University of Udine

Dense Retrieval

Fabio Ionut Ion
157274@spes.uniud.it

January 7, 2026

Outline

Background

Vector Space Model

The foundation is the Vector Space Model:

- t**-dimensional** (sparse) vector space, where each dimension corresponds to a single term
- A **document** $d_j = (w_{1j}, \ldots, w_{tj})$ is a **vector** in the vector space
- The weights w_{ij} are computed using **tf-idf**
- Similarity between vectors (*i.e. cosine or dot product*)
- Treat the query as a document and compute the **similarity**:

$$\mathrm{sim}(d_j, q) = \frac{\sum_{i=1}^{t} (w_{ij}\, w_{iq})}{\sqrt{\sum_{i=1}^{t} w_{ij}^2}\sqrt{\sum_{i=1}^{t} w_{iq}^2}}$$

Sparse vs Dense

Sparse

Vectors:

- High-dimensional (i.e. a lot of zeros)
- Dimension $\iff$ term

Matching:

- Only exact terms
- No synonyms (ex. `"ML"` $\neq$ `"machine learning"`)

Dense

Vectors:

- Low-dimensional (i.e. very few zeroes)
- Dimension $\iff$ concept

Matching:

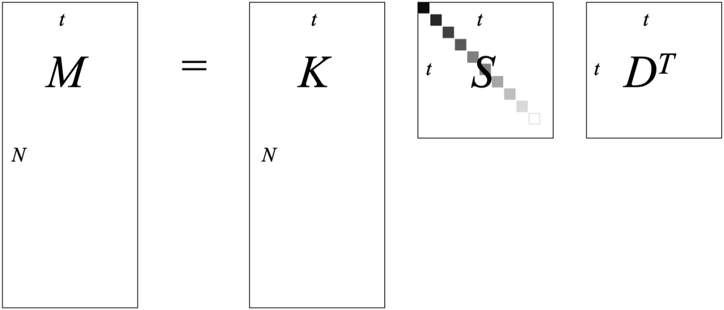
- Semantic similarity
- Synonyms (ex. `"AI algorithms"` $\approx$ `"neural networks"`)

Latent Semantic Indexing (LSI)

In the late 80s, `Latent Semantic Indexing (LSI)` was proposed:

- Interesting idea to move towards a dense space
- The index is reduced, resulting in a more *semantic* vector space, where each dimension corresponds to a concept instead of a term
- The dimensionality reduction is done by Singular Value Decomposition (SVD)

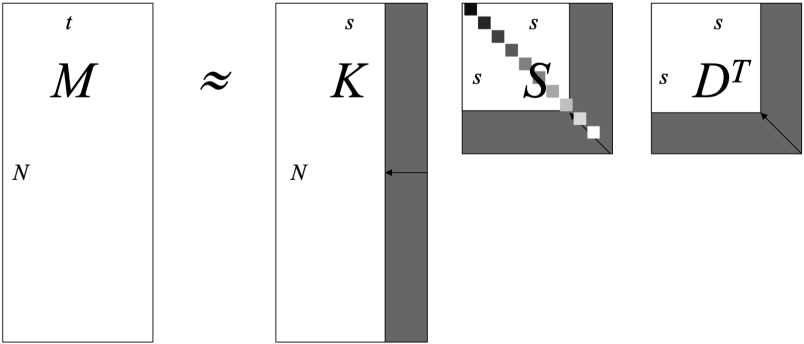
SVD visually



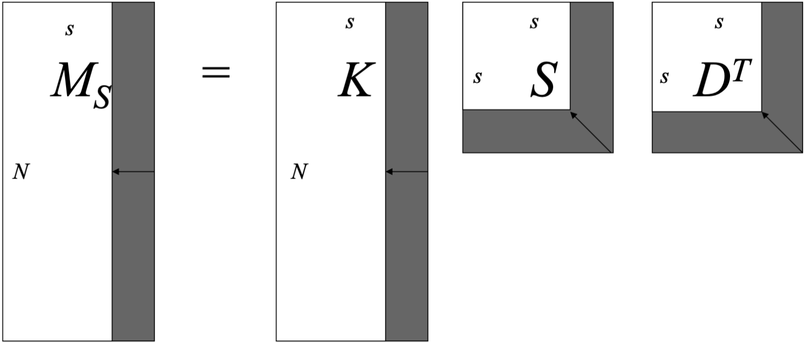
where:

