Navigating Neural Search: Avoiding Common Pitfalls

Jo Kristian Bergum @jobergum

@vespaengine

Vespa.ai

An open-source platform for low latency computations over large, evolving data

Apache 2.0 Licensed
https://github.com/vespa-engine/vespa

Pulls 10M+

- ➔ Search, filter and rank structured and unstructured data
- ➔ Dense and sparse representations
- ➔ Scalable in any dimension
- ➔ Multiphase retrieval & ranking
    - ◆ Dense HNSW - nearest neighbor search
    - ◆ Sparse WAND
    - ◆ Hybrid combinations
- ➔ Tensors and ML are first class citizens
- ➔ Real-time Indexing and true partial updates
- ➔ Elastic content scalability (no pre-sharding)

This talk

- Highlight common pitfalls
- Pre-trained Language Models (PLM)
- Quick overview of neural search using PLM
    - Three neural models built on pretrained language models (PLM)
- Text embedding models and embedding retrieval

Not in this talk

- Retrieval Augmented Generation (RAG)
- Generative Large Language Models (GPT, LLAMA)

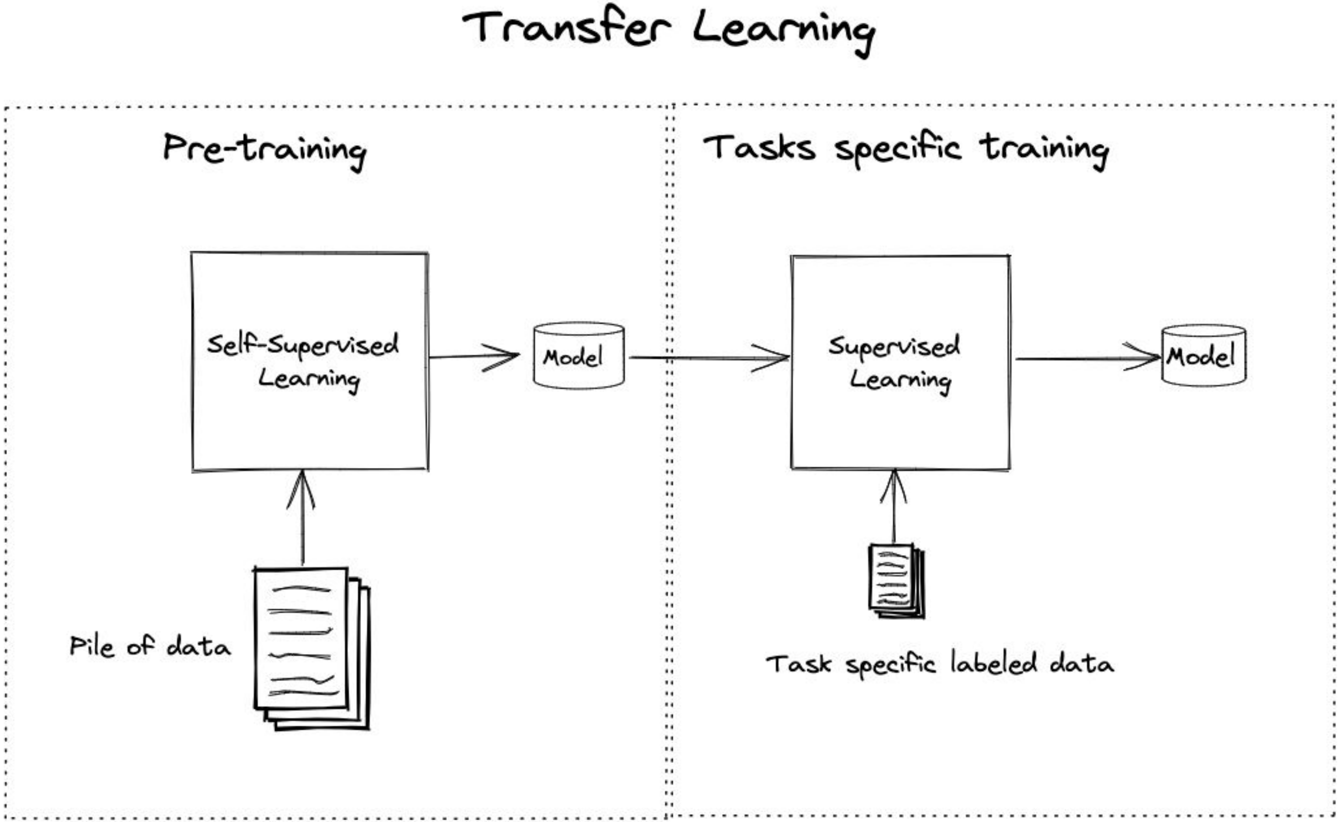
Pretrained Language Models (PLM)

- Attention is All you Need (Google 2017)
- BERT (Bidirectional Encoder Representations from Transformers)
- Trained using masked word language objective

The cat sits on the [MASK] looking at the [MASK]

- Masking objective is genius - Enables self-supervision with large corpuses of text
- Pre-trained model weights uses as starting weights for downstream tasks
  - Search
  - Classification
  - And more

