INFO-I590 Fundamentals and Applications of LLMs

RAG

Retrieval-Augmented Generation

Jisun An

Standard Prompting

- Combine a prompt template together with an input

Please answer this question:

I think Vin Diesel has been a voice actor for several characters in TV series, do you know what their names are?

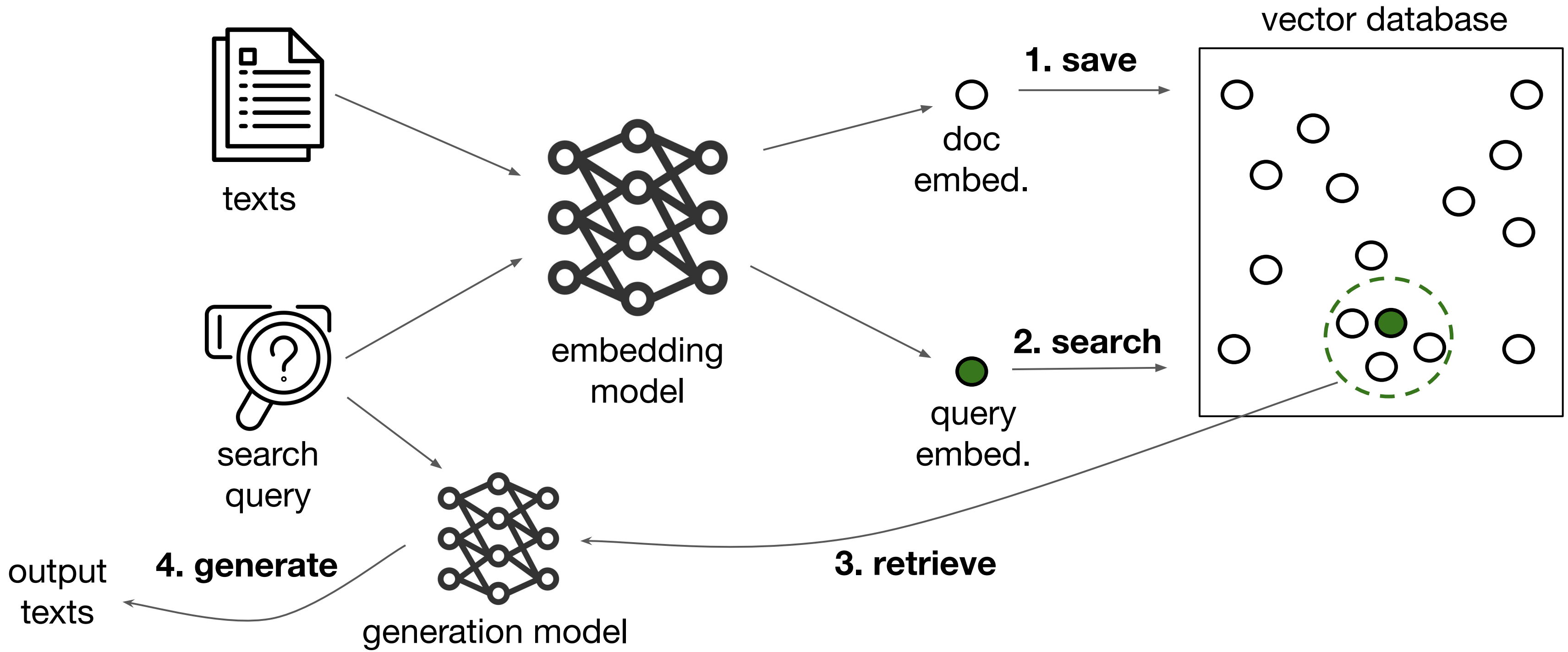
Problems

- Accuracy issues:
  - Knowledge cutoffs: parameters are usually only updated to a particular time
  - Private data: data stored in private text or data repositories not suitable for training
  - Learning failures: even for data that the model was trained on, it might not be sufficient to get the right answer

- *Verifiability issues*: It is hard to tell if the answer is correct (Hallucination)

Retrieval-Augmented Generation

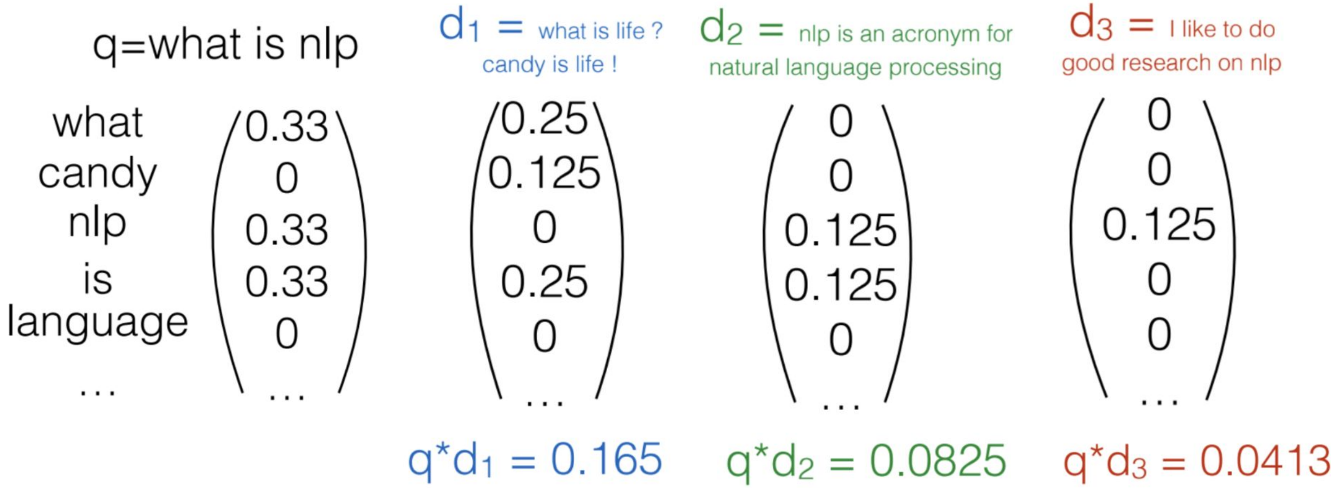
- Efficiently **retrieve** relevant passages based on a query
- **Generate** an answer using the retrieved passages



