Building a Semantic Search Engine

About Us

Vishal Rathi

Nidhin Pattaiyil

Walmart: ML Engineers on the Search team

Agenda

1) Core concepts that are common for any search retrieval systems (15 min)
2) Hands On: Environment Setup (10 min)
3) Non deep learning based retrieval system (30 min)
4) Hands On (30 min)
5) Overview of Embedding based retrieval system (40 min)
6) Speeding Neural IR with Approximate Nearest Neighbors (ANN) (15 min)
7) Hands On (30 min)
8) Conclusion (5 min)

By the end of this workshop, you will:

- Understand evaluation metrics for a search engine metrics to evaluate a search engine
- Build a full text search engine with ElasticSearch
- Build an embedding search engine with Milvus
- Speed up vector retrieval with ANN
- Understand some E2E design considerations

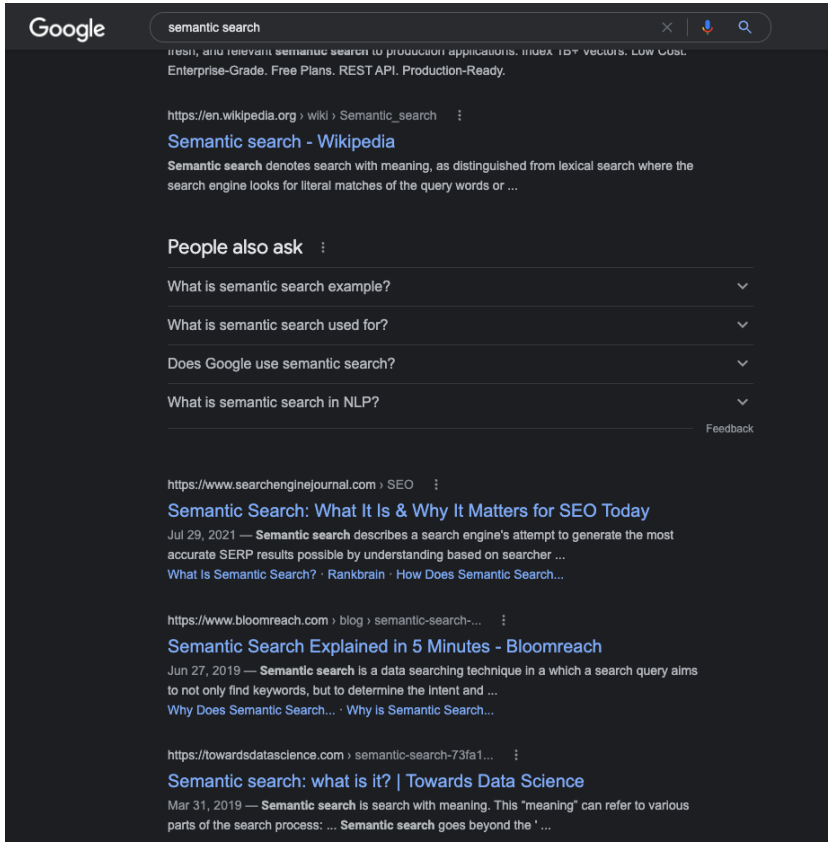
Core Search concepts

Outline

- What is information retrieval
- Evaluation metrics
- Past and current approaches

What is Information retrieval (IR)

- Finding material from a corpus (collection of documents) that satisfies the information need.
- Information need is expressed as text or voice
- Corpus can be collection of documents, audio recordings, videos..etc
- Our focus is queries and documents in text



Evaluation Setup for IR System

In order to measure effectiveness of an ad-hoc information retrieval system we need 3 following things:

1. A document collection

2. A test suite of information needs, expressible as queries

3. A set of relevance judgments that express relevancy of the documents to the information need either binary or in a score range (1-5)

Evaluation Metrics

Order unaware metrics:

- Precision @k
- Recall@k

Order Aware Metrics:

- Mean Reciprocal Rank (MRR)
- Mean Average Precision (MAP)

Graded Relevance:

- Cumulative Gain (CG)
- Discounted Cumulative Gain (NDCG)

Business Metrics

- Click Through Rate (CTR)
- Add to Cart Rate (ATC)

Metrics: Precision @ K

Precision is fraction of the relevant retrieved documents to predefined K value.

Precision at K = (# Relevant docs at top K) / K

Model 1: D1 D2 D3 D4 D5 P@5 = 2/5

Model 2: D1 D2 D3 D4 D5 P@5 = 1/5

Metrics: Recall @ K

Recall at K is fraction of relevant documents in top K to the total number relevant documents in the corpus.

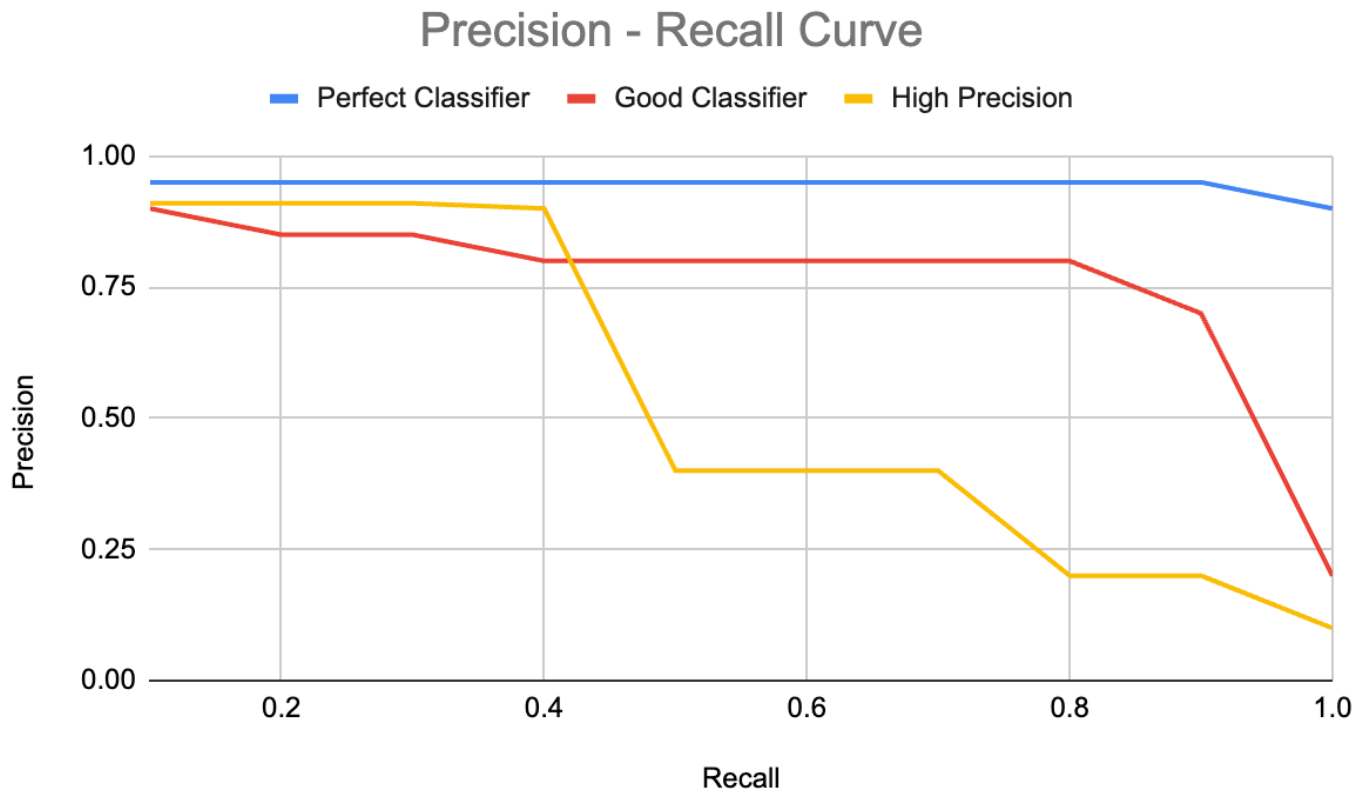
Recall at K = (# Relevant docs at top K) / (# Relevant docs in Corpus)

Corpus: 100 Relevant Docs

Model 1: D1 D2 D3 D4 D5 R@5 = 2/100

Model 2: D1 D2 D3 D4 D5 R@5 = 1/100

Metrics: Precision - Recall Curve



Metrics: Mean Reciprocal Rank (MRR)

MRR rewards systems that rank the relevant documents in the highest positions (question answering, typeahead)

MRR = $\sum$(1/Rank of Highest Relevant Document) / |Q|

Q1: D1 D2 D3 D4 D5

Q2: D1 D2 D3 D4 D5

Q3: D1 D2 D3 D4 D5

MRR = (1/2 + 1/1 + 1/3)/3

Metrics: Mean Average Precision (MAP)

Recall precision@5 for these two models are same!

Model 1: D1 D2 D3 D4 D5 P@5 = 3/5

Model 2: D1 D2 D3 D4 D5 P@5 = 3/5

We need to do better than this!

What if we calculate precision at when we get a relevant document?

Model@1: D1 P@1 = 0

Model@2: D1 D2 P@2 = 1/2

Model@3: D1 D2 D3 P@3 = 2/3

Model@4: D1 D2 D3 D4 P@4 = 0

Model@5: D1 D2 D3 D4 D5 P@5 = 3/5

AP= (1/2 + 2/3 + 3/5)/3 = 0.59

Model@1: D1 P@1 = 1/1

Model@2: D1 D2 P@2 = 2/2

Model@3: D1 D2 D3 P@3 = 3/3

Model@4: D1 D2 D3 D4 P@4 = 0

Model@5: D1 D2 D3 D4 D5 P@5 = 0

AP= (1/1 + 2/2 + 3/3)/3 = 1.0

$$Average\ Precision = \frac{\sum_r P@r}{R}$$

r is the rank of each relevant document and R is the total number of relevant documents.
q is number of queries.

$$MAP = \frac{1}{q}\sum_q AP_q$$

Metrics: Graded Relevance

Graded relevance judgements instead of **binary** relevance judgements!

Relevancy score changes between 0 to 5

0 is non relevant and 1 to 5 are relevant depending on relevancy level

Q : D1 D2 D3 D4 D5

R : 0 2 0 3 1

Spam documents are given score of -2 in some evaluations. This will have some consequences to be taken care of by normalization but it was found useful.
See **The Impact of Negative Relevance Judgments on NDCG** by Gienabb, Frobe, Hagen, Potthast for more info.

Metrics: Normalized Discounted Cumulative Gain

Metrics: Normalized Discounted Cumulative Gain

$$Discounted\ Cumulative\ Gain = \sum_{1}^{p} \frac{relevance(i)}{log_2(i+1)}$$

$$DCG = \frac{3}{log_2(2)} + \frac{1}{log_2(3)} + \frac{0}{log_2(4)} + \frac{2}{log_2(5)} = 4.49$$

$$(Ideal)\ DCG = \frac{3}{log_2(2)} + \frac{2}{log_2(3)} + \frac{1}{log_2(4)} + \frac{0}{log_2(5)} = 5.88$$

Metrics: Normalized Discounted Cumulative Gain

$$Discounted\ Cumulative\ Gain = \sum_{1}^{p} \frac{relevance(i)}{log_2(i+1)}$$

$$DCG = \frac{3}{log_2(2)} + \frac{1}{log_2(3)} + \frac{0}{log_2(4)} + \frac{2}{log_2(5)} = 4.49$$

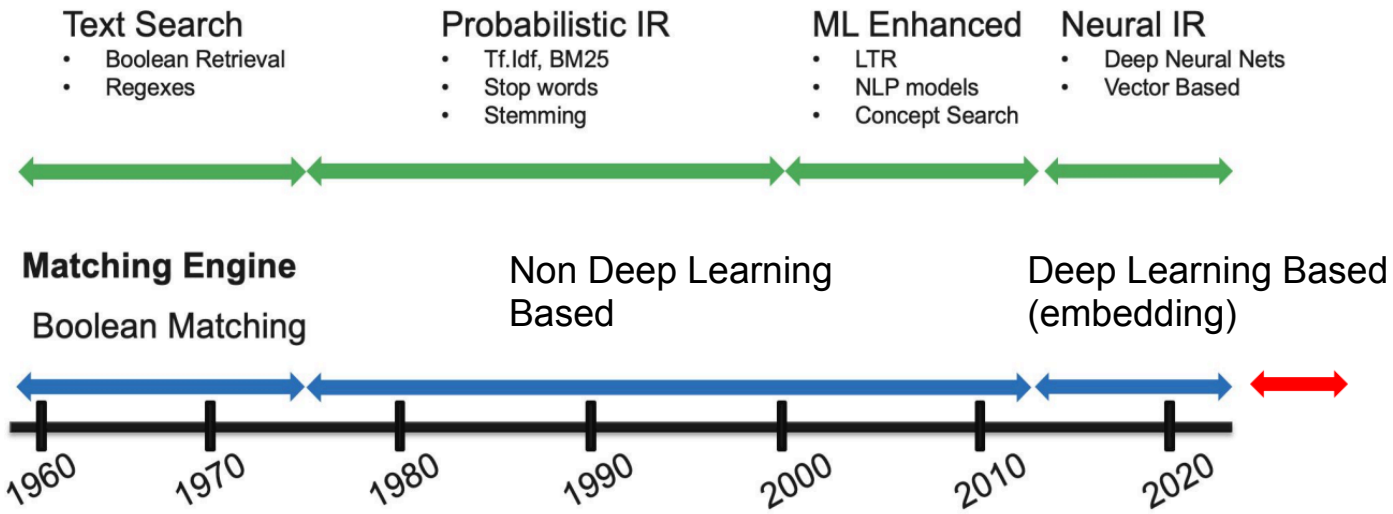
$$(Ideal)\ DCG = \frac{3}{log_2(2)} + \frac{2}{log_2(3)} + \frac{1}{log_2(4)} + \frac{0}{log_2(5)} = 5.88$$

$$Normalized\ Discounted\ Cumulative\ Gain = \frac{DCG}{Ideal\ DCG} = \frac{4.49}{5.88}$$

Metrics: (Business)

- Click through Rate
- Add to Cart Rate
- Session Length / Time on Site
- Introducing new items/ sellers (Coverage)

History of Information Retrieval



Hughes, Simon. "Semantic Product Search – Vector Search for E-Commerce." Conference Presentation at Haystack 2021, https://haystackconf.com/files/slides/haystack2021/Hughes-Haystack_2021_Semantic_Product_Search.pdf, September 29, 2021.