- D^T is the transpose matrix of D
- S is a diagonal matrix, $t \cdot t$ with decreasing values
- K and D are the matrices of eigenvectors of $M \cdot M^T$ and $M^T \cdot M$, respectively

SVD visually



Zeroing out the smallest values in S



M_S is the best approximation of M

Latent Semantic Indexing (LSI)

- Cons
  - Linear SVD decomposition of the term–document matrix: it struggles to model complex semantic relations
  - Efficiency problems
  - Updates are difficult: new documents requires recomputing the SVD

- These limitations motivated further research in the area and a new paradigm was proposed: Dense Retrieval

Dense Retrieval

Dense Retrieval keeps LSI's core idea of a latent space, but implements it with neural networks!

- Low-dimensional space, dimension $\Longleftrightarrow$ concept, synonyms, semantic meaning, etc.
- *Neural networks* (not SVD!) create low-dim. vectors (also called **dense vectors** or **embeddings**)
- Query and document vectors
- Learned from labeled data
- Relevance measured by **semantic similarity** between query–document dense vectors

Informal definition

A neural network is a **learned function** that maps text to a vector

Hmm…alright, but what do we mean by a learned function?

- Intuitively, a learned function is not a formula that can be defined *a priori* (e.g. tf-idf)
- We provide relevance examples, and the neural network automatically adapts the function so that relevant query–document pairs are close in the vector space

The function must have these characteristics:

- texts with **similar meaning** are mapped to **similar vectors**
- **irrelevant** documents are mapped **far** from the query
- small **textual variations** (e.g. rephrase, typo, synonyms) do not change meaning too much

The function is **learned from examples**: **pairs** (query, $\{d_i^+\}$).
For each query a **list of positive documents** is given
(NOTE: binary relevance for simplicity!)

- Why **pairs** (query, $\{d_i^+\}$)?
  - Relevance is defined with respect to a query (a document alone has no notion of relevance!)
  - The neural network learns which documents are relevant to a query
- Why a **list of positive documents**?
  - A query represents an information need, which can be satisfied by multiple relevant documents
  - Multiple (positive) documents allow the model to learn the shared semantic meaning
  - Otherwise, we would not know which similarities between docs matter for relevance

Dense Retrieval uses neural networks to learn query and document representations (i.e. dense vectors) such that:

- queries and documents live in the same latent semantic space
- distance corresponds to relevance
- ranking emerges from similarity

PLMs revolution

Challenge: How do we build dense retrievers?

There are many types of neural network architectures. For dense retrieval, we need an architecture that is particularly good at understanding text meaning...

Challenge: How do we build dense retrievers?

There are many types of neural network architectures. For dense retrieval, we need an architecture that is particularly good at understanding text meaning...

The answer came with Pretrained Language Models (PLMs):

- PLMs (e.g. *BERT*) are neural networks of the same family as those used in modern large language models (LLMs).
- Trained on huge text corpora (e.g. Wikipedia, ...) to learn general language knowledge (*pretrain* phase)
- Then adapted to dense retrieval, i.e. encoding queries and documents into dense vectors (*fine-tune* phase)

Keypoint: **Dense vectors learned by PLMs**

Why PLMs matter

A generic NN is not optimal: dense retrieval requires models that deeply understand language!

PLMs changed this by:

- understanding **word meaning from context**
- capturing **semantic relations** (synonymy, paraphrases)
- producing dense vectors suitable for **similarity-based retrieval**

Thanks to pretraining, PLMs start fine-tuning for dense retrieval with representations that already encode meaning:

- *Pretraining* teaches the model the language
- *Fine-tuning* teaches it how to use that knowledge for retrieval

Formulation of Dense Retrieval

We want to model the **semantic interaction** between queries and documents based on the representations learned in the **latent semantic space**. Intuitively:

- Query q is mapped to a dense vector $\phi(q) \in \mathbb{R}^l$
- Document d is mapped to a dense vector $\psi(d) \in \mathbb{R}^l$

(where $\mathbb{R}^l$ is the latent space and $\phi(\cdot)$ and $\psi(\cdot)$ are PLM-based encoders)

Then, the **relevance score** is computed by a similarity function (e.g. *inner product* or *cosine*) between these dense vectors:

$$\mathrm{Rel}(q, d) = f_{\mathrm{sim}}\big(\phi(q), \psi(d)\big)$$

High relevance $\Rightarrow$ q and d embeddings are similar $\Rightarrow$ semantics are very similar