Sparse Retrieval

Sparse Retrieval

- Express the query and document as a sparse word frequency vector (usually normalized by length)



- Find the document with the highest **inner-product** or **cosine similarity** in the document collection

Term Weighting (See Manning et al. 2009)

- Some terms are more important than others; low-frequency words are often more important
- Term frequency - in-document frequency (TF-IDF)

$$\mathrm{TF}(t,d) = \frac{\mathrm{freq}(t,d)}{\sum_{t'} \mathrm{freq}(t',d)} \qquad \mathrm{IDF}(t) = \log\left(\frac{|D|}{\sum_{d' \in D} \delta(\mathrm{freq}(t,d') > 0)}\right)$$

$$\mathrm{TF\text{-}IDF}(t,d) = \mathrm{TF}(t,d) \times \mathrm{IDF}(t)$$

- BM25: TF term similar to smoothed count-based LMS

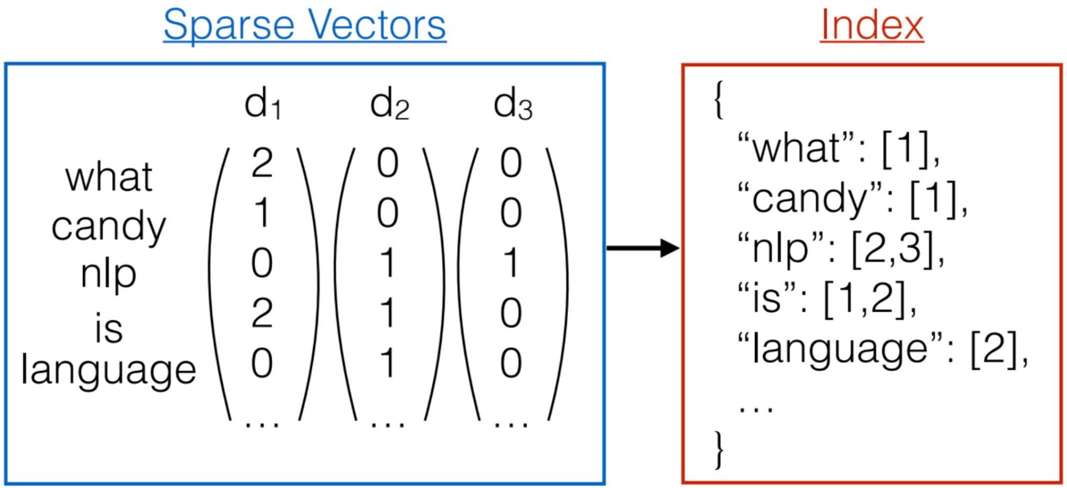
$$\mathrm{BM\text{-}25}(t,d) = \mathrm{IDF}(t) \cdot \frac{\mathrm{freq}(t,d) \cdot (k_1 + 1)}{\mathrm{freq}(t,d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{\mathrm{avgdl}}\right)}$$

