense retrieval architecture for representation-focused neural IR systems.

Neural IR Summary

- ➔ Deep learning methods greatly improve soft matching in document ranking.
- ➔ Pre-trained large language models, avoid the need for vasts amounts of training data to implement effective document retrieval solutions.

- ➔ However, a search system needs to support both <u>soft and hard</u> matching.
- ➔ Retrieval architectures need to combine traditional methods with neural IR methods.
  - ➔ E.g., first-stage neural selection (i.e. “semantic expansion”), followed by a standard retrieval stage over the selected documents.

Solr Dense Vector Search

Vector Search Systems

- ➔ Also called “vector databases”.
- ➔ In dense retrieval systems, document embeddings are pre-computed, thus the need for storing and searching through these document embeddings.
- ➔ Given a query, represented as a dense vector, instead of computing the distance between this vector and all document dense vectors (too expensive), the strategy is to use approximate nearest neighbor search.

- ➔ Solr (since version 9) includes support for Dense Vector Search.
- ➔ https://solr.apache.org/guide/solr/latest/query-guide/dense-vector-search.html

Dense Vector Search in Apache Solr

- Solr's Dense Vector Search adds support for indexing and searching dense vectors.
- Uses deep learning to produce vector representations for queries and documents.
- Dense Vector Representation:
  - Traditional tokenized inverted index vs. dense vector representation.
  - Distills approximate semantic meaning into a fixed number of dimensions.
  - Lower dimensionality compared to sparse vectors, offering efficiency.
- Dense Retrieval:
  - Explores the strategy for efficient retrieval using navigable small-world graph.
  - Provides approximate nearest neighbor search for high-dimensional vectors.

Dense Vector Indexing Configuration

- A new field type is added — DenseVectorField:
    - Field type designed for dense vector search.
    - Indexing and searching dense vectors of float elements.
- Example configuration in schema using vectorDimension and similarityFunction.
- Advanced Configuration:
    - Configuring advanced parameters like knnAlgorithm, vectorEncoding, hnswMaxConnections, and hnswBeamWidth.
    - Customizing the codec format and hyper-parameters of the HNSW algorithm.

Semantic Search Tutorial

- A new tutorial is available in GitLab > PRI > Tutorials > Semantic Search.
- Based on the SIGARRA course collection.
- Indexing:
  - Uses Hugging Face sentence transformers for document encoding.
  - Adds a new field for storing document vectors.
- Retrieval:
  - Encode query using same model.
  - Uses Solr's k-nearest neighbors query parser to retrieve similar documents.
- Expected exploration for M3.

Vector Search Open-Source Frameworks

- ➔ Milvus, https://milvus.io
- ➔ Vespa, https://vespa.ai
- ➔ Qdrant, https://qdrant.tech
- ➔ pgvector (PostgreSQL extension), https://github.com/pgvector/pgvector

- ➔ Vector Search benchmarks
  - ➔ http://ann-benchmarks.com

References

Relevant Search: With applications for Solr and Elasticsearch
Doug Turnbull and John Berryman
Manning, 2016

Solr in Action
Trey Grainger and Timothy Potter
Manning, 2014