Hands On: Environment Setup

https://hub.np.training

https://bit.ly/search-workshop-2022

- Code the metrics
- Explore the data used in the workshop

Non deep learning based retrieval system

Outline

- Token based search
- Why Inverted Index
- Searching with Inverted Index
- Overview of tf-idf and BM-25
- ElasticSearch
- Hands-on lab

Token Based Search

Document & Corpus Representation

- Tokenize and normalize the docs and represent docs as **“bag of words”**
- Example:
  - Document: “*The PageRank paper was rejected from SIGIR.*”
  - Bag of words: [the, pagerank, paper, was, rejected, from, sigir

Search

- Query - “pagerank paper”

Sample Index

Doc	from	…	page rank	…	paper	..	rejected	..	was
doc 1	1	0	0	0	0	0	0	0	1
doc 2	1	0	1	0	1	0	1	0	1
…	0	1	0	0	1	0	0	0	0
doc_n	0	0	0	1	0	0	1	0	0

Space Complexity: O(|D|*|V|) where |D| number of documents and |V| is vocabulary size
Search Complexity: O(|Q|*|V|*|D|)

https://www.cs.virginia.edu/~hw5x/Course/IR2020-Spring-Old/docs/PDFs/Inverted%20Index.pdf

Inverted Index

- Term oriented index instead of document oriented index
- Tokenize, stop and stem
- Create index

ID	Term	Doc-List
…	…	…
109273	pagerank	5,1005,6492
109274	paper	5,1005
…	…	
120001	reject	1005,7921
120002	sigir	10,328

Time complexity
- $O(|q| * |L|)$
- |q| is the length of the query, |L| is the average length of the Document List

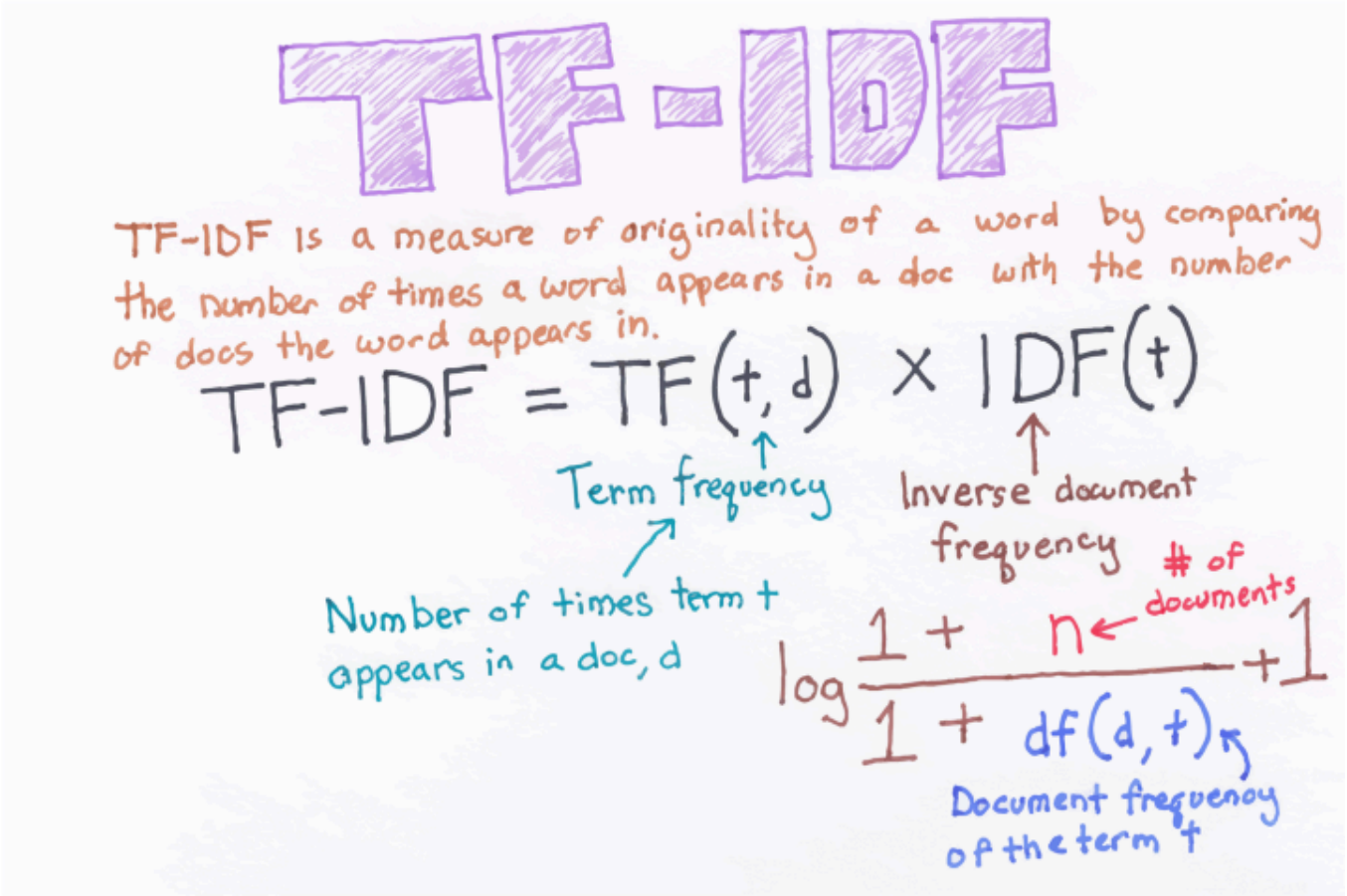
Space complexity
- $O(L * V)$
- V is vocabulary size
- By Zipf's law, $L \ll D$

https://www.linkedin.com/pulse/what-inverted-index-saral-saxena/?trk=related_artice_What%20is%20inverted%20index%3F_article-card_title

Early Models - Boolean Retrieval

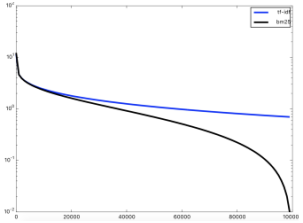
- Query terms are connected with logical operators (AND, OR, NOT)
- Scan through the index and retrieve documents that satisfy the condition
- EG: *(red **OR** blue) **AND** dress*
- Retrieves a set of documents without ranking.

ID	Term	Doc-List
…	…	…
9000	blue	1005
…	…	…
109273	dress	5,1005,6492, 7921
109274	dry	5,1005
…	…	…
120001	red	1005,7921
120002	sigir	10,328



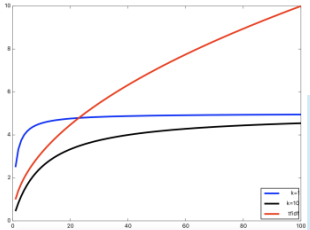
Credit : Chris Albon @ https://machinelearningflashcards.com/

BM25



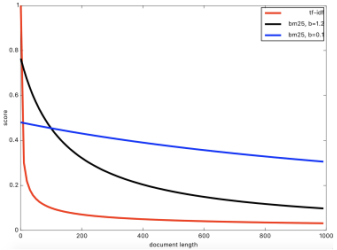
idf - how popular is the term in the corpus?

$$\text{bm25}(d) = \sum_{t \in q, f_{t,d} > 0} \log\left(1 + \frac{N - df_t + 0.5}{df_t + 0.5}\right) \cdot \frac{f_{t,d}}{f_{t,d} + k \cdot (1 - b + b\frac{\text{l}(d)}{avgdl})}$$



saturation curve - limit influence of tf on the score

length weighing - tweak influence of document length



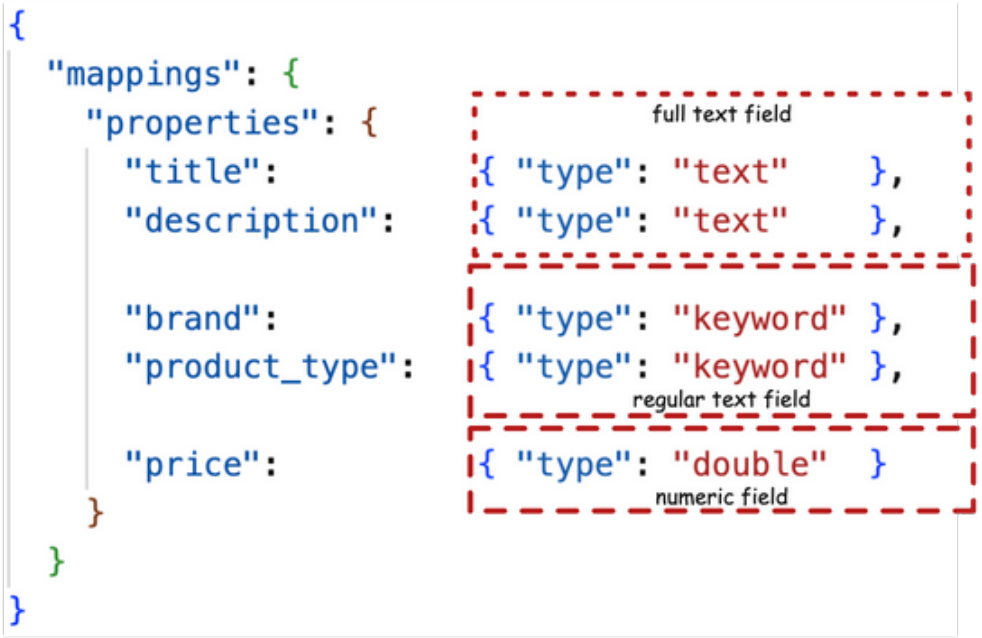
Weber, Britta. "Improved Text Scoring with BM25." Conference Presentation at ElasticCon 2016, https://speakerdeck.com/elastic/improved-text-scoring-with-bm25, February 11, 2016.