Term Frequency Saturation

Document length normalization

Inverted Index

- A data structure that allows for efficient sparse lookup of vectors

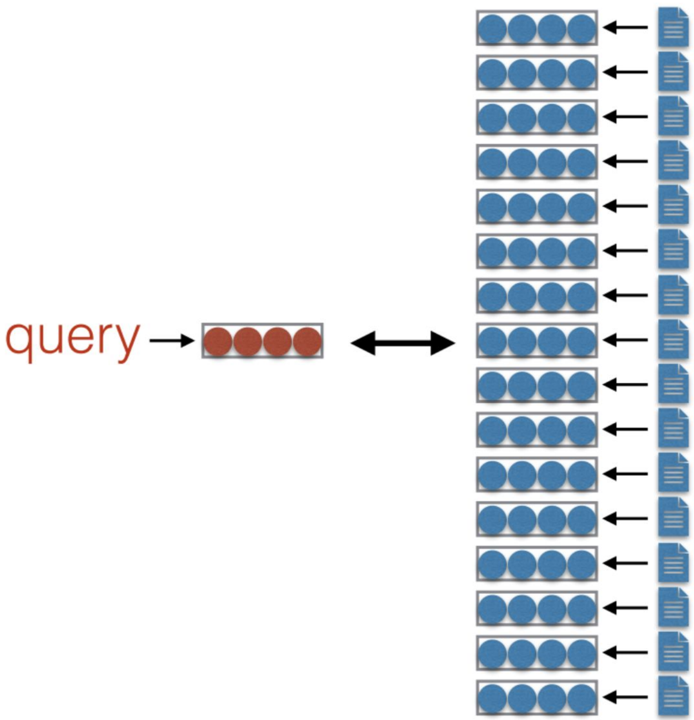


- Example software: Apache Lucene

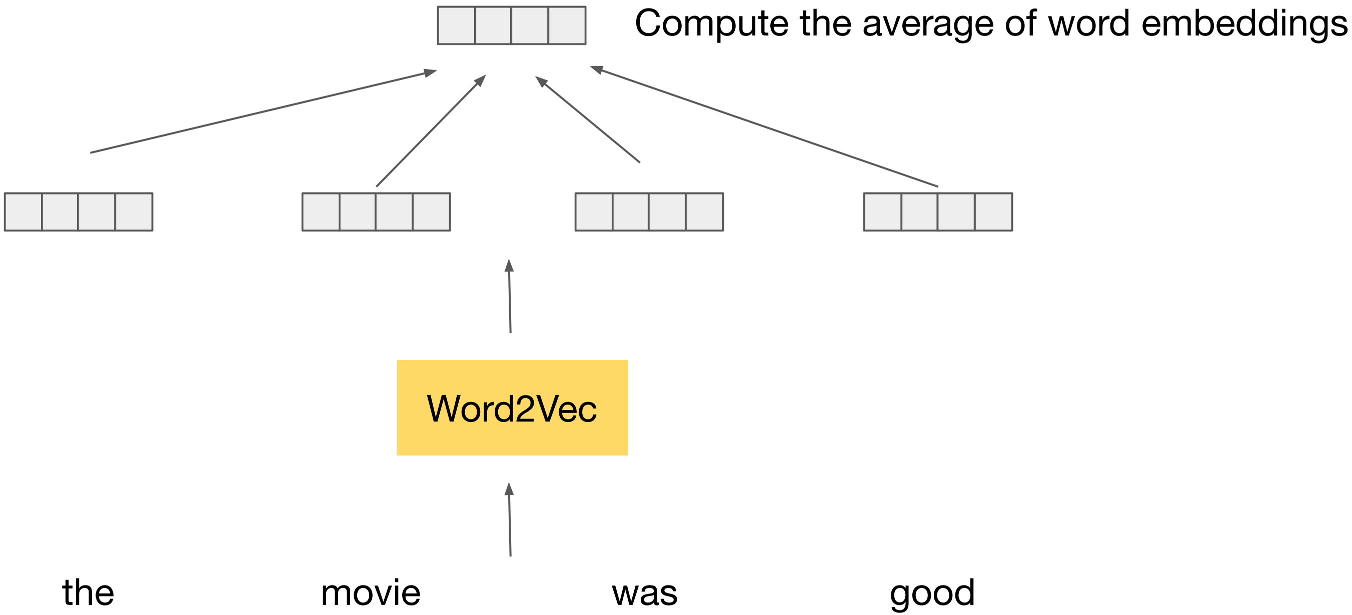
Dense Retrieval

Dense Retrieval

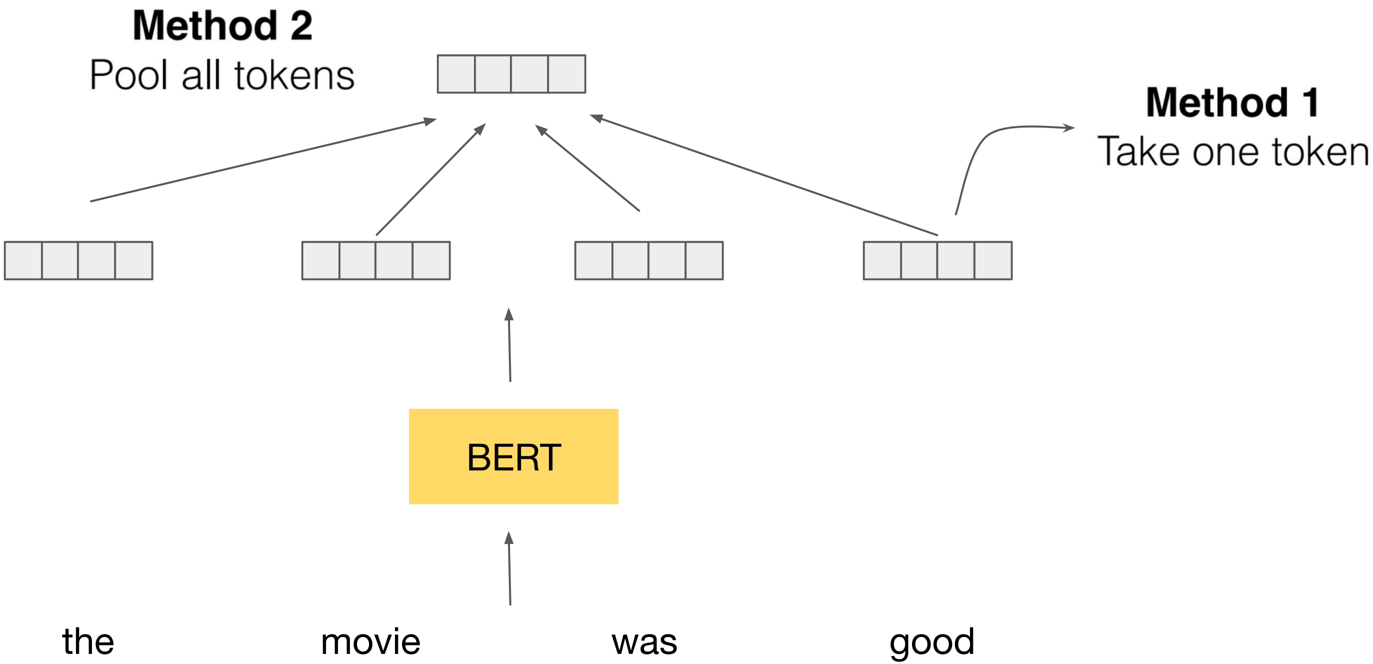
- Encode document/query into dense embeddings and find the nearest neighbors
- Can use:
  - Out-of-the-box embeddings
  - Learned embeddings



Creating Query/Document Embeddings (1)

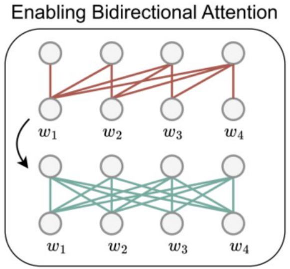


Creating Query/Document Embeddings (2)

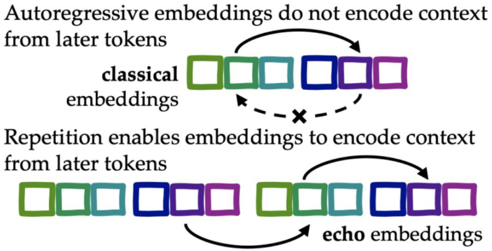


Bidirectional vs. Unidirectional Attention

- **Bidirectional Attention**: Use a masked language model like BERT, RoBERTa, ModernBERT etc. as base
- **LLM2Vec** (BehnamGhader 2024): Transform any decoder-only LLM into a strong text encoder. Remove mask and use/train



- **Echo Embeddings** (Springer et al 2024): Repeat the string multiple times in a unidirectional model