Transfer Learning 101

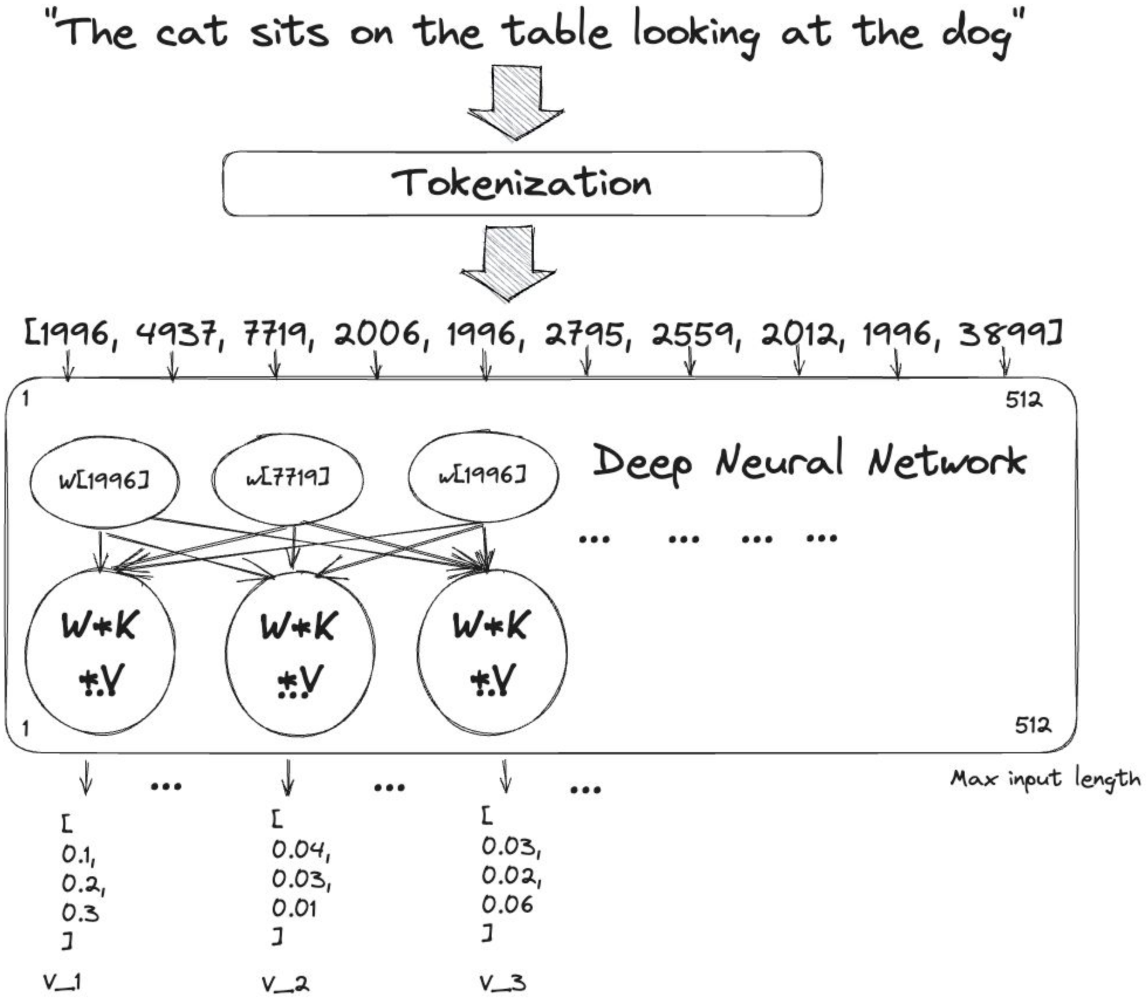


Language Model

A tokenizer + fixed vocabulary

A deep neural network architecture

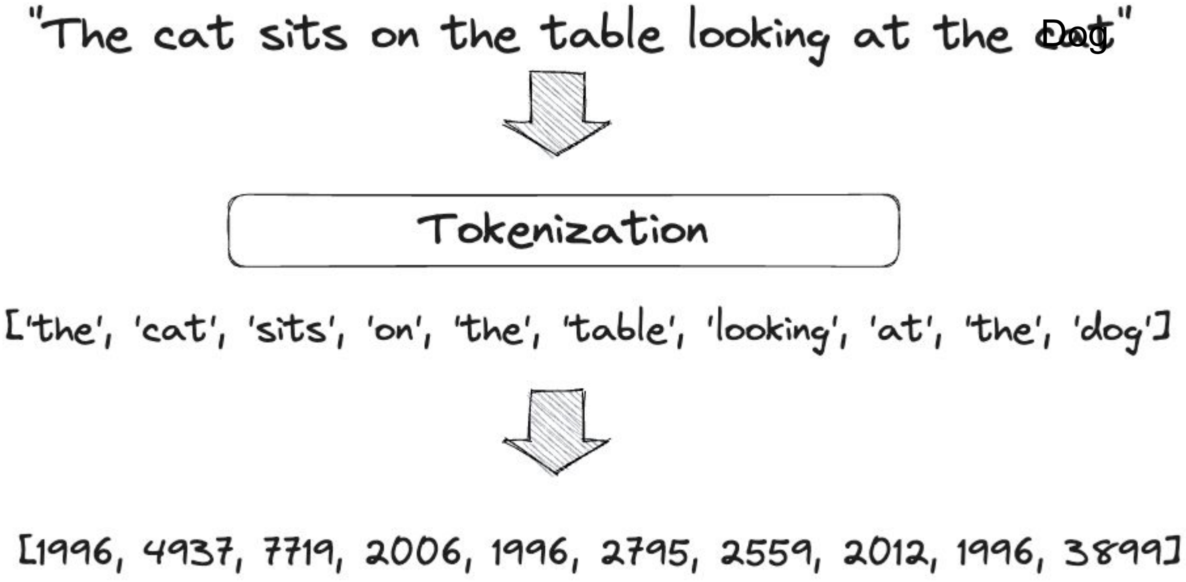
Small, medium, large, xxx large?



LM Tokenization

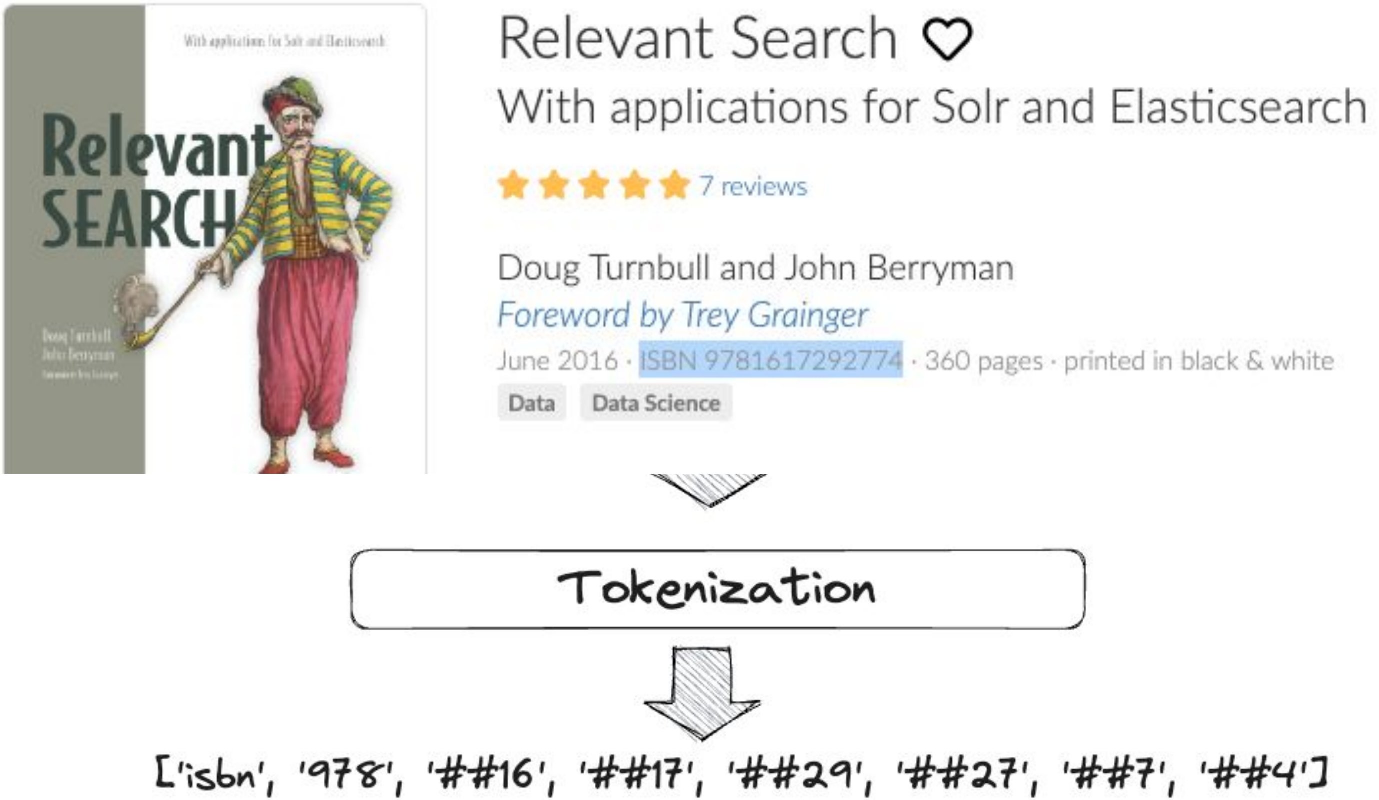
Happy Path Tokenization

10 words maps to 10 token ids



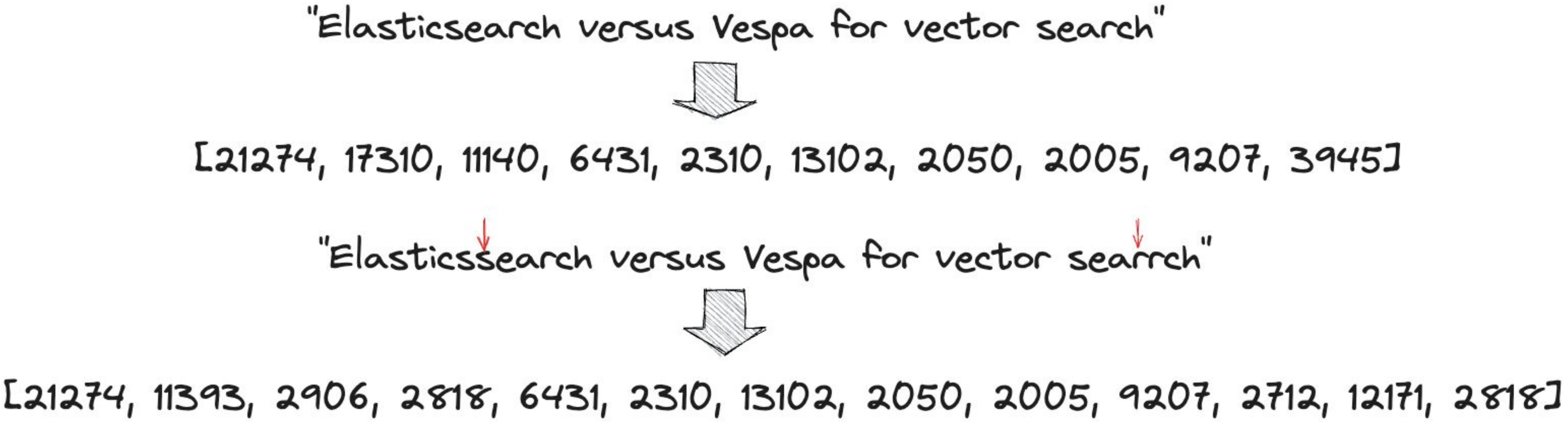
LM tokenization

- Different tokenizer implementations
- Tied to model
- Fixed vocabulary size
- Learned “word” embedding vectors per word in vocab
- Vocab fixed before pre-training of neural network weights



LM Tokenization

Are LM insensitive to spelling mistakes?