ElasticSearch

- Open source search engine based on Lucene library
- Supports BM25 and other similarities (link)
- Supports boosting , filtering , phrase match, autocomplete
- Distributed : index shards, replicas

elasticsearch

Elastic Search Schema



Possible schema for an e-commerce item

Elastic Search Query

Query: Nike shoes under 100$

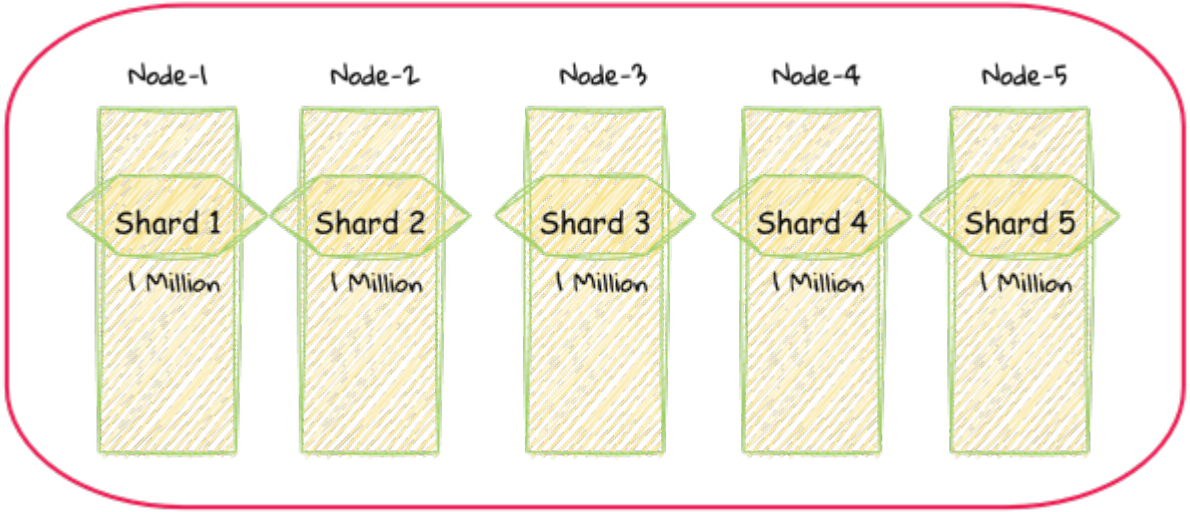
```
{
  "query": {

    "multi_match": {
        "query": "Nike shoe under 100$",
        "fields": ["title^2", "Description^1"]
    }

    ,"bool": {
      "filter": [
        { "term": { "brand": "nike" }}
      ]
    }
    ,"filtered": {
        "filter": {
            "range":  {
                "price" : {  "lte": 100 }
            }
        }
    }
}
```

Search for query tokens in title and description
weight matches found in title double

filter items who has brand nike

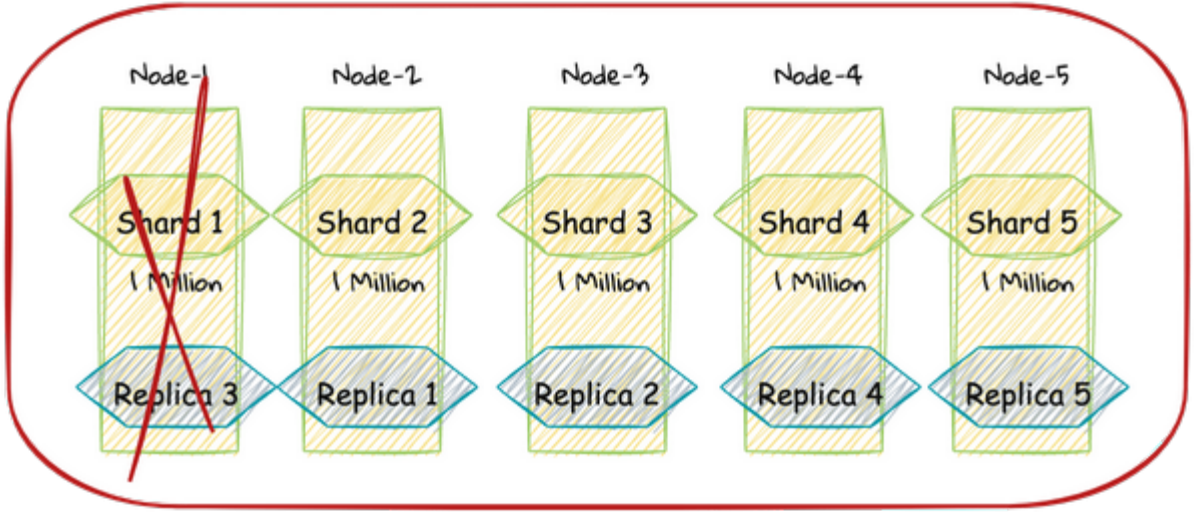
filter items with price <= 100

Index with multiple shards



- Index can be composed of shards
- Reading / Writing can be faster

Index with replicas



Replicating a shard improves resilience and read throughput

Hands On

- Indexing some documents with PyElastic
- Reviewing Retrieval Results from BM25

Overview of Embedding based retrieval system

Outline

- What are embeddings
- Sparse Embedding
- Gaps with Sparse Representation
- Dense embeddings
- Different ways to generate Dense embeddings
- How to use embeddings for retrieval
- How to create training data

Embeddings

Representing a word / concept / document as a vector

Sparse Embeddings

Word Representation (one hot encoding)

fox: [0 , 1, 0, 0, …. , 0]

dimension = size of the vocabulary

Document Representation

Quick brown fox: [0 , 1, 0, 1, …. , 1]

Issues with Sparse Representation

- Lexical GAP: Covid vs Coronavirus vs Omicron variant
- Ambiguity: bank (institution) vs bank (geography)
- Position matters: “river bank” vs “bank river”
- Lack of Contextualized embeddings

She will ***park*** the car so we can walk in the ***park***.

Dense Embeddings

Word Representation

car: [0.2 , 0.3, 0.7]

automobile: [0.2 , 0.3, 0.7]

Similar concepts have similar embeddings

Regardless of content length, similar items should have similar embeddings

Size of embedding is independent of #tokens

Review Representation

Review 1: 🍨 was great

Review 2: Chocolate ice cream was the best ..

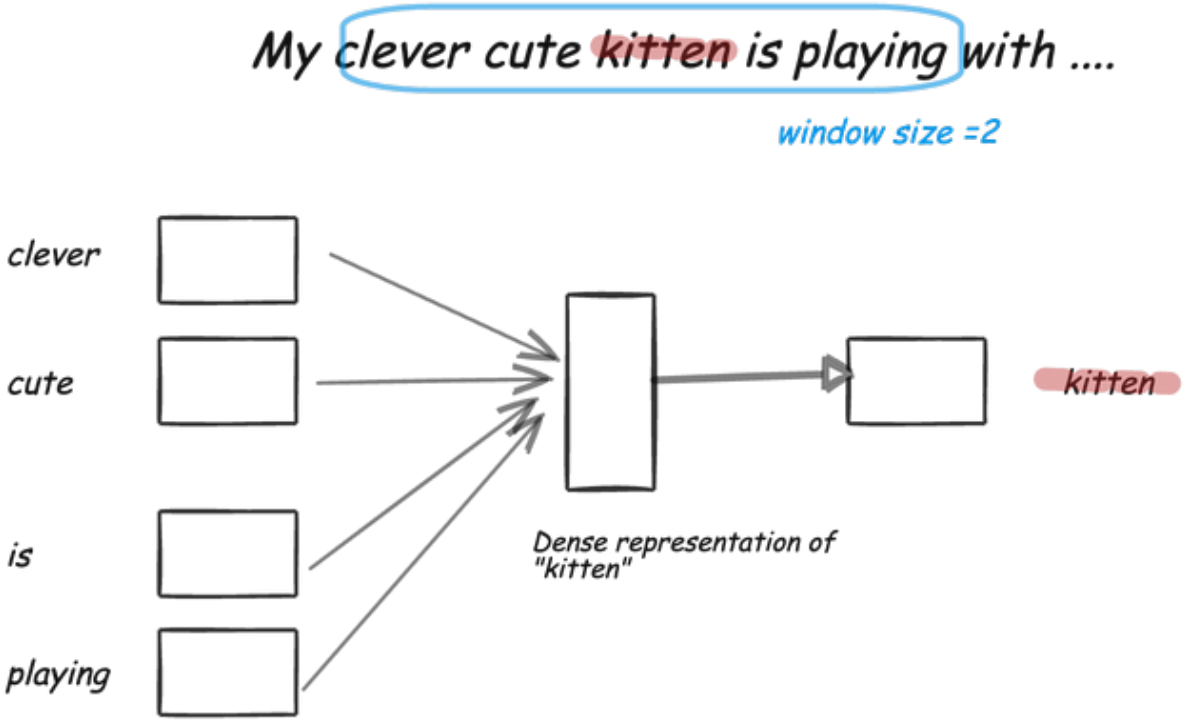
[0.5 , 0.1, … , 0.6]

[0.5 , 0.1, … , 0.5]

Generate Dense Vectors

- The ‘2vec’ methods
- Sentence Transformers
- Multimodal Embeddings (clip)

Word2Vec

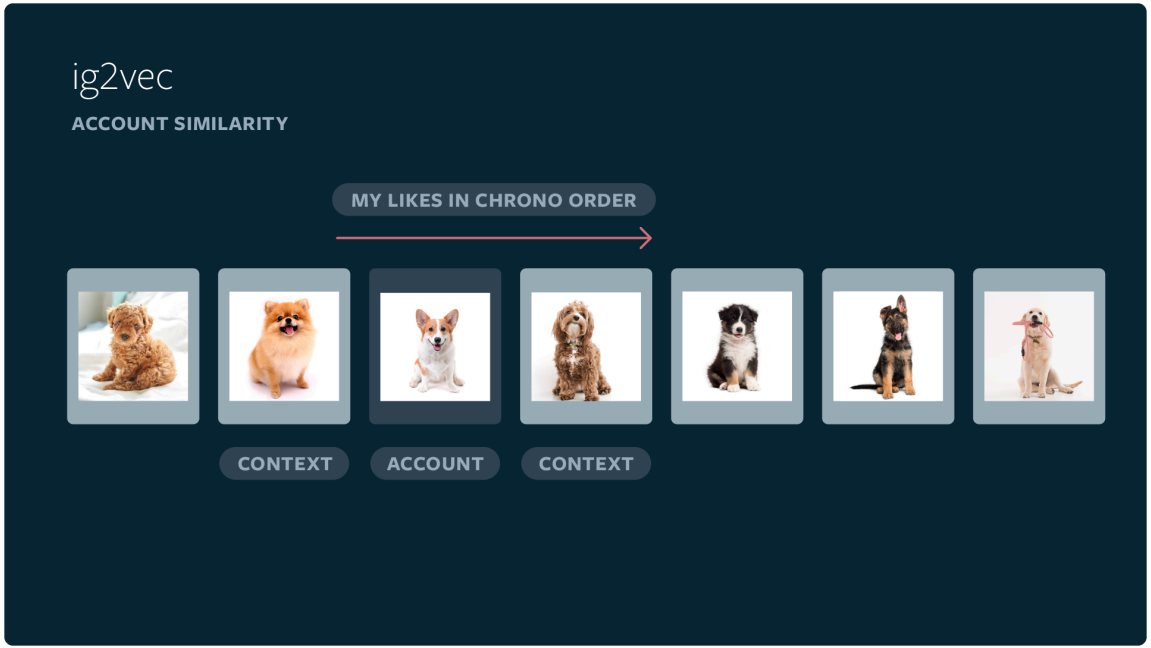


- Published in 2013
- Represent each word as a dense vector

- Uses a neural network model to capture linguistic concept of words

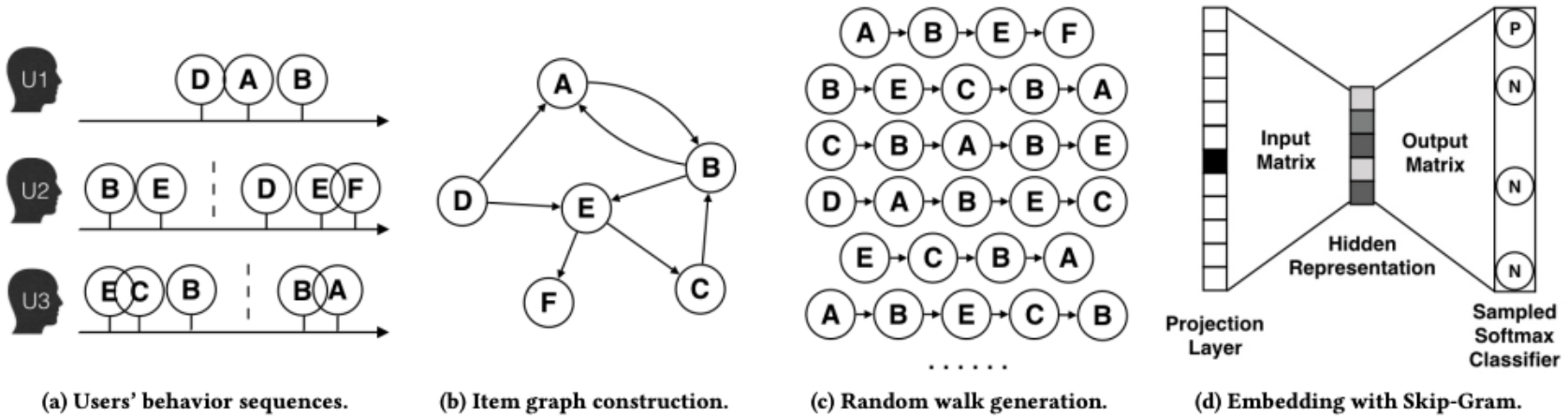
W2Vec Continuous Bag of Words

Instagram: ig2Vec



Medvedev, Ivan, Haotian Wu, and Taylor Gordon. “Powered by AI: Instagram’s Explore Recommender System.” Powered by AI: Instagram’s Explore recommender system, November 2019. https://ai.facebook.com/blog/powered-by-ai-instagrams-explore-recommender-system/

Taobao: Graph2Vec



(a) Users' behavior sequences. (b) Item graph construction. (c) Random walk generation. (d) Embedding with Skip-Gram.