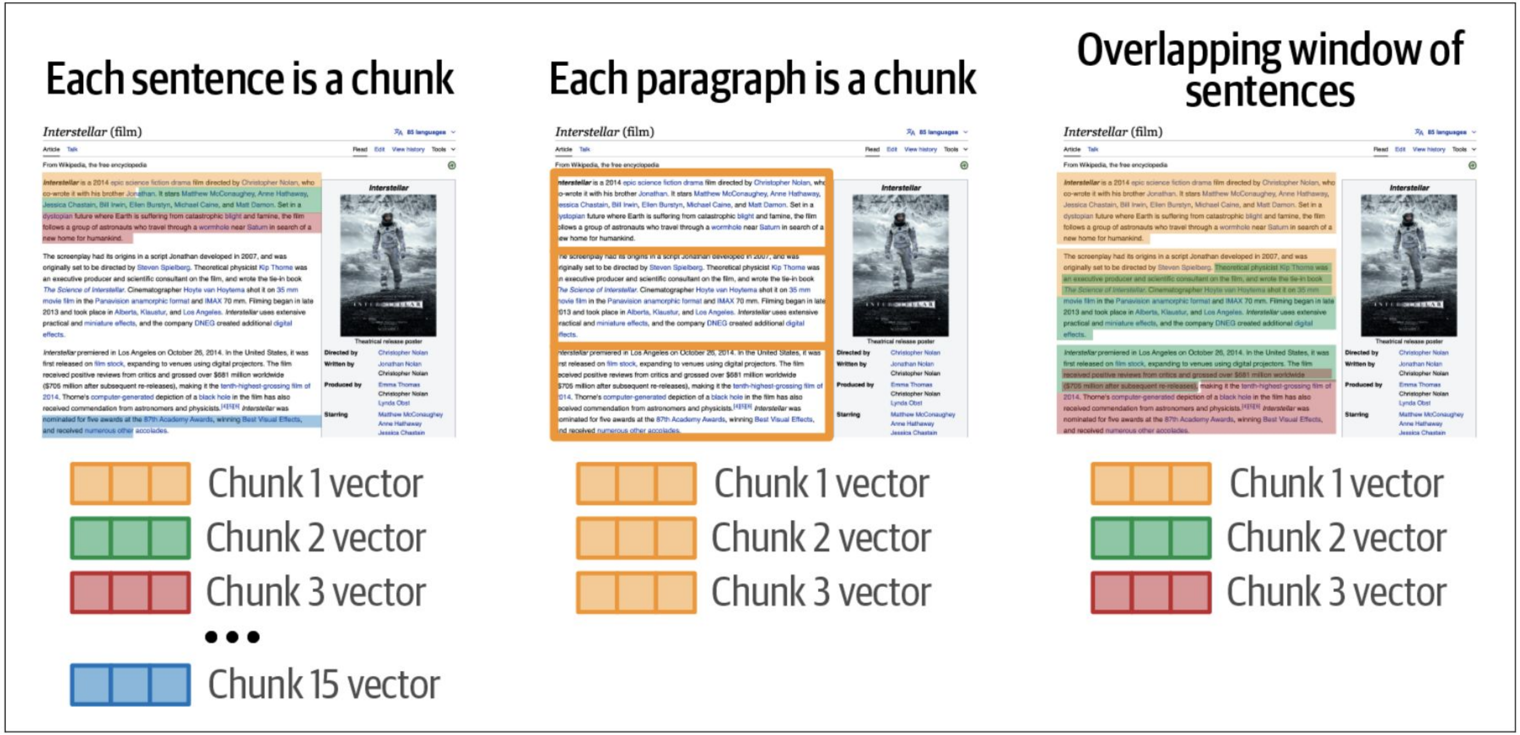


Caveats of Dense Retrieval
+ Solutions

Caveats of Dense Retrieval

- Returns results even if no exact answer exists
  - Solution: Set a relevance threshold to filter out low-confidence results
- Answers may span multiple sentences
  - Solution: Optimize **text chunking** strategies
- Struggles with finding exact phrases
  - Solution: Use **hybrid search** (semantic + keyword search)
- Models trained on one domain (e.g., Wikipedia) may not perform well in another (e.g., legal texts)
  - Solution: **Train on domain-specific data**

Chunking long texts



Img: Hands on Large Language Models

Hybrid Search

- Keyword search (sparse retrieval, e.g., TF-IDF, BF25)
  - Matches exact words; precise but misses synonyms.
- Semantic search (dense retrieval)
  - Understands meaning; flexible but may retrieve loosely related results.
- Hybrid Search
  - Combines both for better accuracy and coverage.

Reciprocal Rank Fusion (RRF)

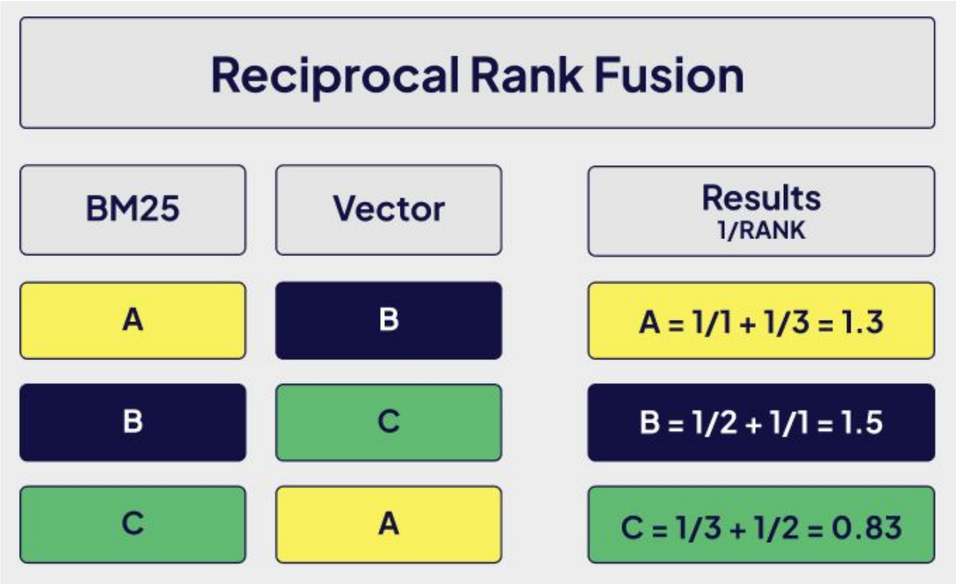
- Reciprocal Rank Fusion (RRF) is an ensemble ranking method used to combine multiple ranked lists of search results.
- Given multiple ranked lists (e.g., from different retrieval models), RRF assigns each document a score based on its rank in each list using the following formula:

$$\text{RRF Score}(d) = \sum_{i=1}^{N} \frac{1}{k + \text{rank}_i(d)}$$

where:

- N = number of ranking lists
- $\text{rank}_i(d)$ = rank of document d in the i-th ranked list
- k = a small constant (typically 60) to prevent dominance by top-ranked results

RRF Example



Img: https://weaviate.io/blog/hybrid-search-for-web-developers

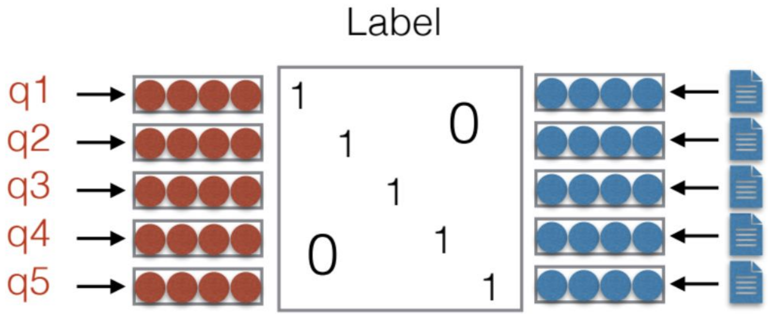
Learning Retrieval-oriented Embeddings

- Basic idea: move the positive documents closer, negative documents farther away
- Select positive and negative documents, train using a contrastive loss (e.g. triplet loss)

$$L = \max(d(A, P) - d(A, N) + m, 0)$$

How to get negative examples? - In-batch negatives

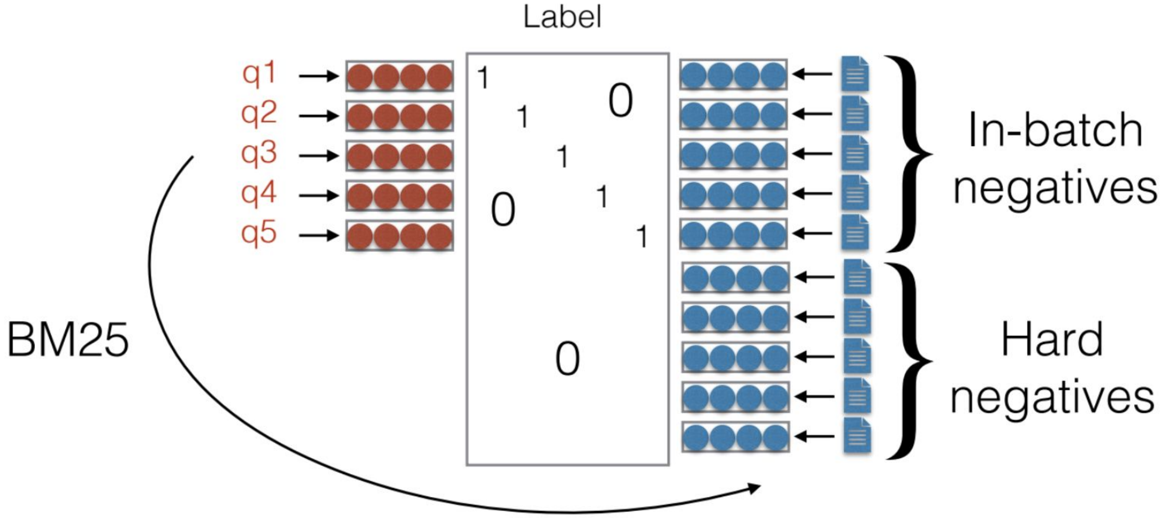
- Create a batch of queries and associated documents
- Treat all other documents in the batch as negative examples



- Problem: not enough hard examples

Negative examples - Hard negative mining

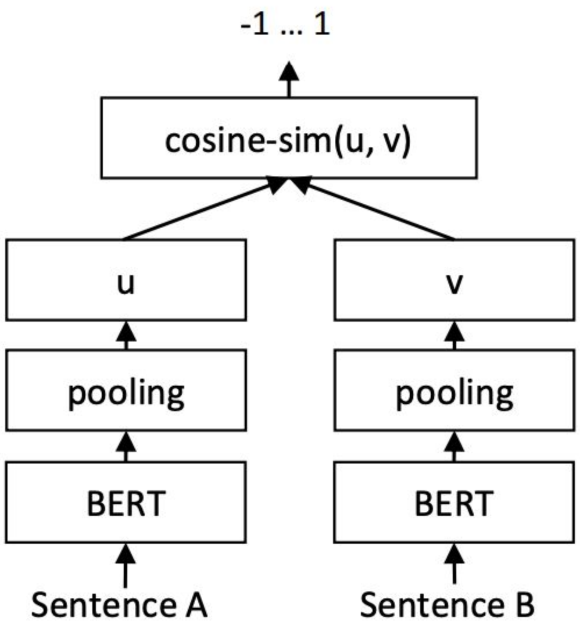
- Use a weaker retriever (e.g. BM25) to more examples and treat them as negatives



- Problem: hard “negatives” might actually be positive

(reminder) Sentence-BERT (SBERT) (Reimers and Gurevych 2019)

- A modification of BERT designed for sentence embeddings
- Optimized for semantic similarity tasks via siamese or triplet networks, enabling fast & accurate sentence comparison
- Widely used in semantic search, clustering, and QA



Reimers and Gurevych 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. https://arxiv.org/abs/1908.10084