LLM tokenization impact vector representation

Variant and tokens	Top-3 retrieved words (vector search over WordNet)
annoyance => annoyance	frustration, anger, rage (0.91)
anoyance => ['an', '##oya', '##nce']	loyalty, consciousness, treasure (0.83)
annyoance => ['ann', '##yo', '##ance']	anniversary, old age, tendency (0.84)

WordNet® is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept.

LM tokenization

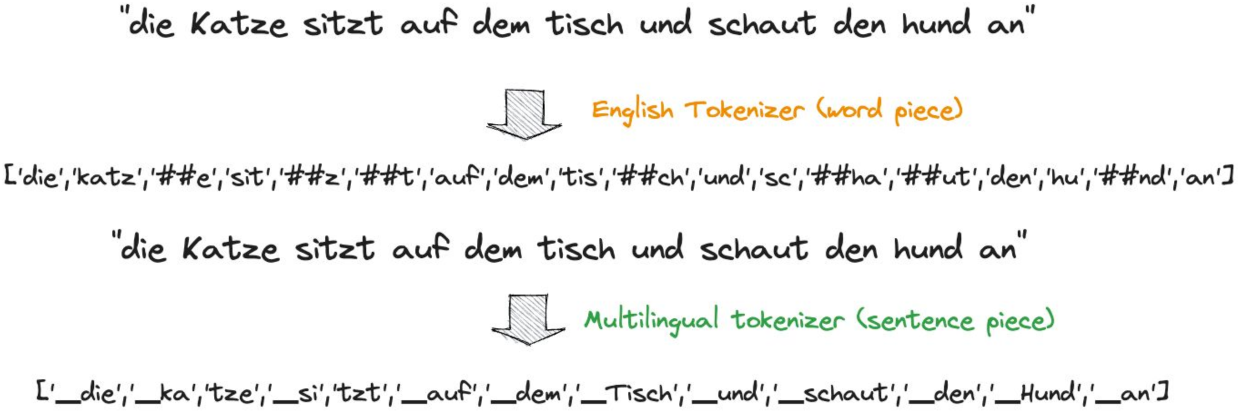
Linguistics and language matters !(?)

- Multilingual
- English

Not that many language specific LM models (except for English)

Don’t know newer words

- 2023 (202, ##3)
- Covid-19 (co,##vid,-, 19)
- GPT (gp, ##t)

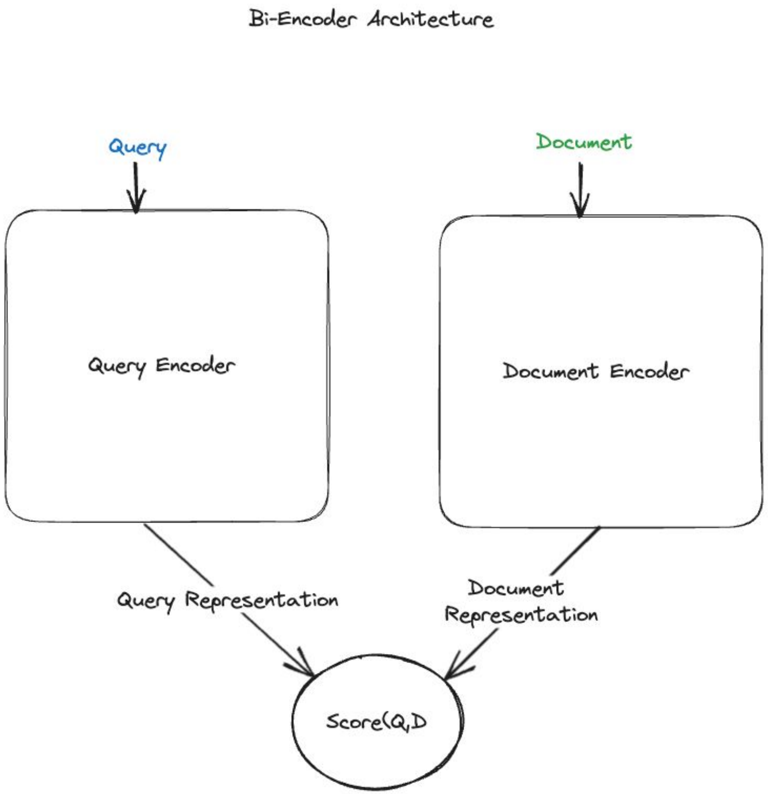


Applying LMs to search

Searching over data with sublinear complexity

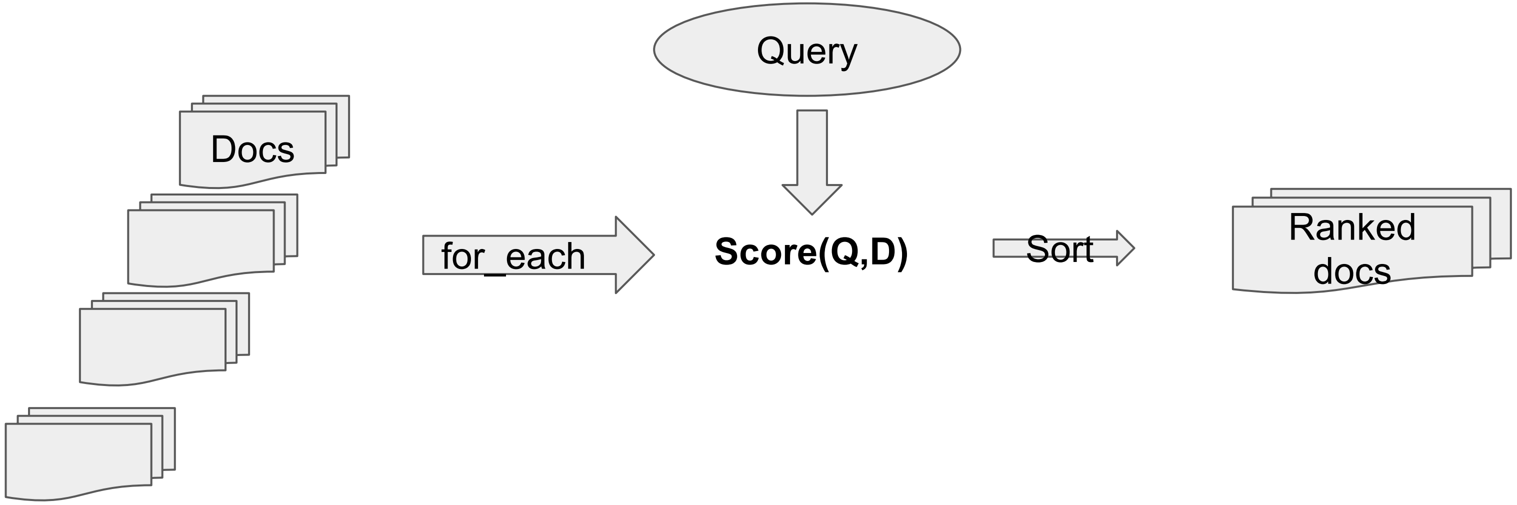
Conceptual representational model for retrieval

- Representation of queries and documents
    - So that relevant documents for a query is scored higher than irrelevant documents
- Dense/Sparse/Mixed
- Score(Q,D) complexity constraints
- Supervised (learned) versus unsupervised