Figure 2: Overview of graph embedding in Taobao: (a) Users' behavior sequences: One session for user u1, two sessions for user u2 and u3; these sequences are used to construct the item graph; (b) The weighted directed item graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; (c) The sequences generated by random walk in the item graph; (d) Embedding with Skip-Gram.

Wang, Jizhe, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, και Dik Lun Lee. 'Billion-scale Commodity Embedding for E-commerce Recommendation in Alibaba'. arXiv, 2018. https://doi.org/10.48550/ARXIV.1803.02349.

Evaluating Word2Vec

- Dense Representation ✓
- Train Domain specific embedding ✓
- Context Dependant embedding ✗
- Considers position of words ✗
- Support for out of vocab words ✗
- General language model ✗

Rare and Out of Vocabulary Tokens (OOV)

Source of OOV tokens:
- tokens not training corpus
- typos
- very rare queries

tokens
Word2vec: 1 million

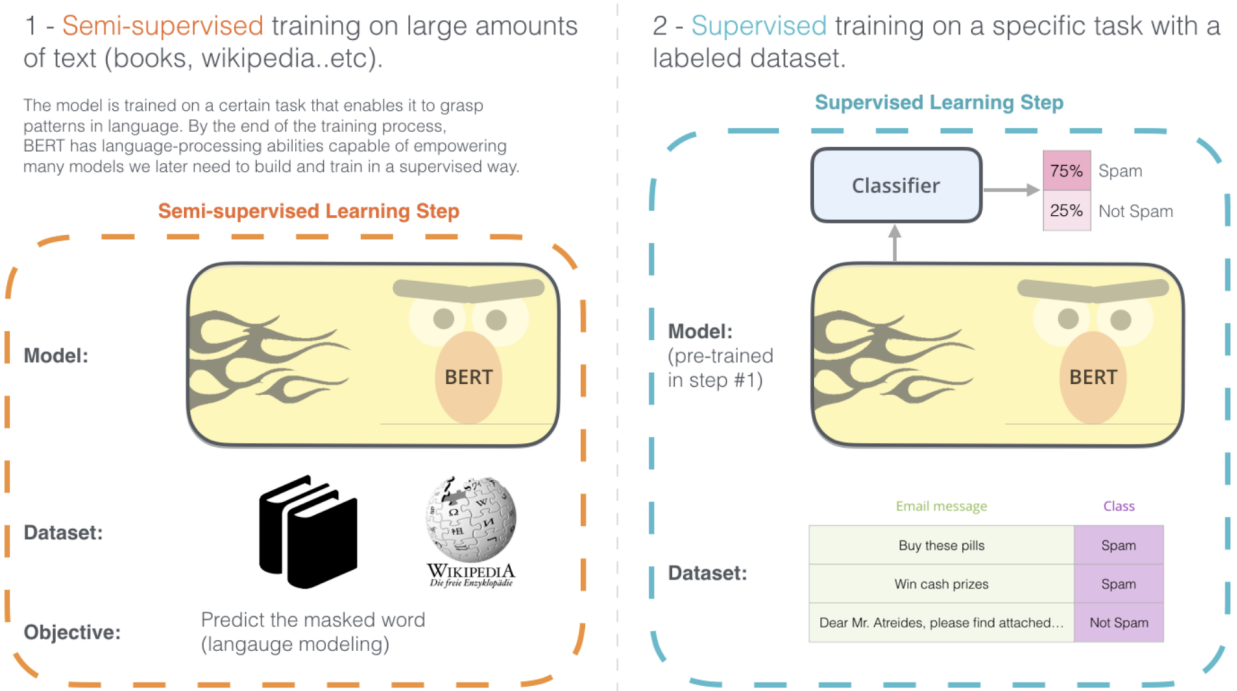
WordPiece: 10k

Solution:

Rule based:
- stemming / lemmatization
- Character n-grams

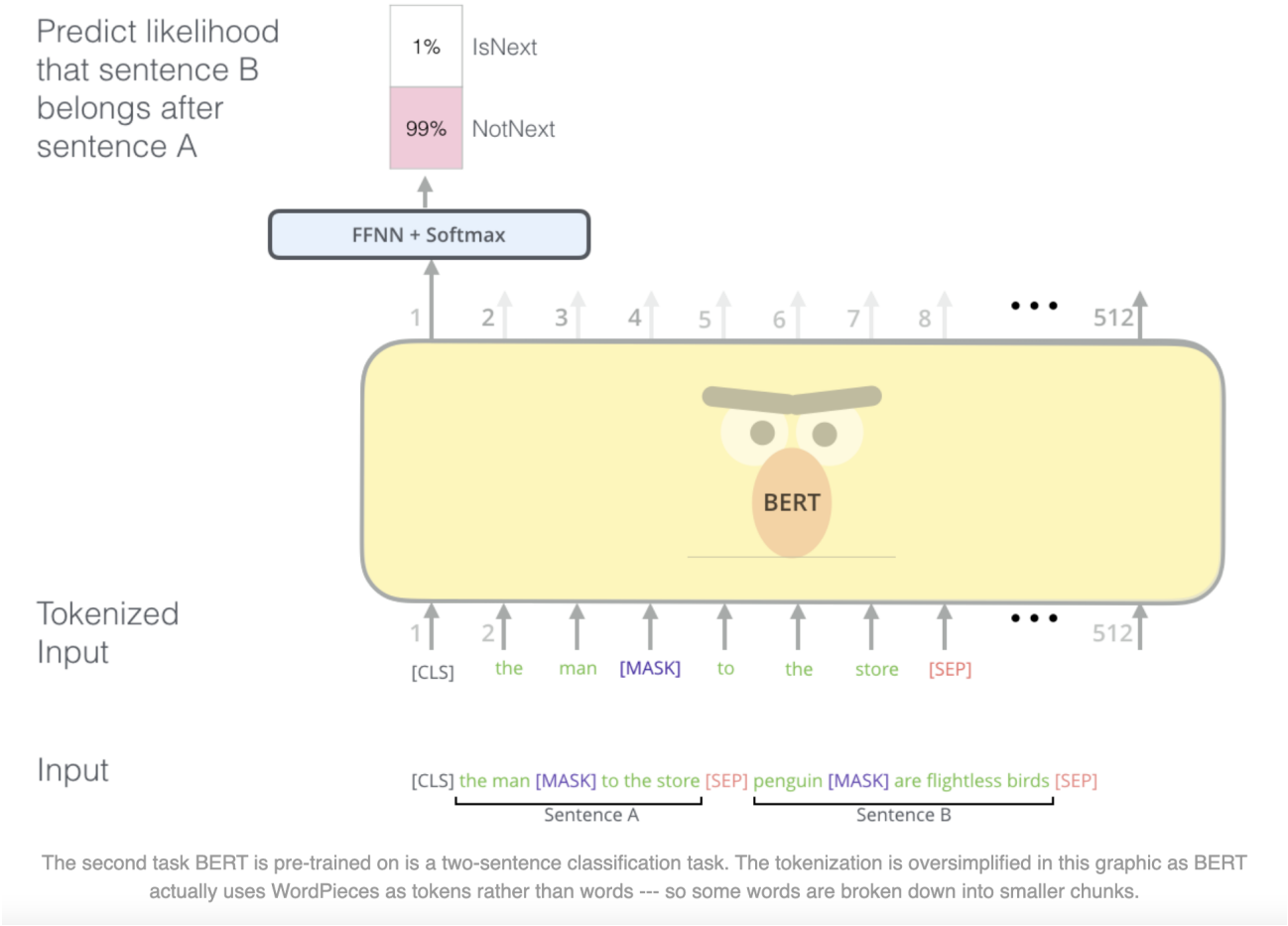
Trainable Tokenizer:
- WordPiece / Sentence Piece

BERT



The two steps of how BERT is developed. You can download the model pre-trained in step 1 (trained on un-annotated data), and only worry about fine-tuning it for step 2. [Source for book icon].

Alammar, J (2018). The Illustrated Transformer [Blog post]. Retrieved from https://jalammar.github.io/illustrated-transformer/



The second task BERT is pre-trained on is a two-sentence classification task. The tokenization is oversimplified in this graphic as BERT actually uses WordPieces as tokens rather than words --- so some words are broken down into smaller chunks.

Alammar, J (2018). The Illustrated Transformer [Blog post]. Retrieved from https://jalammar.github.io/illustrated-transformer/

Sentence Transformer

- feed the input sentence to BERT.
- BERT produces contextualized word embeddings
- Use pooling like mean pooling to get a single representation

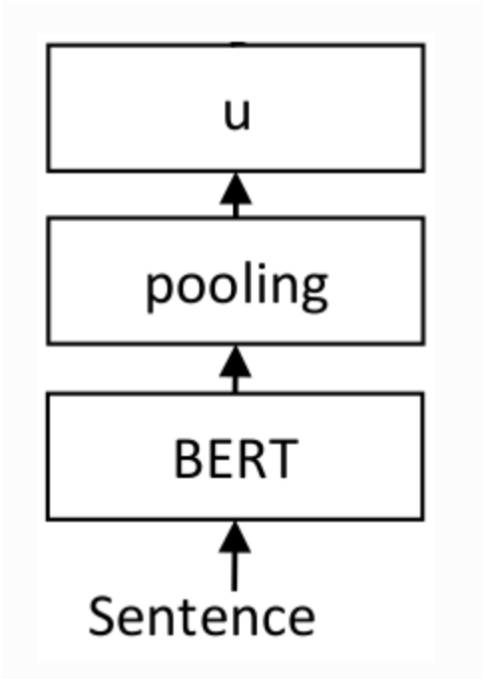


Image from sbert.net
https://www.sbert.net/docs/training/overview.html

Vision Transformers (ViT) / CLIP

- CLIP model uses two encoders (text and image).
- These two models are trained in parallel and optimized via a contrastive loss function — producing high similarity vectors for image-text pairs.

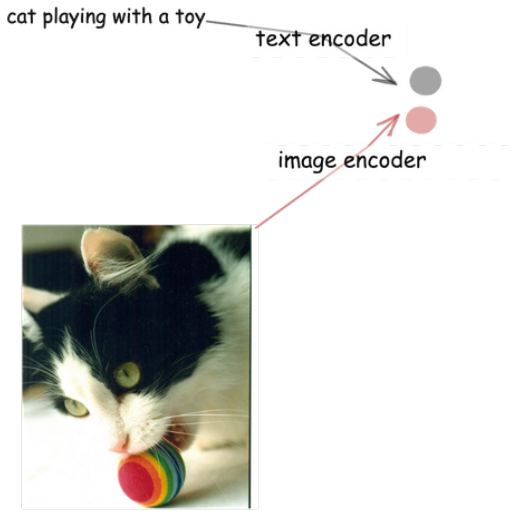
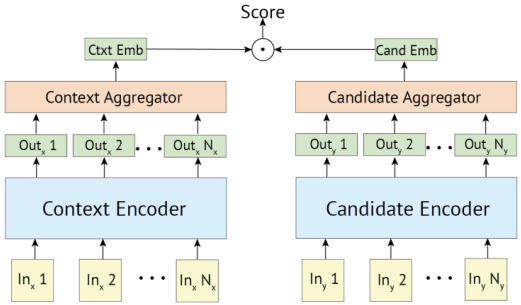
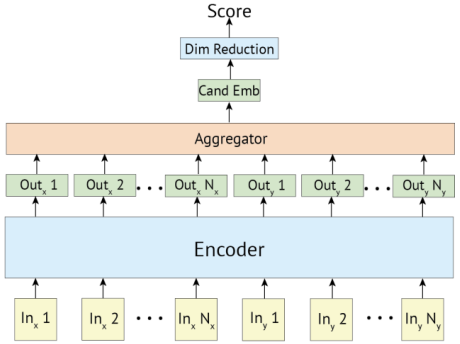


Image from Wikipedia “Cat play and toys”

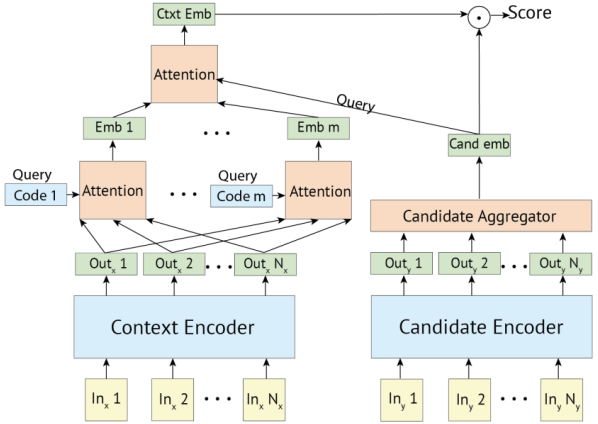
Semantic Similarity Architectures



(a) Bi-encoder



(b) Cross-encoder



(c) Poly-encoder

- Bi-Encoders : pass context and candidate separately; generate embedding and then compute score
- Cross Encoder: pass context and candidate together; does not generate embedding only provides score;
- Poly-Encoder: mixture of bi-encoder and cross-encoder; generate multiple embeddings for context

Humeau, Samuel, Kurt Shuster, Marie-Anne Lachaux, και Jason Weston. ‘Poly-encoders: Transformer Architectures and Pre-training Strategies for Fast and Accurate Multi-sentence Scoring’. arXiv, 2019. https://doi.org/10.48550/ARXIV.1905.01969.