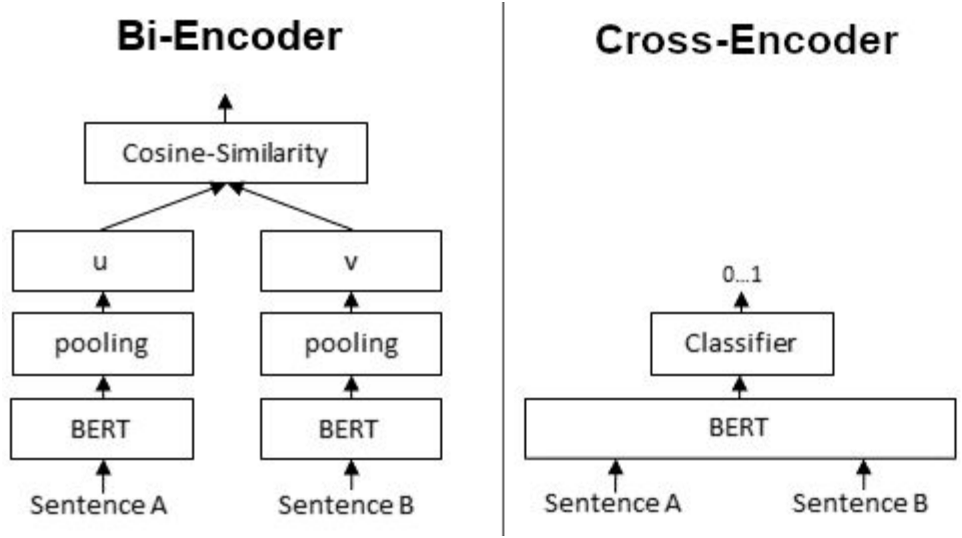
Datasetstructures to train your SentenceTransformers model		
Datasetstructure	**Example datasets(repo id in Hugging Face Hub)**	**Loss functions(imported from sentence_transformers)**
Pair of sentences and a label indicating how similar they are	snli	ContrastiveLoss; SoftmaxLoss; CosineSimilarityLoss
Pair of positive (similar) sentences without a label	embedding-data/flickr30k_captions_quintets; embedding-data/coco_captions_quintets	MultipleNegativesRankingLoss; MegaBatchMarginLoss
Single sentence with an integer label	trec; yahoo_answers_topics	BatchHardTripletLoss; BatchAllTripletLoss; BatchHardSoftMarginTripletLoss; BatchSemiHardTripletLoss
Triplet (anchor, positive, negative) sentences	sembedding-data/QQP_triplets	TripletLoss

Img: https://huggingface.co/blog/how-to-train-sentence-transformers

Reranking

Bi-Encoder vs Cross-Encoder

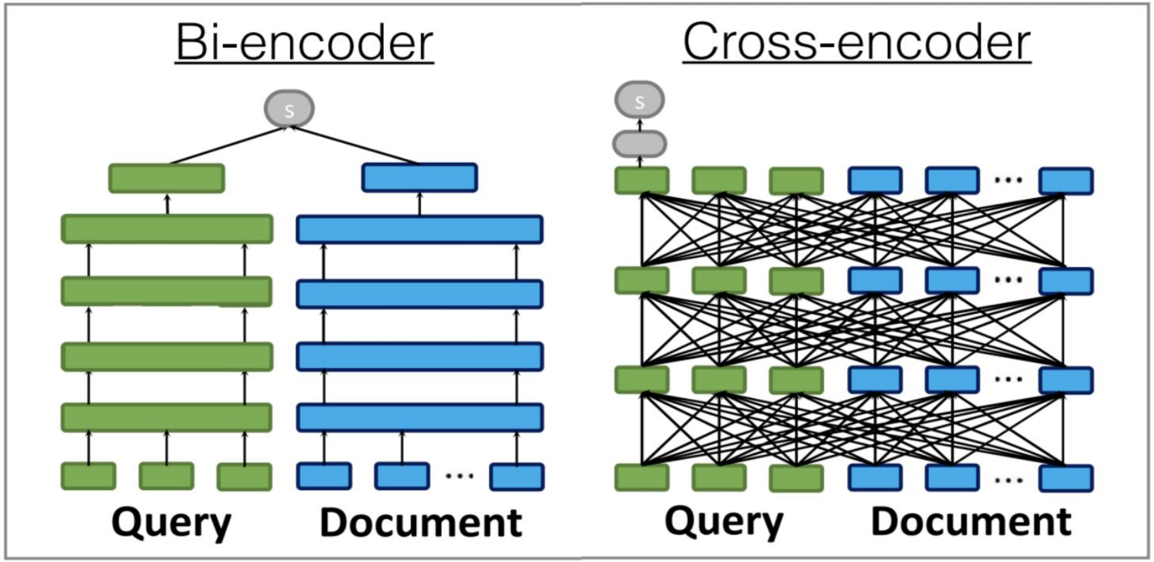
- **Bi-Encoder**: Encodes sentences independently into embeddings, which are then compared using cosine similarity.
- **Cross-Encoder**: Jointly encode both queries and documents using neural model (Nogueira et al. 2019), directly outputting a similarity score between 0 and 1.



Img: https://sbert.net/examples/applications/cross-encoder/README.html

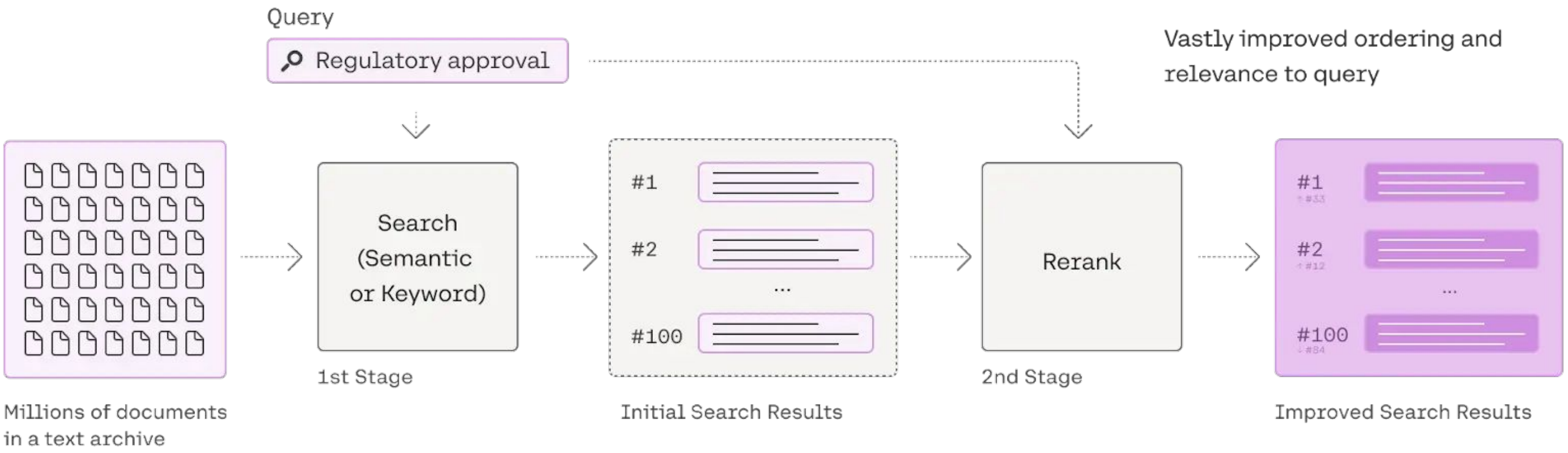
Bi-Encoder vs Cross-Encoder

Given 100 queries and 1,000 documents, how much computation is needed to identify similar sentences?