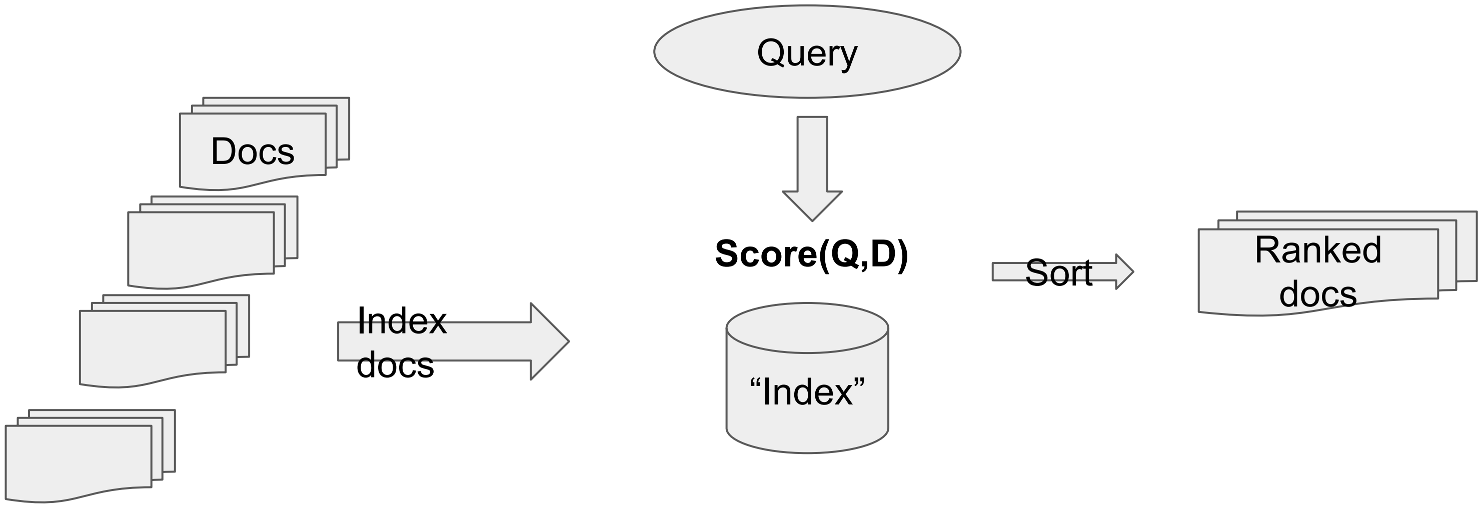
Motivation for representational approach

Avoid scoring all documents D in collection for a query Q



Motivation for representational approach

Avoid scoring all documents D in collection for a query Q



Make it more concrete

Logical representation versus physical implementation.

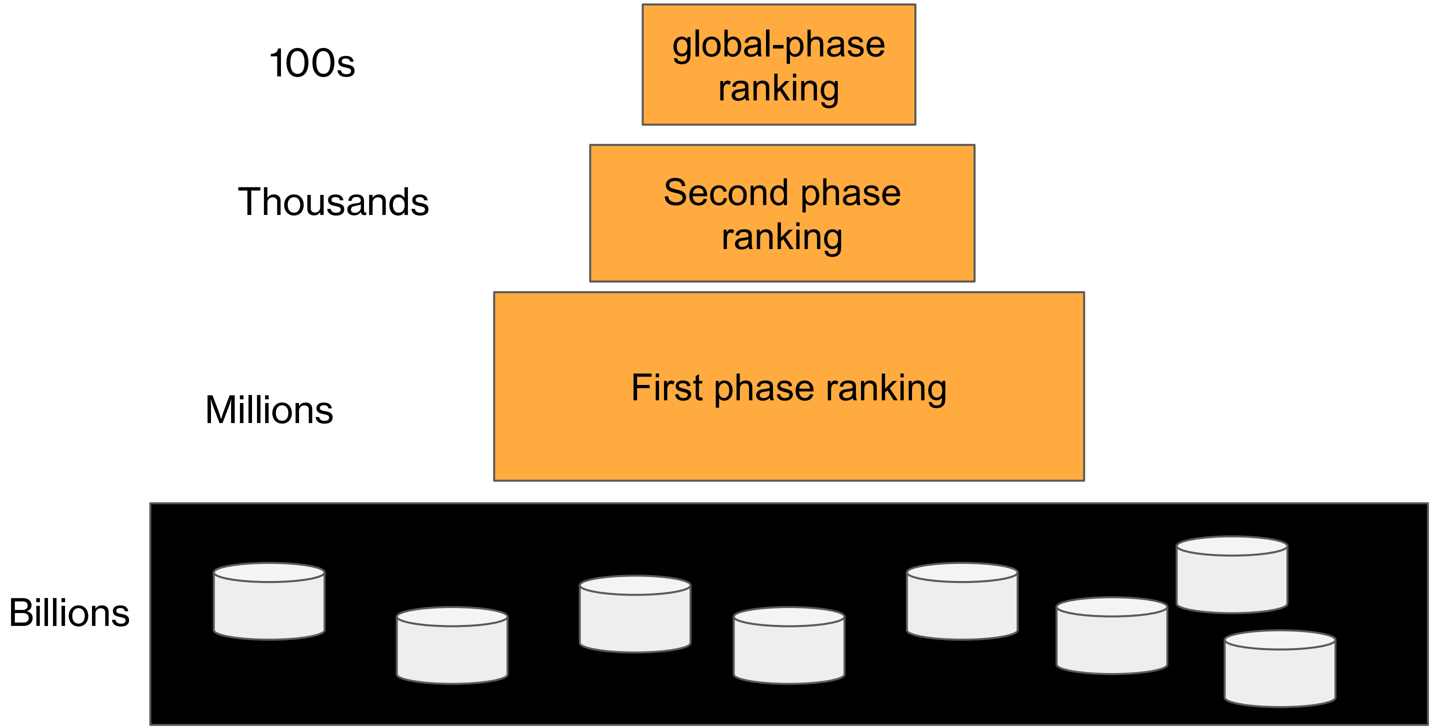
Accelerating scoring over sparse representations

- Build Inverted Index data structures
- Search accelerated with algorithms like WAND, MaxScore, BM-WAND++

Accelerating scoring over dense representations

- Build Vector Index (IVF, Quantization, HNSW, ++)
- Search accelerated with algorithms tied to vector index structure