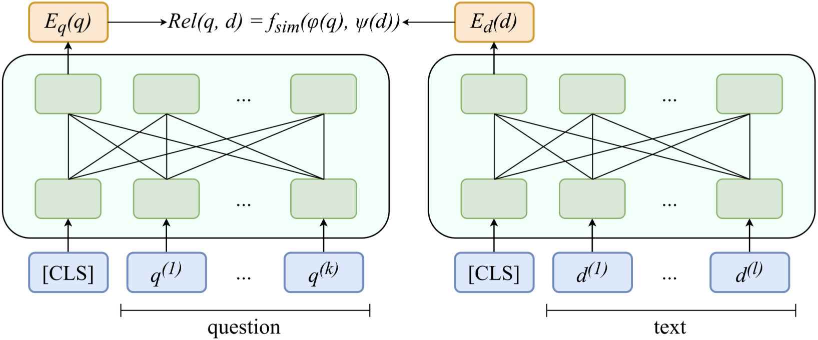
Architectures

In dense retrieval, relevance is modeled using two mainstream architectures: bi-encoders and cross-encoders.

They are typically combined:

- bi-encoders maximize recall
- cross-encoders maximize precision

Bi-encoder: How It Works I



The bi-encoder architecture uses two separate PLM encoders.

1. An encoder that maps queries to dense vectors
2. An encoder that maps documents to dense vectors
3. The relevance score is computed via a similarity function

Bi-encoder: How It Works II

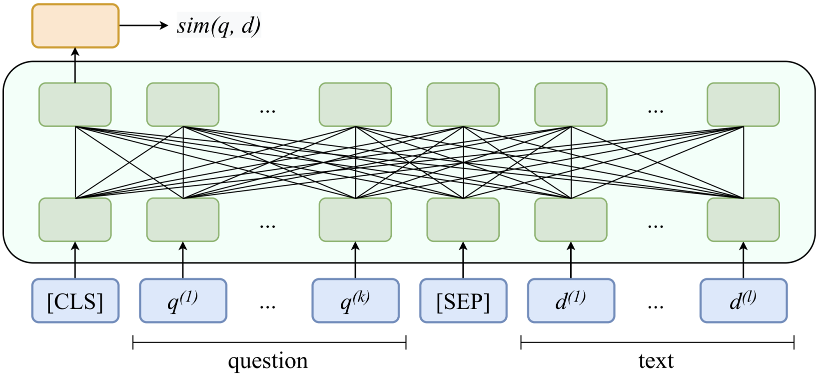
Few notes:

- All document embeddings are computed once and stored (during indexing)
- At query time, the query embedding is computed
- Relevance scores are obtained by comparing the query embedding to the precomputed document embeddings

Naively, this would require computing a score for every document!

In practice, efficient retrieval techniques are used to avoid this overhead (e.g. *approximate k-nearest neighbor (k-NN) search*).

Cross-encoder: How It Works I



The cross-encoder architecture uses a single PLM encoder.

1. Takes query and document together as input
2. Outputs a single relevance score

Cross-encoder: How It Works II

The cross-encoder does not produce independent embeddings for queries and documents.

They are typically used in a second-stage re-ranking step, after candidate documents have been retrieved by a bi-encoder.

Summary

- Original vector space model relies on sparse, term-based representations
- LSI was a first attempt: semantic spaces via SVD, but with limitations (linear, expensive to update)
- Dense Retrieval uses neural networks to learn a semantic space from labeled query–document pairs
- Pretrained Language Models (PLMs) made dense retrieval practical
  - Understand language context
  - Scale to millions of documents
- Modern systems for dense retrieval combine:
  - Bi-encoders for efficient retrieval
  - Cross-encoders for accurate re-ranking

Thank you!

Bibliography

Stefano Mizzaro (2025a). *Slide (W)IR: IR Models – 1, Lecture 6*. pp. 16–37.

– (2025b). *Slide (W)IR: IR Models – 3, Lecture 8*. pp. 13–27.

Wayne Xin Zhao et al. (Feb. 2024). "Dense Text Retrieval Based on Pretrained Language Models: A Survey". In: *ACM Trans. Inf. Syst.* 42.4. ISSN: 1046-8188. DOI: 10.1145/3637870. URL: https://doi.org/10.1145/3637870.