Img: https://arxiv.org/abs/2004.12832

Reranking



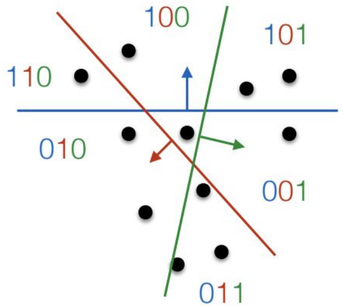
Img: https://cohere.com/blog/rerank

Vector Database

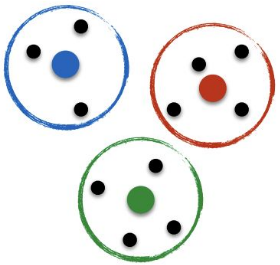
Approximate Nearest Neighbor Search

- Methods to retrieve embeddings in sub-linear time

Locality sensitive hashing: make partitions in continuous space, use like inverted index



Graph-based search: create "hubs" and search from there

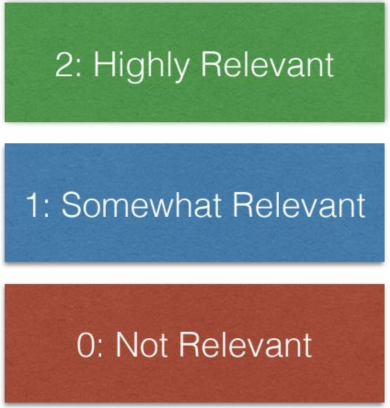


- Software: FAISS, ChromaDB

Evaluating Retrieval

Ranking Metrics

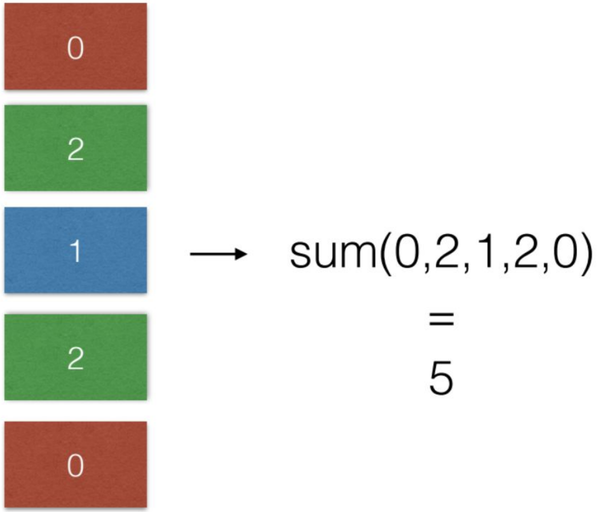
- What is good results:
  - Relevant information is included in the top of the ranking list
- Generally start with gold-standard relevance judgements and try to match them
- e.g.



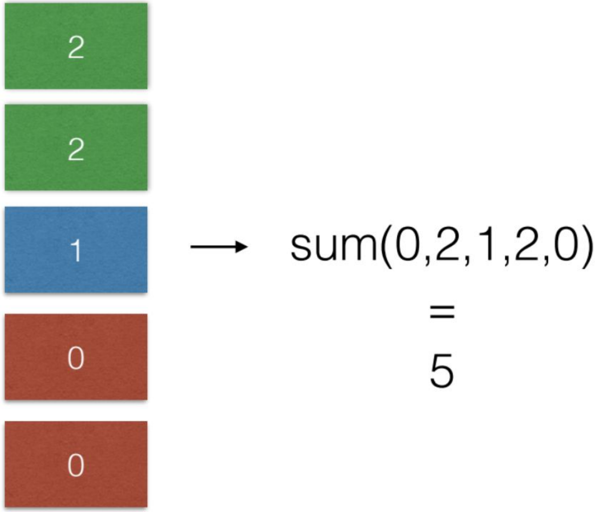
Cumulative Gain (Hegde 22)