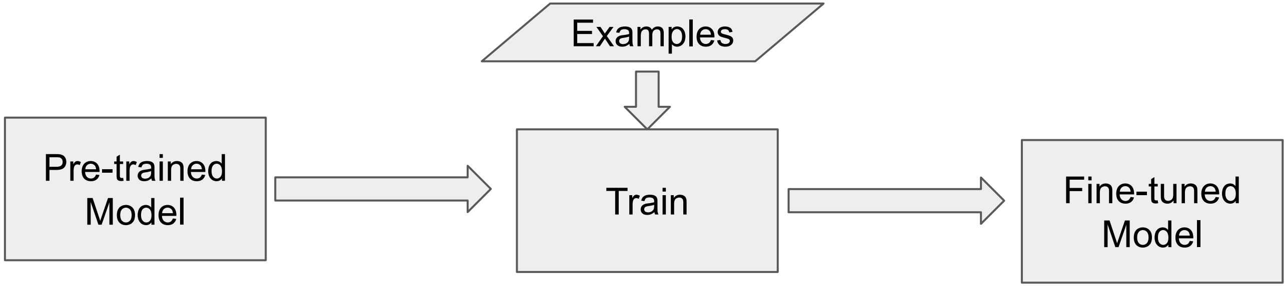
Also: Phased retrieval and ranking



3 Neural Methods for Search using LM

All methods require - **Labeled examples** - usually triplets

<query, relevant document, irrelevant document>



Cross-Encoder

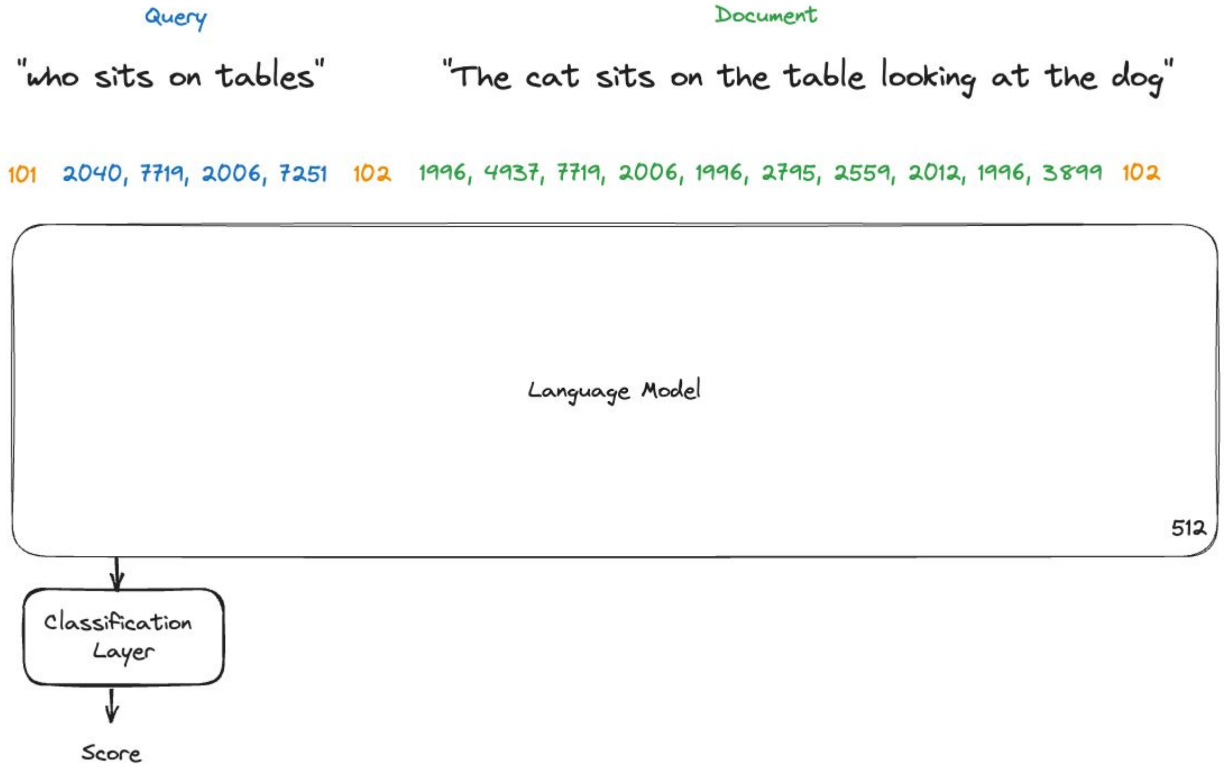
Encodes both query and document at the same time (cross)

all-to-all attention between all tokens in query and document

Most effective on IR benchmarks (nDCG)

High compute complexity (n^2)

No efficient way to "index"



Bi-Encoder

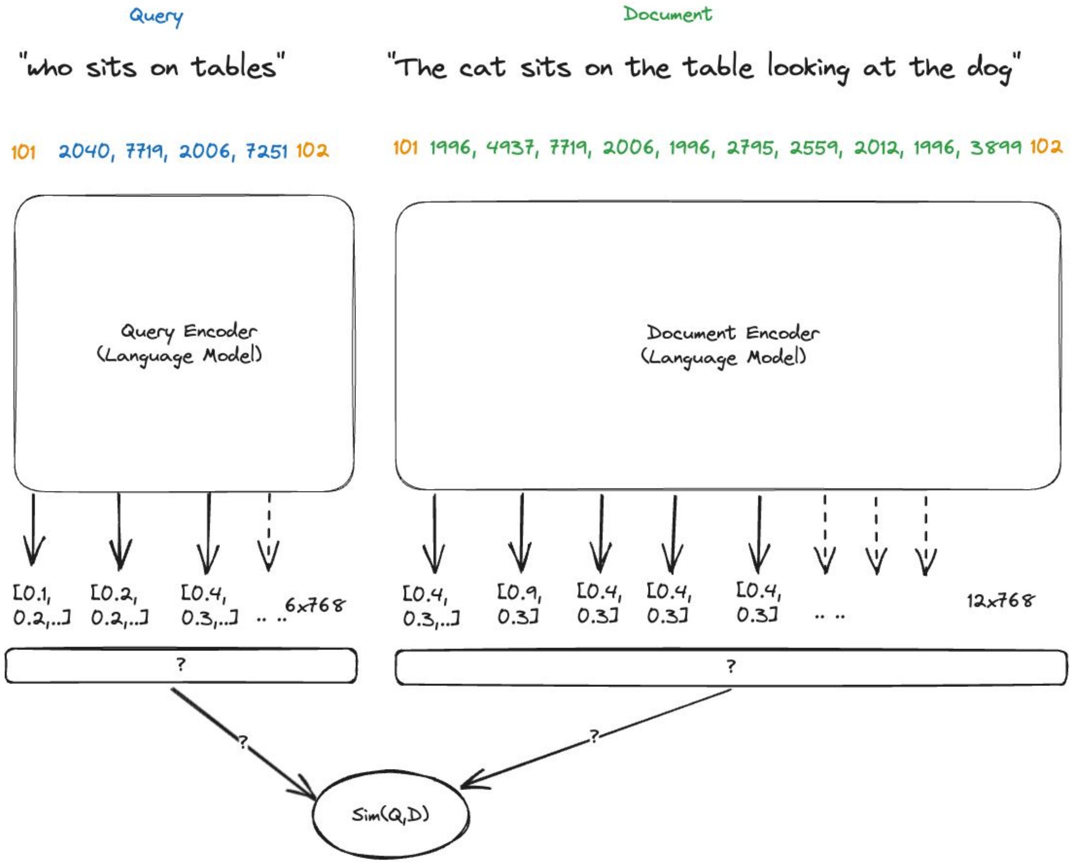
Encode queries and documents independently

No token level attention between query and document (no cross)

Enables indexing documents offline

Sim(Q,D):