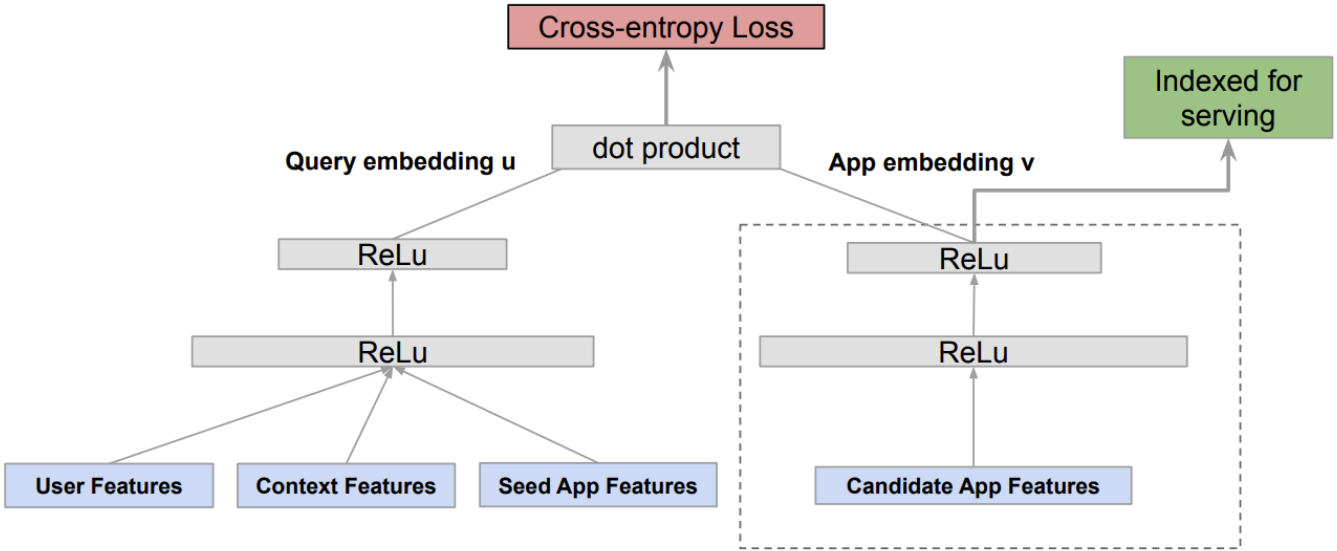


Figure 5: Two-tower model architecture for Google Play app recommendation.

Context or Candidate Encoder can include metadata from multiple fields

Yang, Ji, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, και Ed H. Chi. 'Mixed Negative Sampling for Learning Two-Tower Neural Networks in Recommendations'. Στο Companion Proceedings of the Web Conference 2020, 441–47. WWW '20. New York, NY, USA: Association for Computing Machinery, 2020. https://doi.org/10.1145/3366424.3386195.

Mining Training Data Signals

Positive Signals

Review (positive)

Like

Click

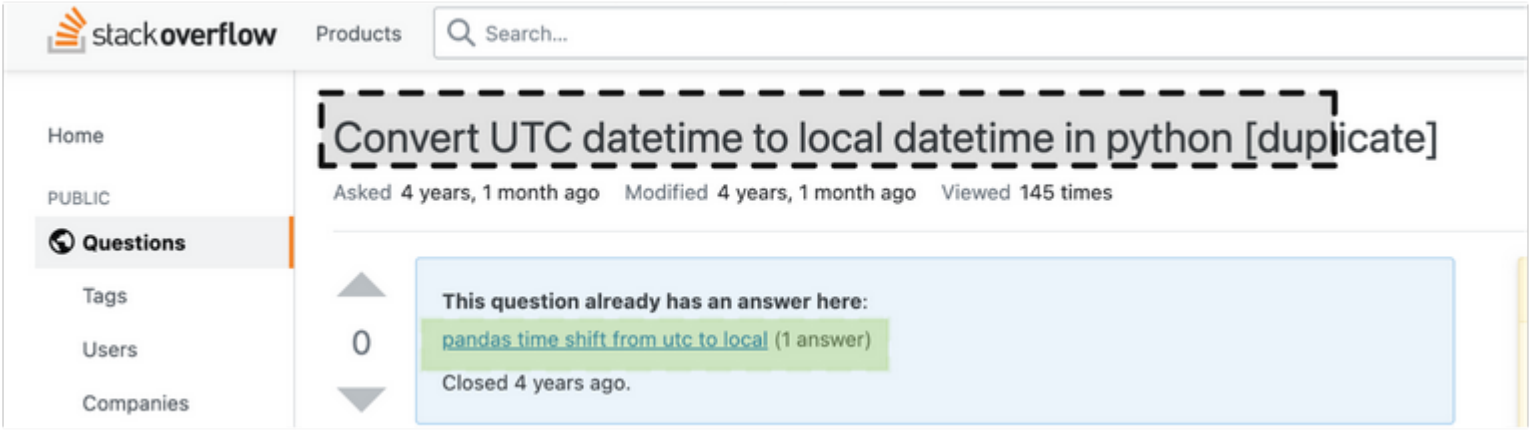
Negative Signals

Review (negative)

Dislike

Ignored Items

Random



ID	Anchor / QN	Positive Pair / Duplicate QN
1	pandas time shift from utc to local	Convert UTC datetime to local datetime in python
2	Shading a kernel density plot between two points.	r density plot - fill area under curve
3	How do you run a Python script as a service in Windows?	Deploy Flask app as windows service

Creating training data from StackOverflow Duplicate Questions

ID	Anchor / QN	Positive Pair / Duplicate QN
1	pandas time shift from utc to local	Convert UTC datetime to local datetime in python
2	Shading a kernel density plot between two points.	r density plot - fill area under curve
3	How do you run a Python script as a service in Windows?	Deploy Flask app as windows service

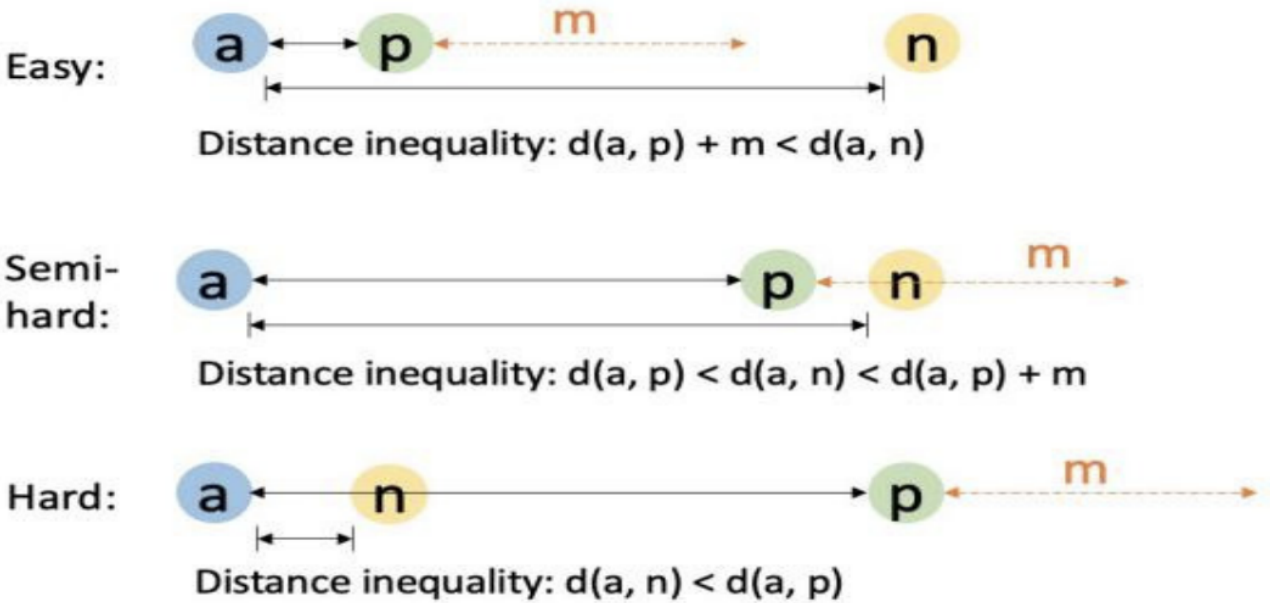
multiple negatives ranking

using alternate pairs in a batch, to create negatives

A1 P1	A1 P2	A1 P3
A2 P1	A2 P2	A2 P3
A3 P1	A2 P2	A3 P3

Creating negative training data from only positive pairs

Different types of negatives

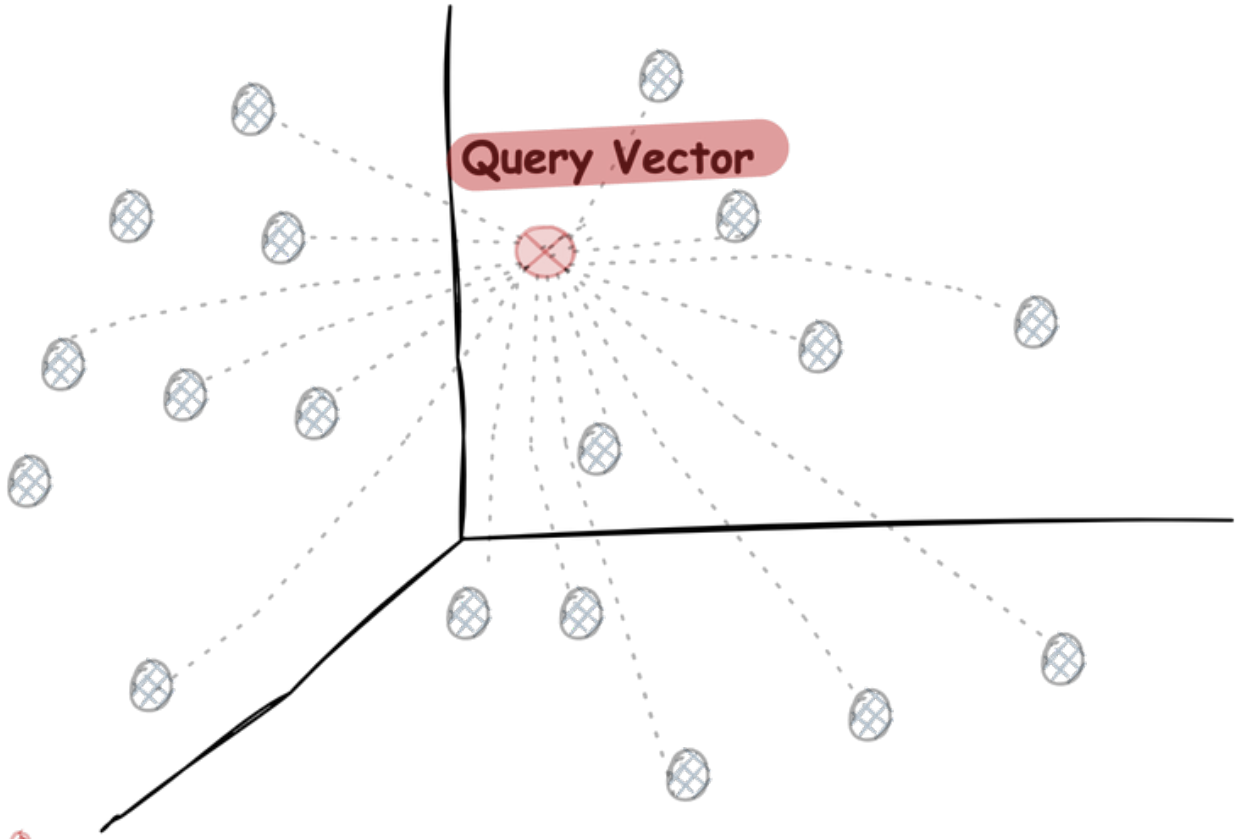


Wang, Zeyu, Xiaoyu Liang, και Cheng Wang. ‘Controllable Text-to-Image Generation with Enhanced Text Encoder and Edge-Preserving Embedding’. Journal of Physics: Conference Series 1856 (04 2021): 012003. https://doi.org/10.1088/1742-6596/1856/1/012003.