- Sum of relevance score @ N values retrieved

CG@5 for ranking

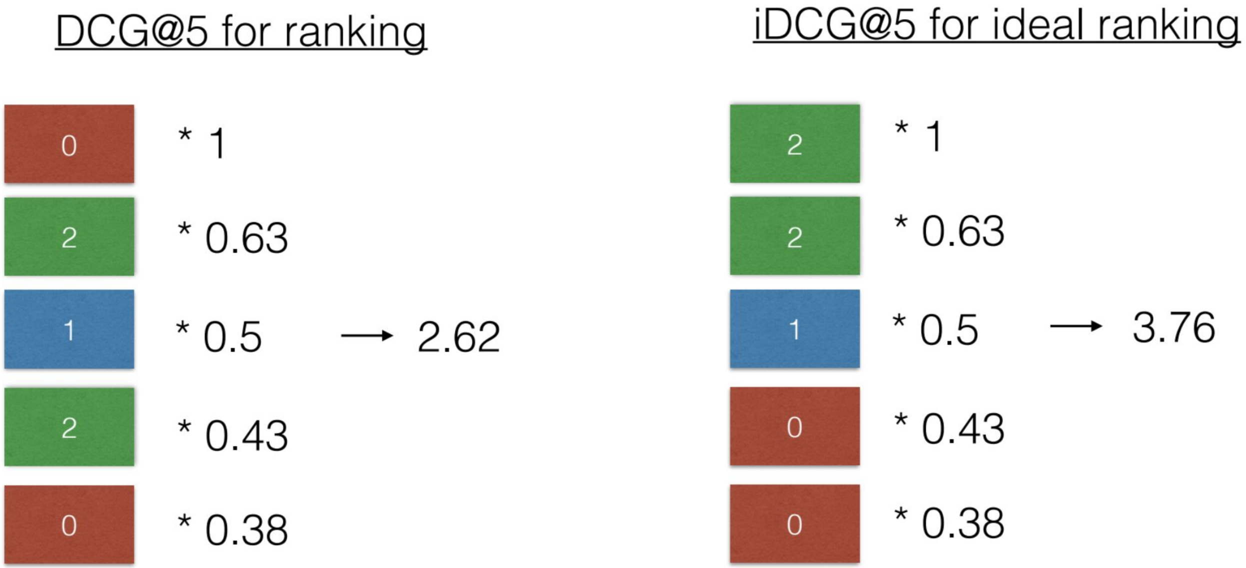


CG@5 for ideal ranking



Discounted Cumulative Gain (Hegde 22)

- Add a discount 1/log2(i+1) for lower ranked values



Normalized Discounted Cumulative Gain

- Makes sure that as you pick up more good docs you get a better score
- nDCG = DCG/iDCG

0	* 1	2	* 1
2	* 0.63	2	* 0.63
1	* 0.5	1	* 0.5
2	* 0.43	0	* 0.43
0	* 0.38	0	* 0.38

If 3rd item was not relevant ('0') →

DCG@3=1.26
iDCG@3=3.76
nDCG@3=0.34

DCG@2	iDCG@2
1.26	3.26

nDCG@2

0.386

DCG@3	iDCG@3
1.76	3.76

nDCG@3

0.468

Other Metrics

- Mean Average Precision: The average precision at which each relevant document is retrieved

no	~~0/1~~	
yes	1/2	
yes	2/3	→ 0.638
yes	3/4	
no	~~3/5~~	

- Recall@N: The proportion of relevant documents retrieved within the top-N results, relative to the total number of relevant documents available.

no
yes
yes
yes
no

If there are 10 relevant documents for a query,

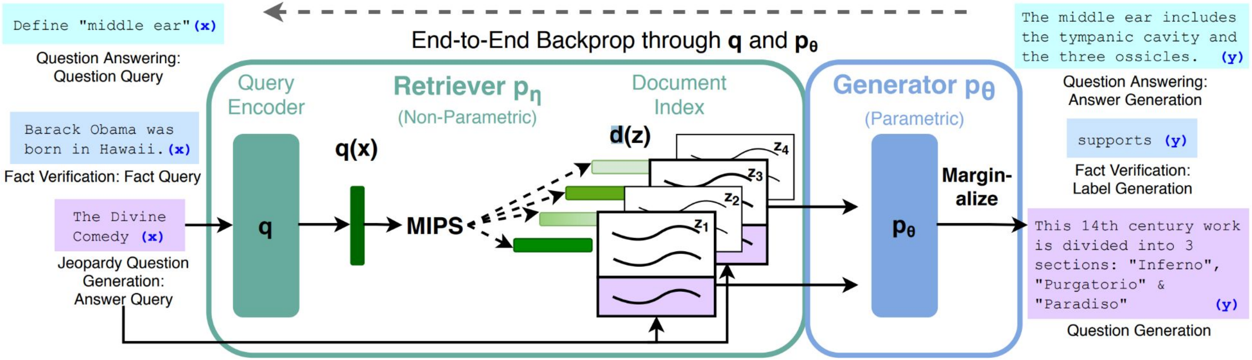
R@1 = 0
R@2 = 1/10 = 0.1
R@3 = 2/10 = 0.2
R@4 = 3/10 = 0.3
R@5 = 3/19 = 0.3

Retriever-Generator Models

Retriever + Generator End-to-end Training ("RAG")

(Lewis et al. 2020)

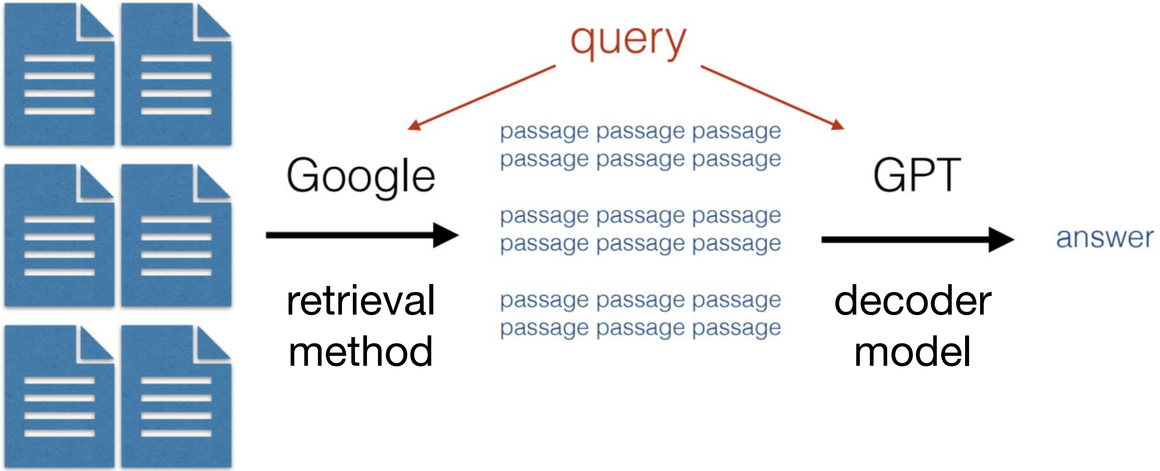
- Train the retriever and generator to improve accuracy
- Generator: Maximize the generation likelihood given retrieved document(s)
- Retriever: Optimize mixture weights over documents to maximize overall likelihood (training applies only to the query encoder)



Lewis et al. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. https://arxiv.org/abs/2005.11401

Simple: Just Chain Retrieval+Generator

- Use an out-of-the-box retriever and out-of-the-box generator



- Passages are concatenated to the context

Any Questions?