- Dot product (sparse or dense)
- Cosine/Euclidean/Hamming/Many

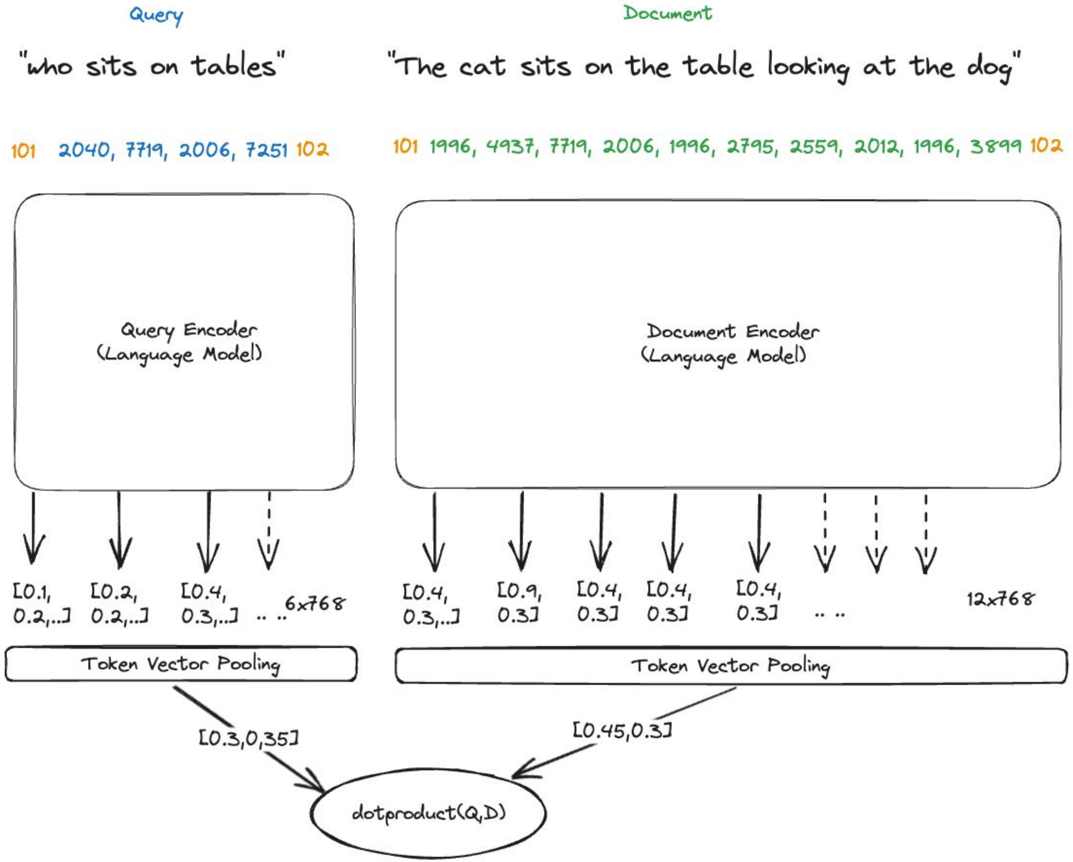


Bi-Encoder

Output Pooling

From a **token** vector representations to a vector representation of a sequence

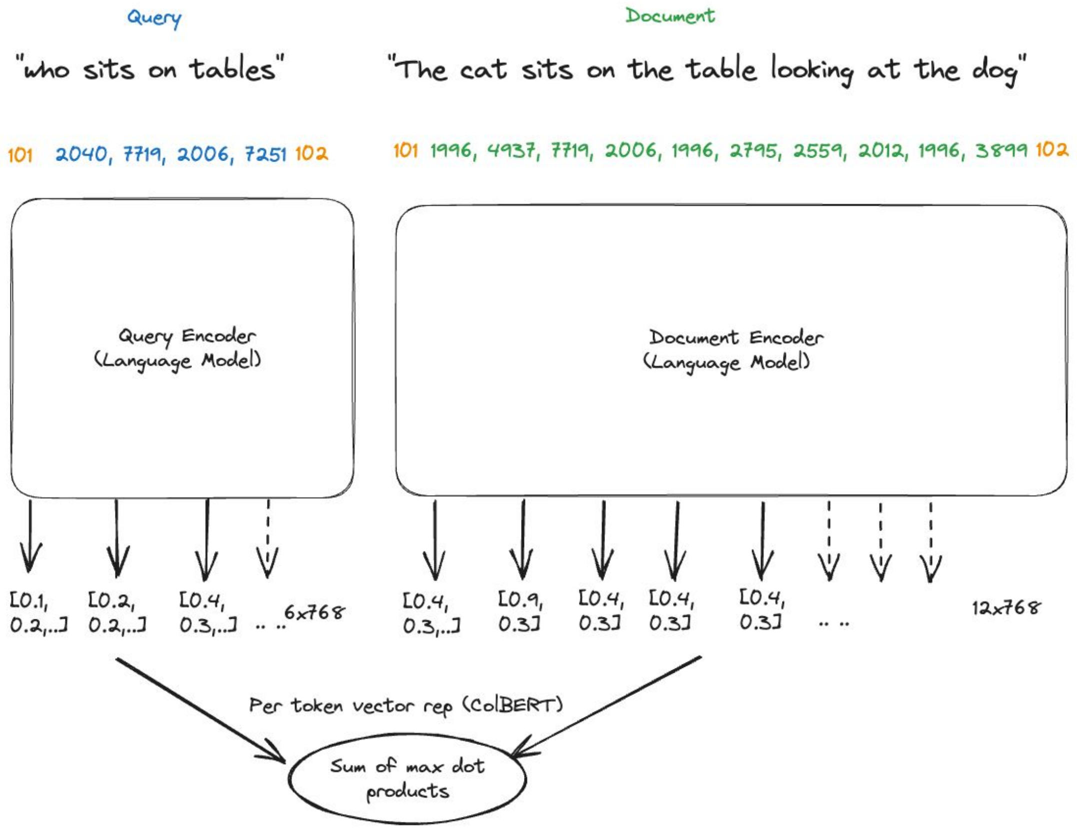
- Average?
- 101/CLS token?



Bi-Encoder adv

Learn token vectors instead of sequence vectors

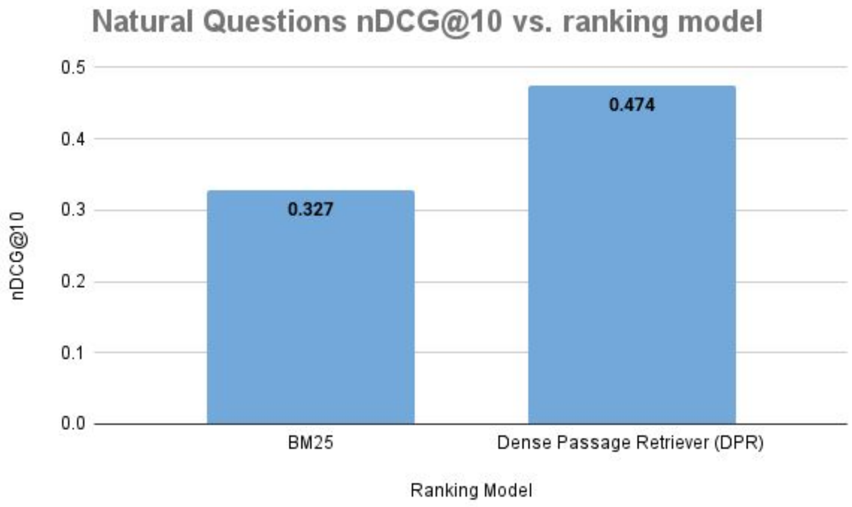
Not pooled

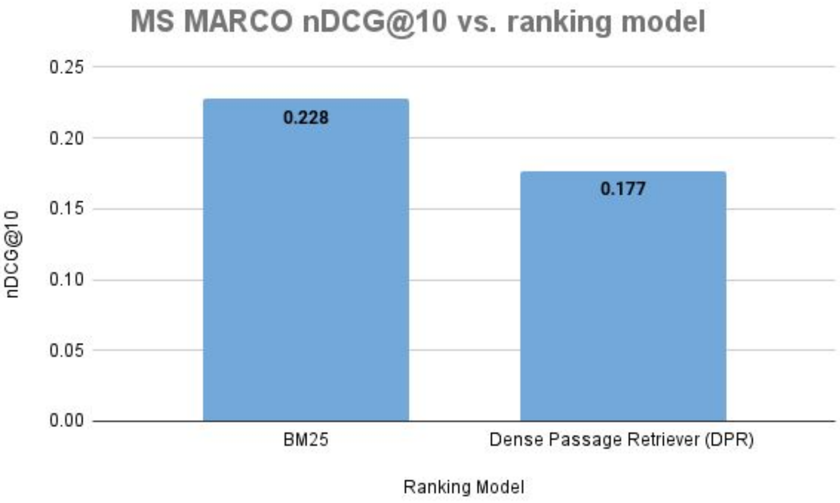


Learned representations - No better than the examples?

Remember: The representation of queries and documents are learned

- Your data might not look like the examples





Data the vector model was trained on

Photo by Vidar Nordli-Mathisen on Unsplash

Your data

Photo by Oskar Kadaksoo on Unsplash

TLDR; Neural Methods for Retrieval & Ranking

Accuracy versus cost

Model not better than the examples it was trained on

Explain/score interpretability difficult with pooled representations

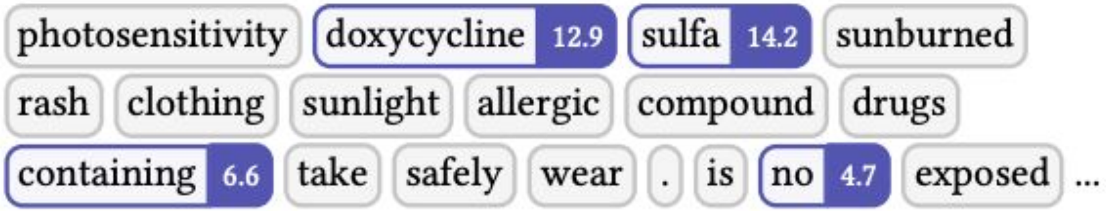
Introducing Neural Bag of Whole-Words with ColBERTer

https://arxiv.org/abs/2203.13088

does doxycycline contain sulfa

BERT tokenized (9 subword-tokens): 'does', 'do', '##xy', '##cy', '##cl', '##ine', 'contain', 'sul', '##fa'