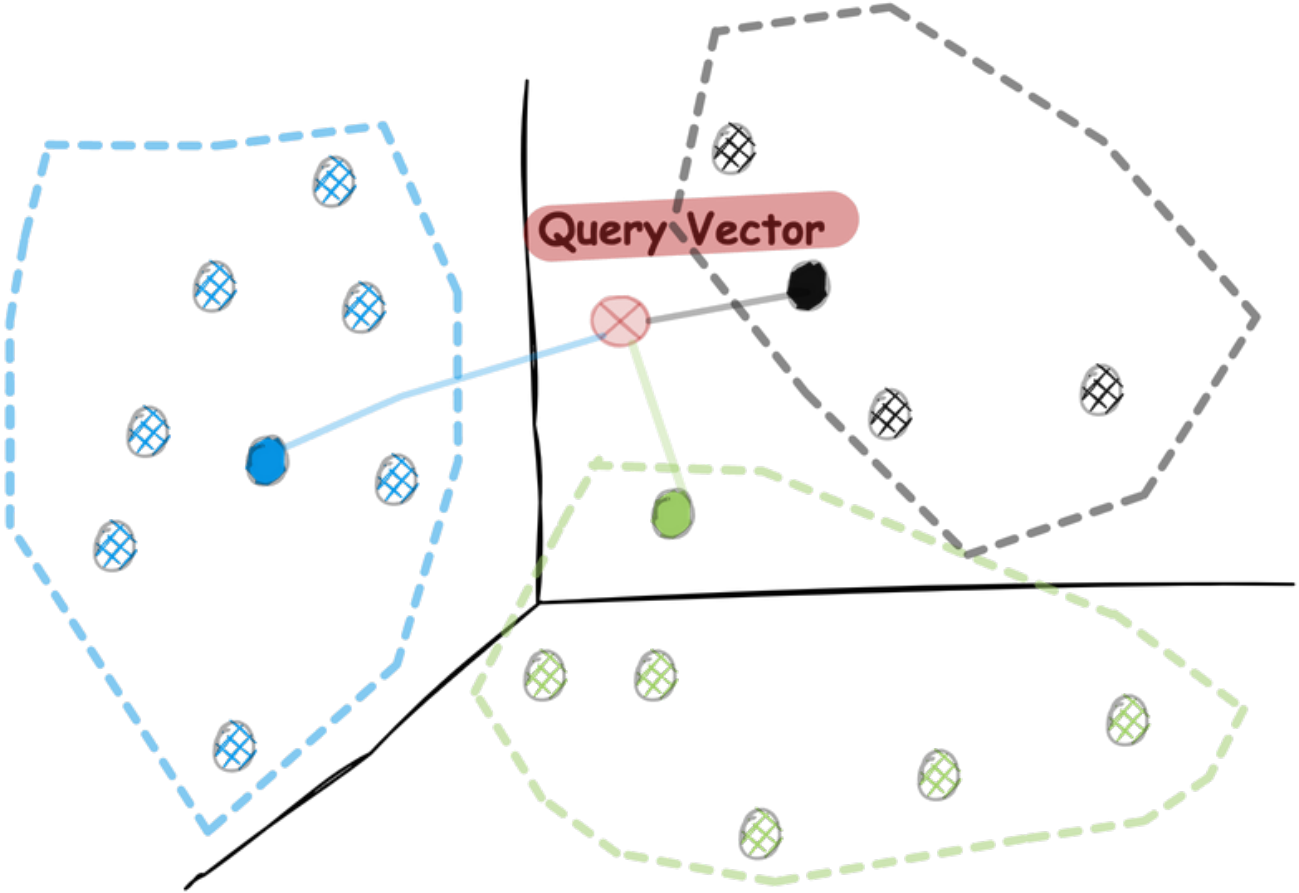
Approximate Nearest Neighbors

Outline

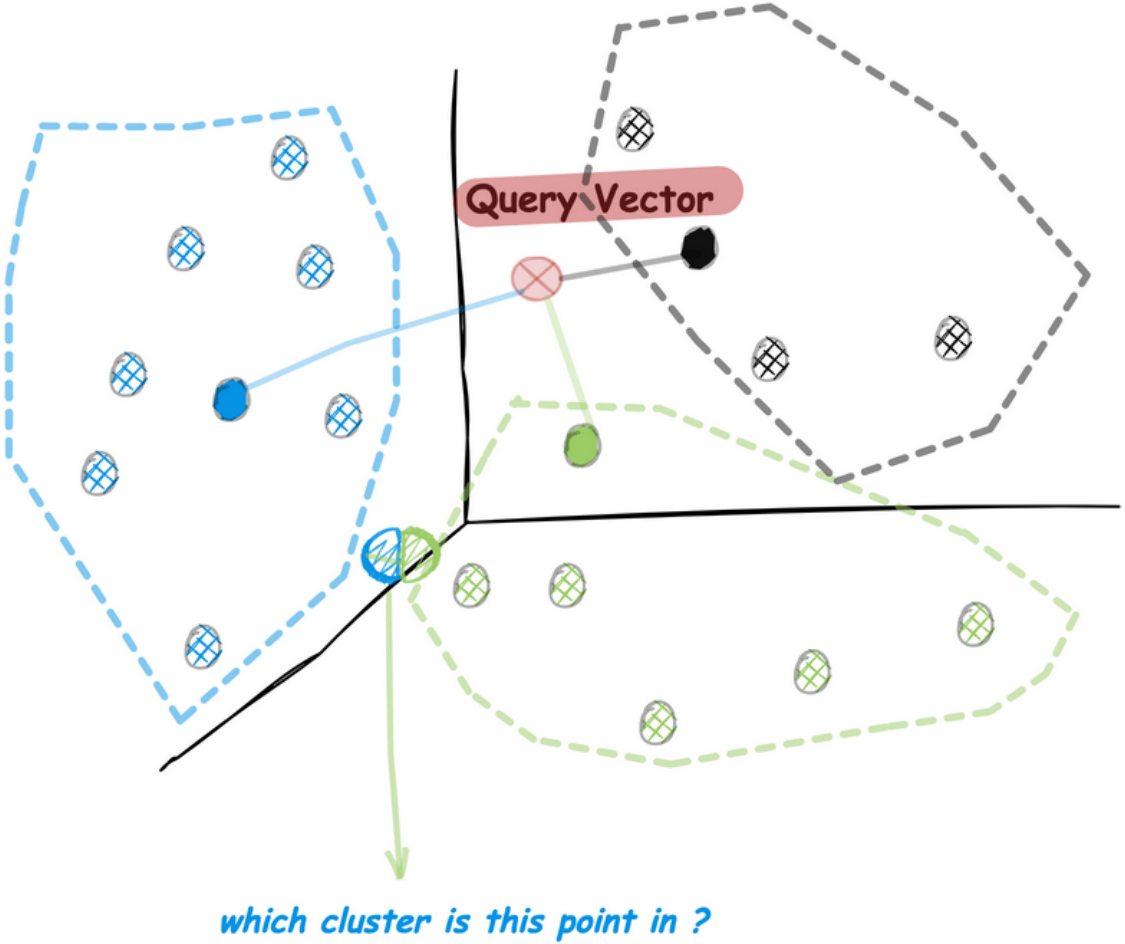
- Embedding Retrieval with no index
- Why we need Approximate Nearest Neighbor
- Indexing Algorithm: IVF
- Indexing Algorithm: HNSW
- Reducing Memory with Quantization
- ANN Benchmark



In a flat index / no index, our query vector is compared against all the vectors in the database.



Partition the vector space and compare against partition centroids



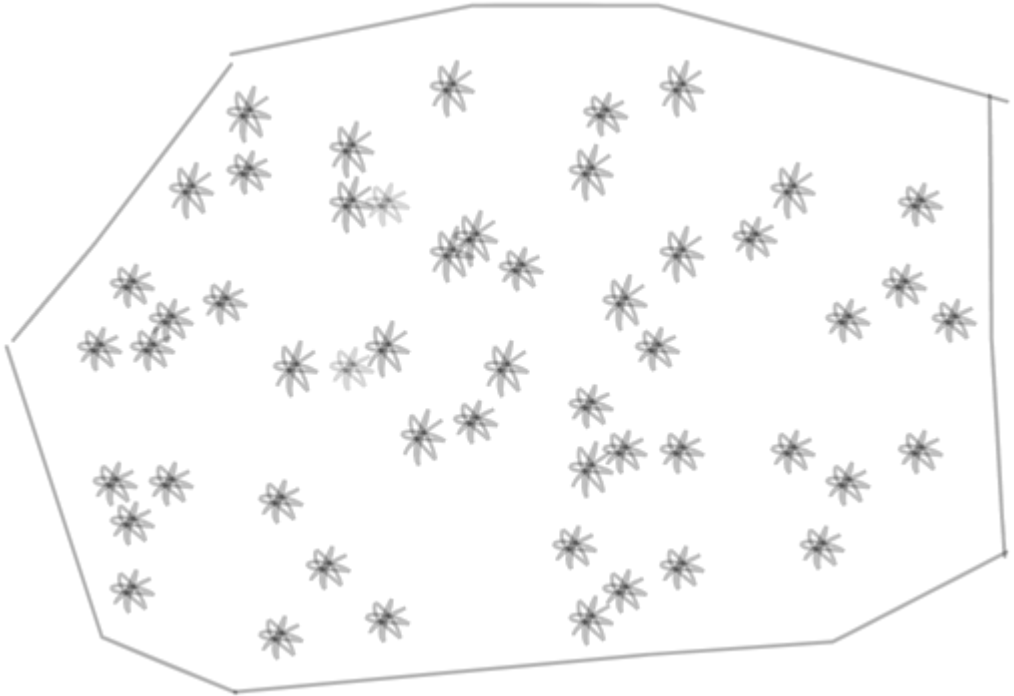
Depending on the partitioning, you will miss candidates in your search.

Hence the “Approximate” Nearest Search

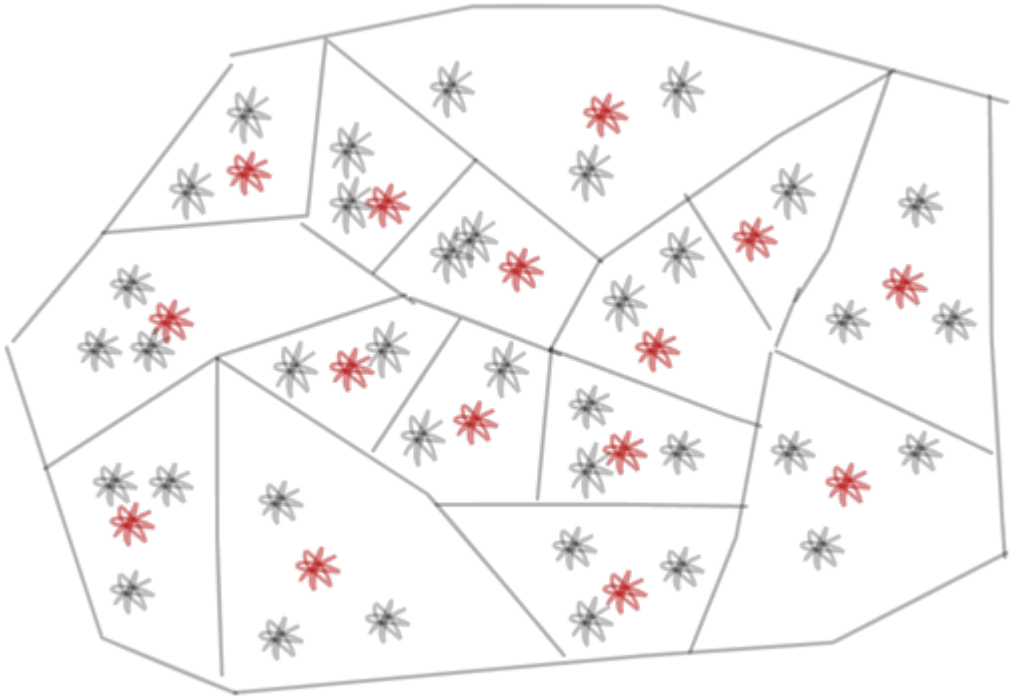
Types of ANN

- Inverted File Index
- Hierarchical Navigable Small World (HNSW)