ColBERTer BOW^2 *(30 saved vectors from 84 subword-tokens):*

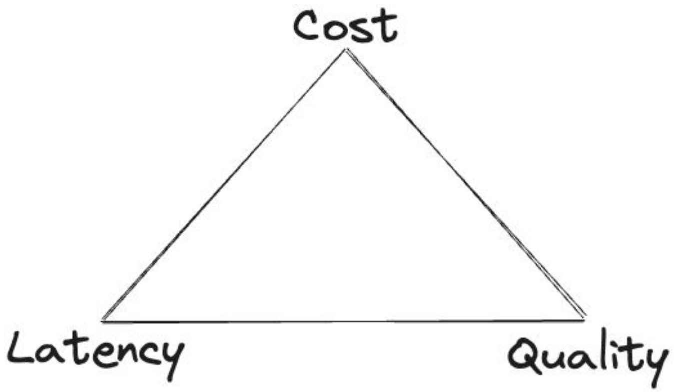


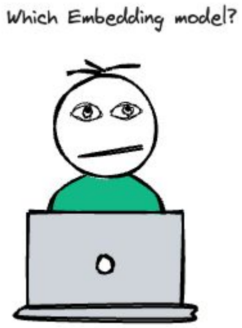
Fulltext: No doxycycline is not a sulfa containing compound, so you may take it safely if you are allergic to sulfa drugs. You should be aware, however, that doxycycline may cause photosensitivity, so you should wear appropriate clothing, or you may get easily sunburned or develop a rash if you are exposed to sunlight.

Figure 1: Example of ColBERTer's BOW^2 (Bag Of Whole-Words): ColBERTer stores and matches unique whole-word representations. The words in BOW^2 are ordered by implicitly learned query-independent term importance. Matched words are highlighted in blue with whole-word scores displayed in a user-friendly way next to them.

Off-the-shelf text-embedding models

- Size of model
- Embedding dimensionality
- Sequence length
- Quality/Accuracy (for your use case)
- Language capabilities
- Licence/Commercial use





MTEB (massive text embedding benchmark)

Great guide

Many different tasks

https://huggingface.co/spaces/mteb/leaderboard

Benchmark hacks?

Rank	Model	Model Size (GB)	Embedding Dimensions	Sequence Length	Average (56 datasets)
1	bge-large-en-v1.5	1.34	1024	512	64.23
2	bge-base-en-v1.5	0.44	768	512	63.55
3	gte-large	0.67	1024	512	63.13
4	gte-base	0.22	768	512	62.39
5	e5-large-v2	1.34	1024	512	62.25
6	bge-small-en-v1.5	0.13	384	512	62.17
7	instructor-xl	4.96	768	512	61.79
8	instructor-large	1.34	768	512	61.59

Embedding Retrieval

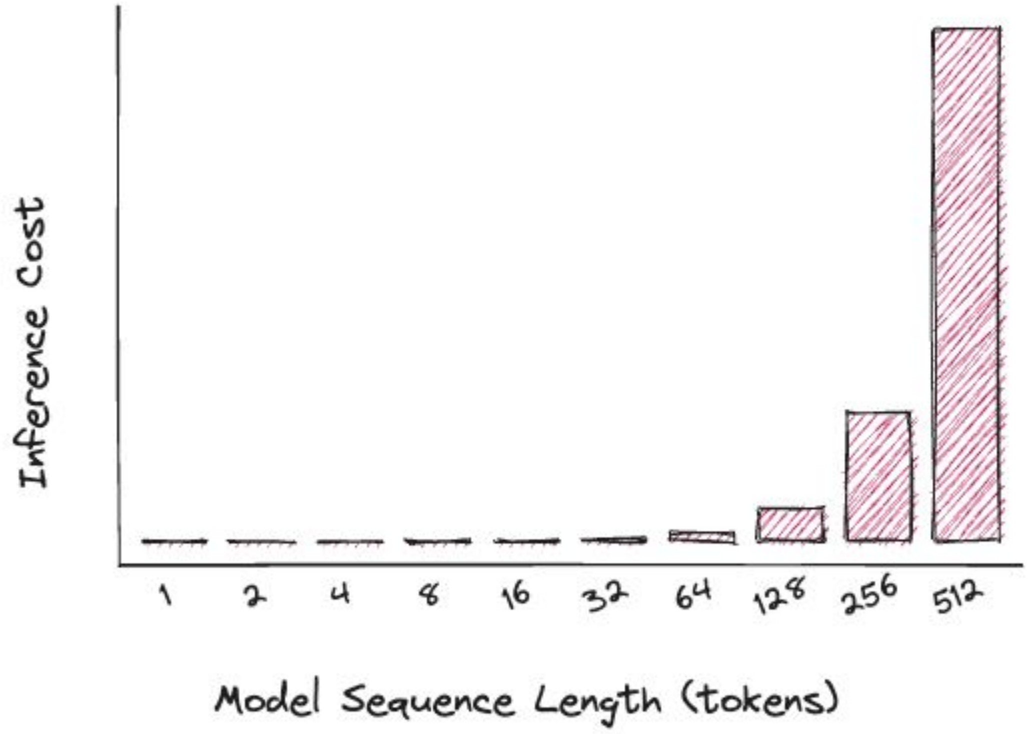
Embedding inference + Retrieval

Model size (GPU needed?)

Sequence length scaling

Dimensionality

- 1536 dims (4x cost of 384)
- Not 4x accuracy !



Vector Search

Brute Force Search Might Be All You Need?

Assume 64GB/s memory bandwidth

1M vectors with1536 dimensions using float is approx 6GB

Quiz: How many QPS can one node support at max?

The A in ANN

Approximate search instead of brute-force search

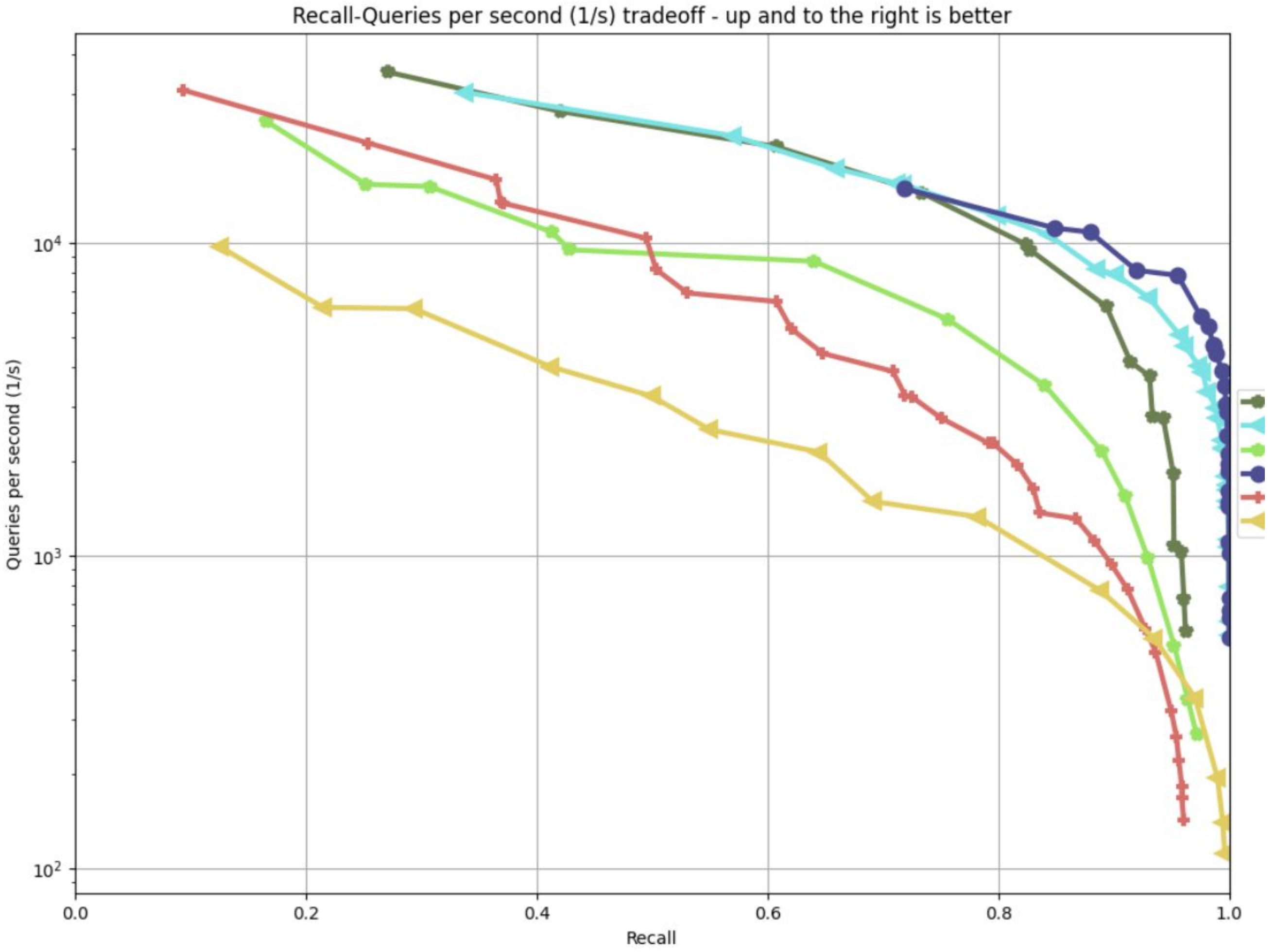
Speed up retrieval, by building an index, **sounds familiar**?

Many different ANN algorithms and associated tradeoffs

- Query speedup
- Quality (What is the error introduced by approximate search)
- Real-time (Mutatable, grow from to zero to N)
- Resource footprint, index build time

Exact and Approximate (recall@k)

Overlap@k is a better name for us working with search metrics



Impact of ANN choice & parameters on search quality

Our search quality metrics

- Recall (Are we finding all the relevant hits)
- Precision (Are we finding nothing but relevant?)

LADR

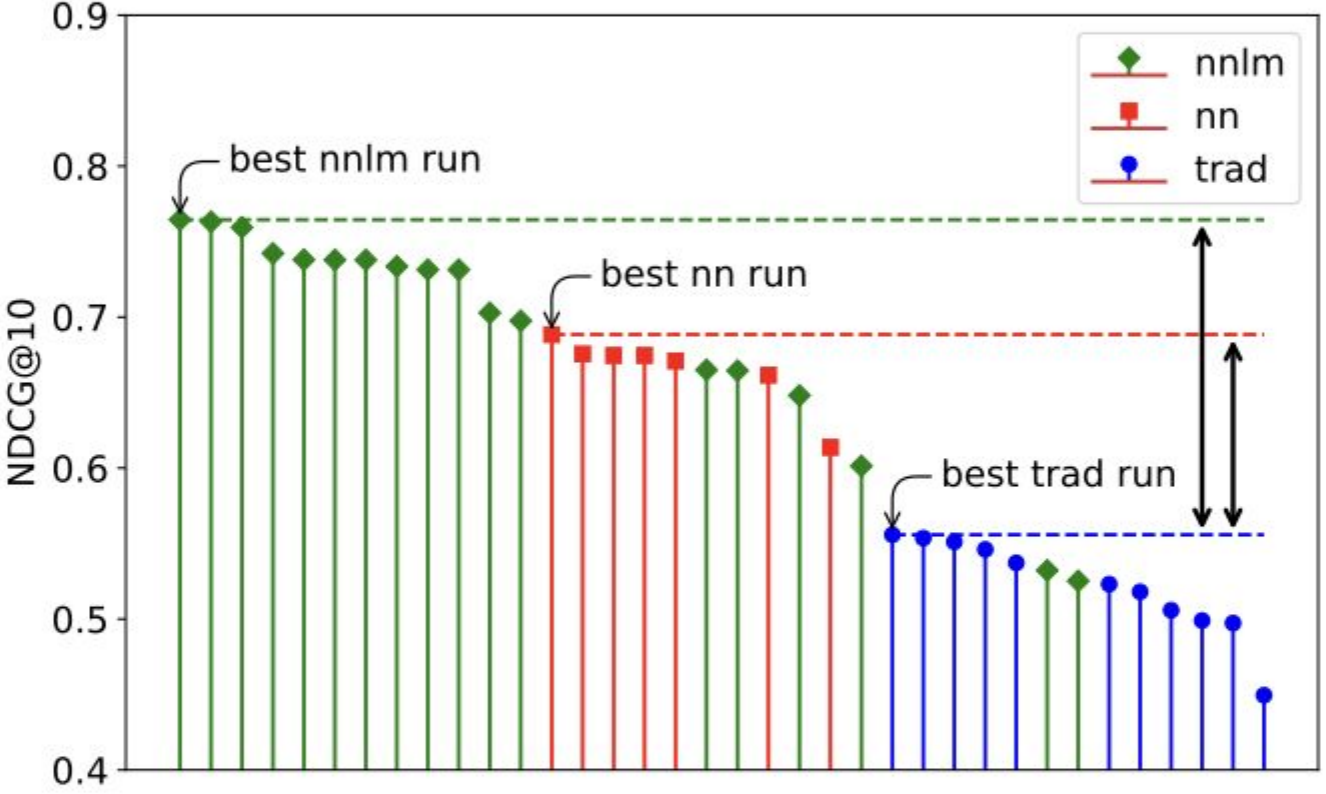
https://arxiv.org/abs/2307.16779, BM25 on DL19 is about 0.55 NDCG@10

	DL19 ~4ms		DL19 ~8ms		DL20 ~4ms		DL20 ~8ms	
Method	nDCG	R@1k	nDCG	R@1k	nDCG	R@1k	nDCG	R@1k
TAS-B (Exh.)	0.715	0.842	0.715	0.842	0.713	0.875	0.713	0.875
IVF [I]	0.374	0.414	0.474	0.536	0.503	0.559	0.579	0.677
ScaNN [S]	0.475	0.519	0.537	0.598	0.476	0.527	0.553	0.641
HNSW [H]	-	-	0.614	0.707	-	-	0.699	0.836
GAR [G]	0.543	0.540	0.688	0.755	0.568	0.594	0.684	0.796
Re-Ranking [R]	0.589	0.605	0.684	0.755	0.615	0.667	0.691	0.805
Proactive LADR	$^{IS}_{GR}$**0.690**	$^{IS}_{GR}$**0.771**	$^{ISH}_{GR}$0.730	$^{ISH}_{GR}$0.850	$^{IS}_{GR}$**0.691**	$^{IS}_{GR}$**0.807**	$^{IS}_{GR}$0.722	$^{IS}_{GR}$0.857
Adaptive LADR	-	-	$^{ISH}_{GR}$**0.738**	$^{ISH}_{GR}$**0.872**	-	-	$^{ISH}_{GR}$**0.739**	$^{ISH}_{GR}$**0.900**

max-links-per-node	neighbors-to-explore-at-insert	hnsw.exploreAdditionalHits	NDCG@10
16	100	0	0.5115
16	100	100	0.6415
16	100	300	0.6588
32	500	0	0.6038
32	500	100	0.6555
32	500	300	0.6609

Summarization of the HNSW parameters and the impact on NDCG@10.

As the table above demonstrates, we can reach the same NDCG@10 as the exact search by using `max-links-per-node` 32, `neighbors-to-explore-at-insert` 500, and `hnsw.exploreAdditionalHits` 300. The high `hnsw.exploreAdditionalHits` setting indicates that we could alter the index time settings upward, but we did not experiment further. Note the initial HNSW setting in row 1 and the significant negative impact on retrieval quality.



(b) Passage retrieval task

TLDR;

- Tokenization and vocabulary matters
- Language matters
- **Representations, representations, representations**
- Your data (queries and documents) might not match training examples

- Embedding inference
    - Sequence length
    - Dimensionality
- Embedding retrieval (vector search)
    - Brute force versus approximate

- Approximate Search Does Introduce Errors..

Resources

Lots on Blog.vespa.ai, for example

https://blog.vespa.ai/improving-zero-shot-ranking-with-vespa-part-two/

https://blog.vespa.ai/accelerating-transformer-based-embedding-retrieval-with-vespa/

Hated it? Tweet me

Jo Kristian Bergum jobergum