Inverted File Index: Building

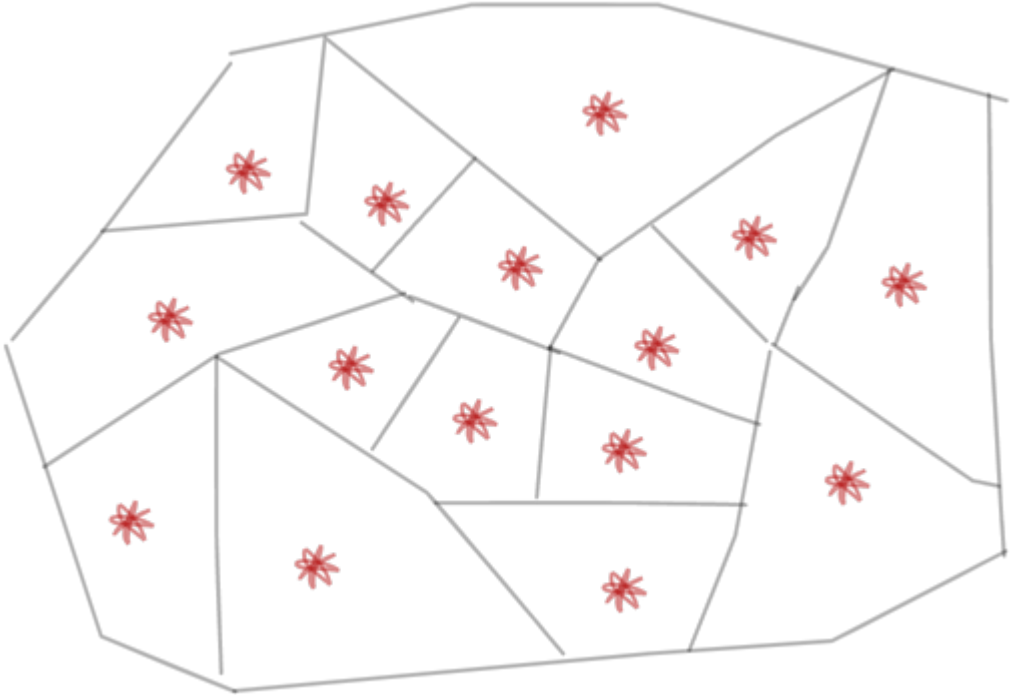


Inverted File Index: Building



- Find centroid and create Voronoi Cells
- Number of centroids is determined by **nlist** parameter

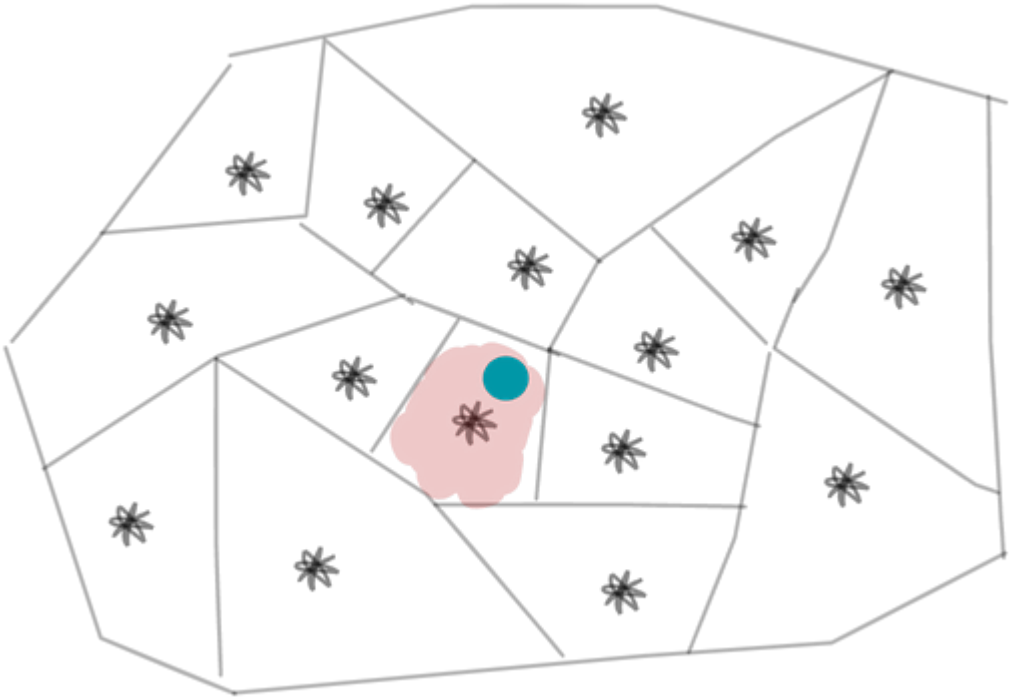
Inverted File Index: Building



- Built Voronoi Index with 15 centroids centroids

- Memory usage is not reduced
- But retrieval is faster

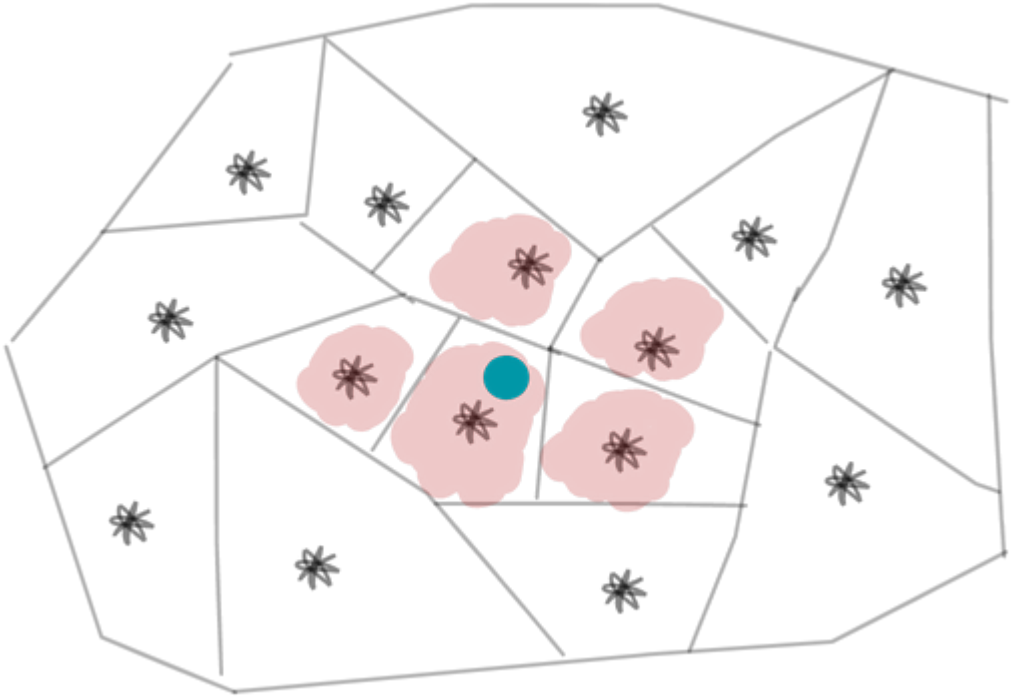
Inverted File Index: Searching



nprobe = 1

- Find the distance between query vector and all the centroids
- Query all the elements inside the n closest cluster determined by **nprobe** parameter

Inverted File Index: Searching



nprobe = 5

- Increasing the value of nprobe, improves recall but increases latency
- If nprobe=nlist, similar to flat index

Hierarchical Navigable Small World (HNSW)

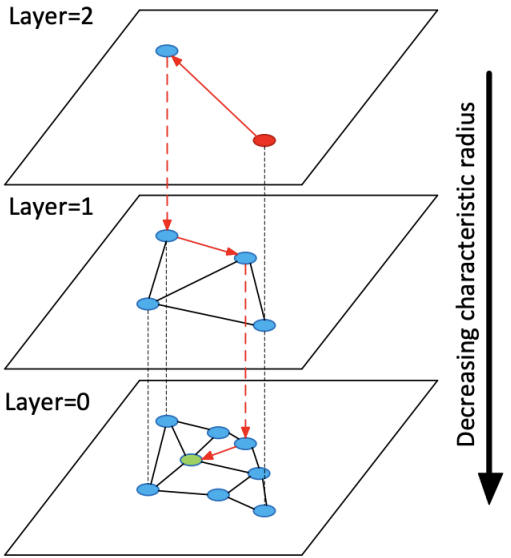


Fig. 1. Illustration of the Hierarchical NSW idea. The search starts from an element from the top layer (shown red). Red arrows show direction of the greedy algorithm from the entry point to the query (shown green).

Search Parameters:

- ef_search: size of list used during neighbor search

Construction Parameters:

- **M**: numbers of neighbors a new node is connected to
- **ef_construction**: size of list used during index construction

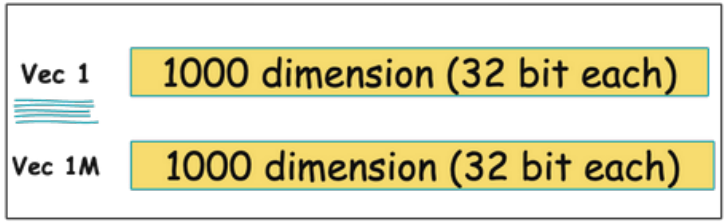
Malkov and Yashunin, “Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs.” https://arxiv.org/abs/1603.09320

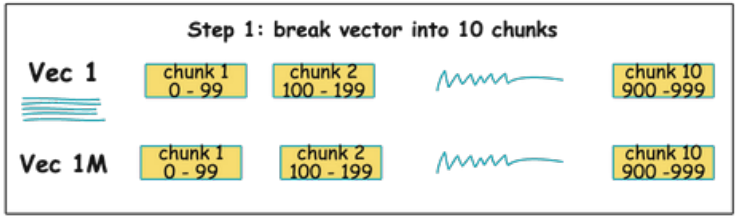
ANN Memory requirements

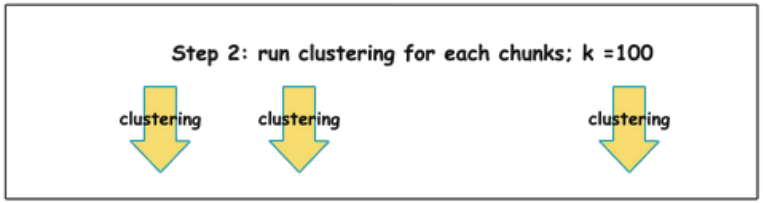
algorithm\# vectors	1M	10M	100M
FLAT	6GB	57GB	572GB
IVF_FLAT	6GB	57GB	572GB
HNSW (M=20)	20GB	60GB	602 GB
IVF_SQ8	2GB	21GB	214 GB

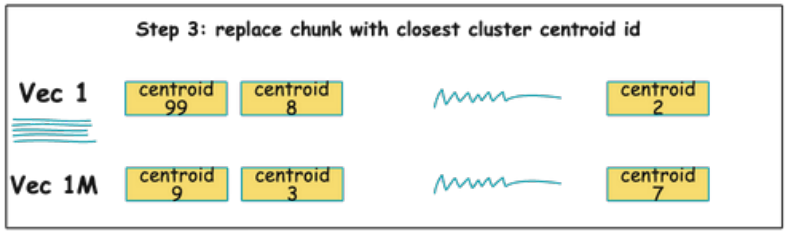
Memory needed for 768 dimension vectors
https://milvus.io/tools/sizing/

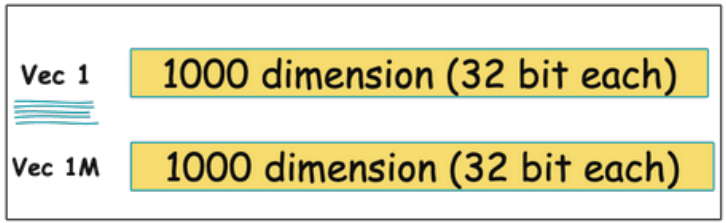
Quantization





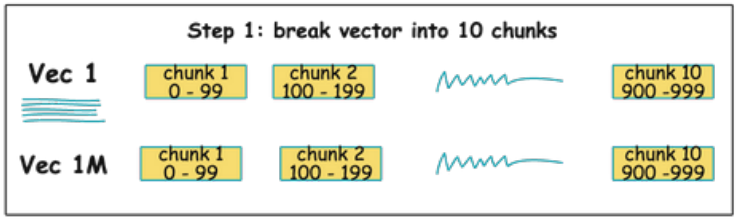


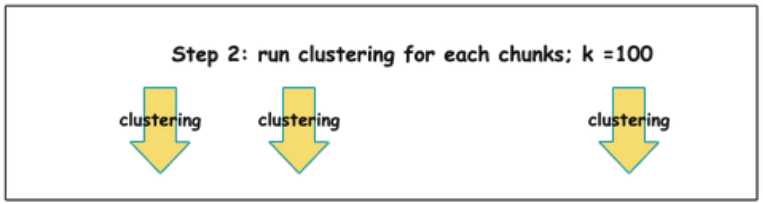




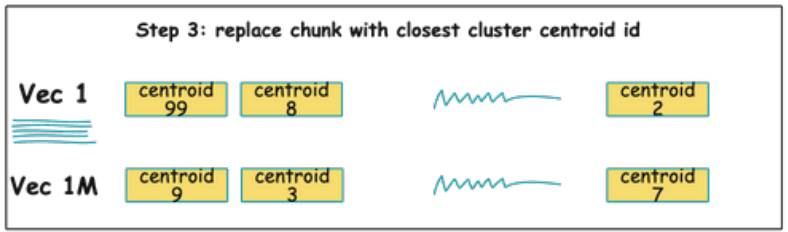
Single Vector: 4000 bytes

Total: 4 GB





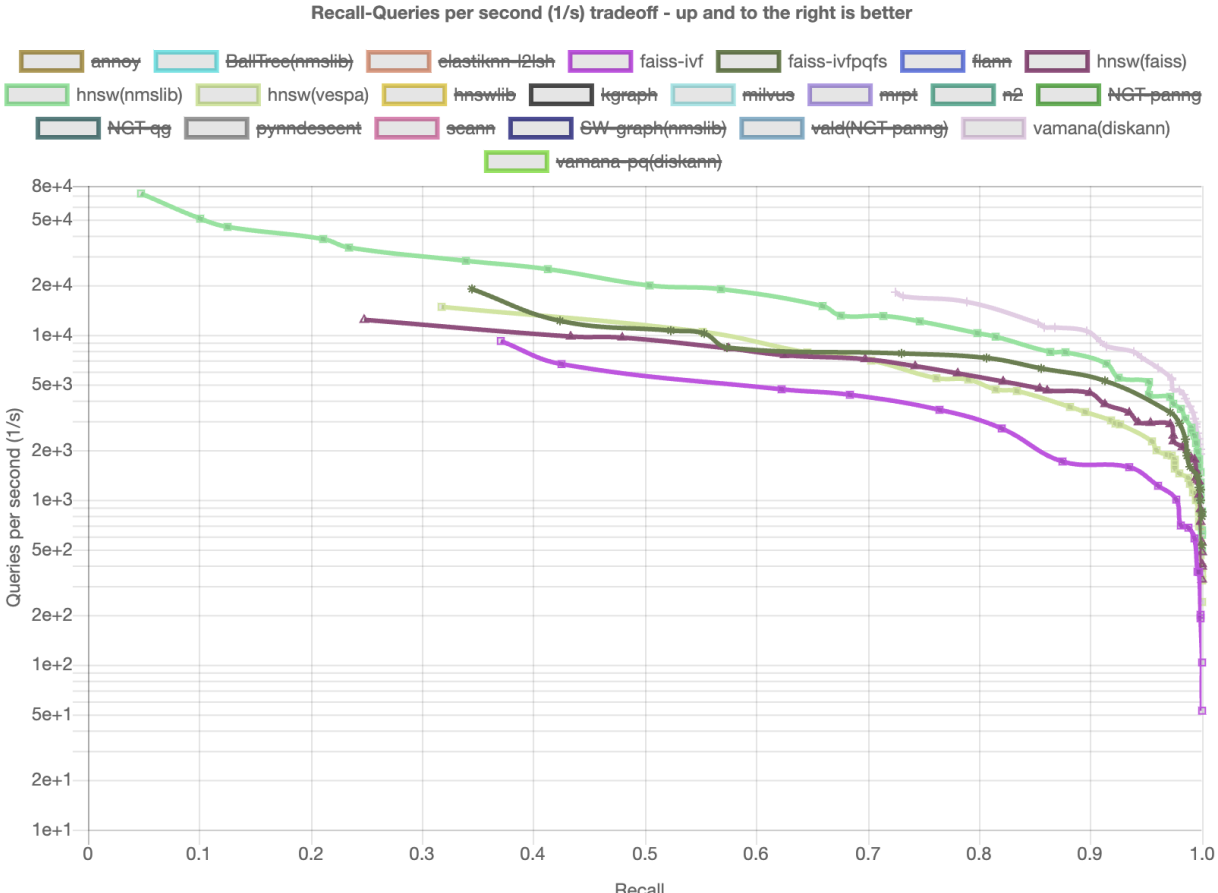
100 ids can be represented with 7 bits



Single Vector : 10 chunks * 7 bit = 9 bytes

Total: 8MB

ANN Benchmarks



Source:
http://ann-benchmarks.com

ANN Solutions

Lot of ANN Option

Evaluation Considerations

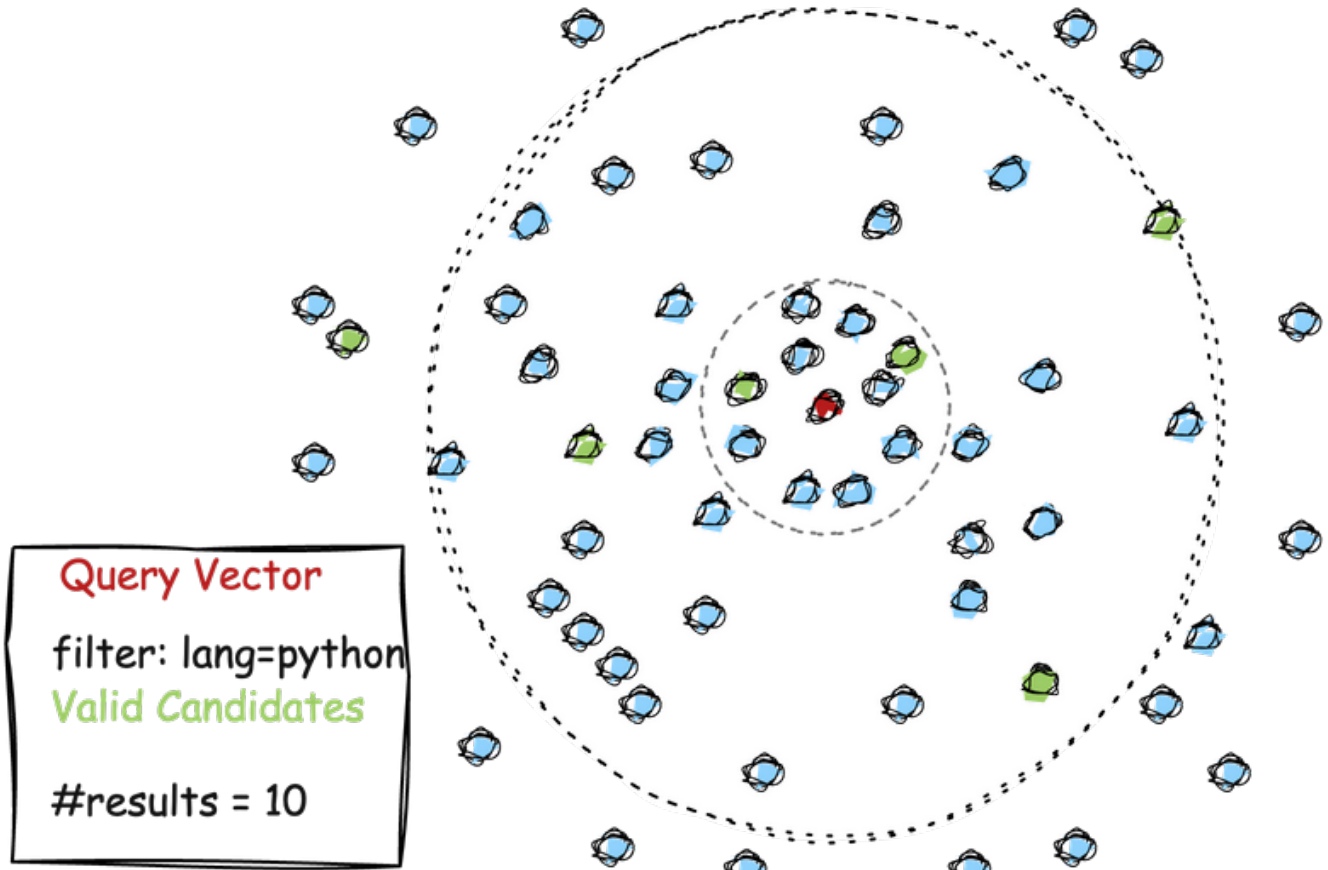
- Managed vs Self-hosted
- Performance
- Update Embeddings / Partial Updates
- Metadata Filtering
- Filtering / Hybrid Retrieval
- Plugins

Update Embeddings / Partial Inserts

Support:

Lack of Support

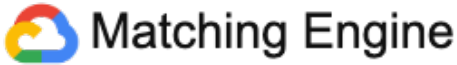
Filtering



Closest 10 candidates don’t meet our filter

Pre Filtering

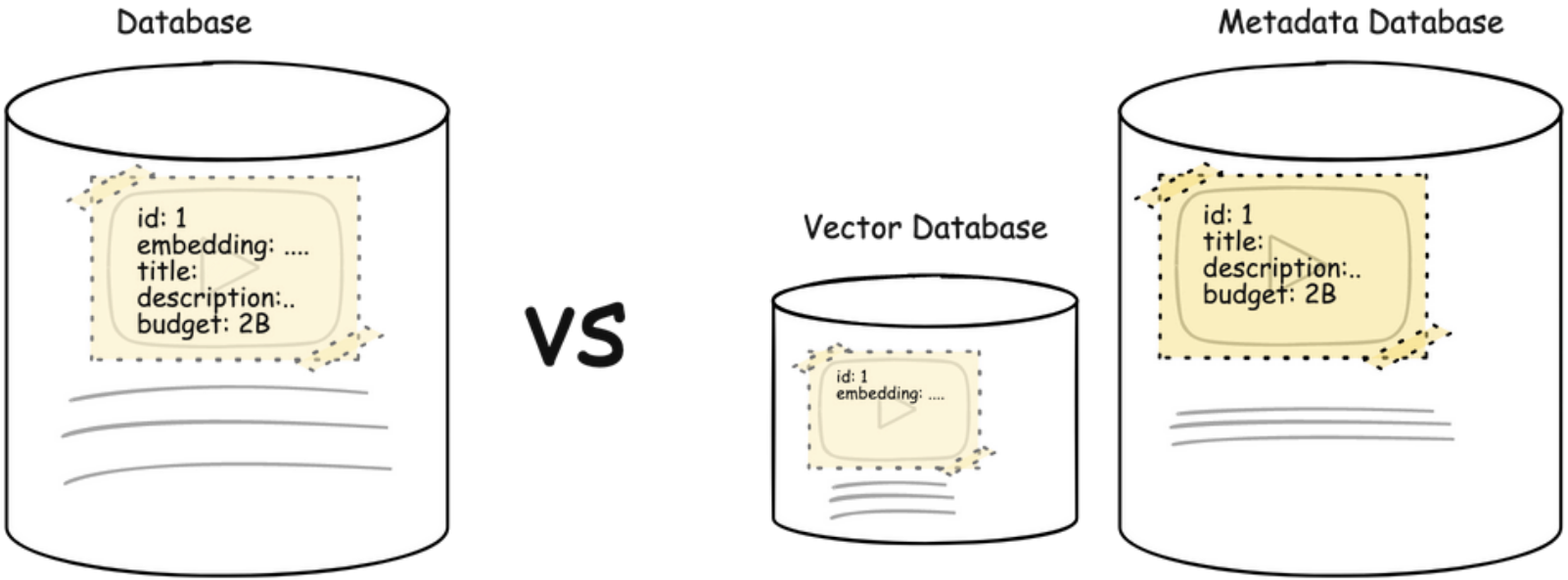
Support



No Support

- Create bloom filter for each filter
- Use bloom filter when traversing the HNSW graph
- Default to brute force if filter is too restrictive

Metadata


Database
id: 1
embedding: ....
title:
description:..
budget: 2B
VS
Vector Database
id: 1
embedding: ....
Metadata Database
id: 1
title:
description:..
budget: 2B

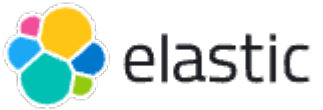
Metadata

Support

Lack of Support

Hybrid / Full Retrieval

Supports



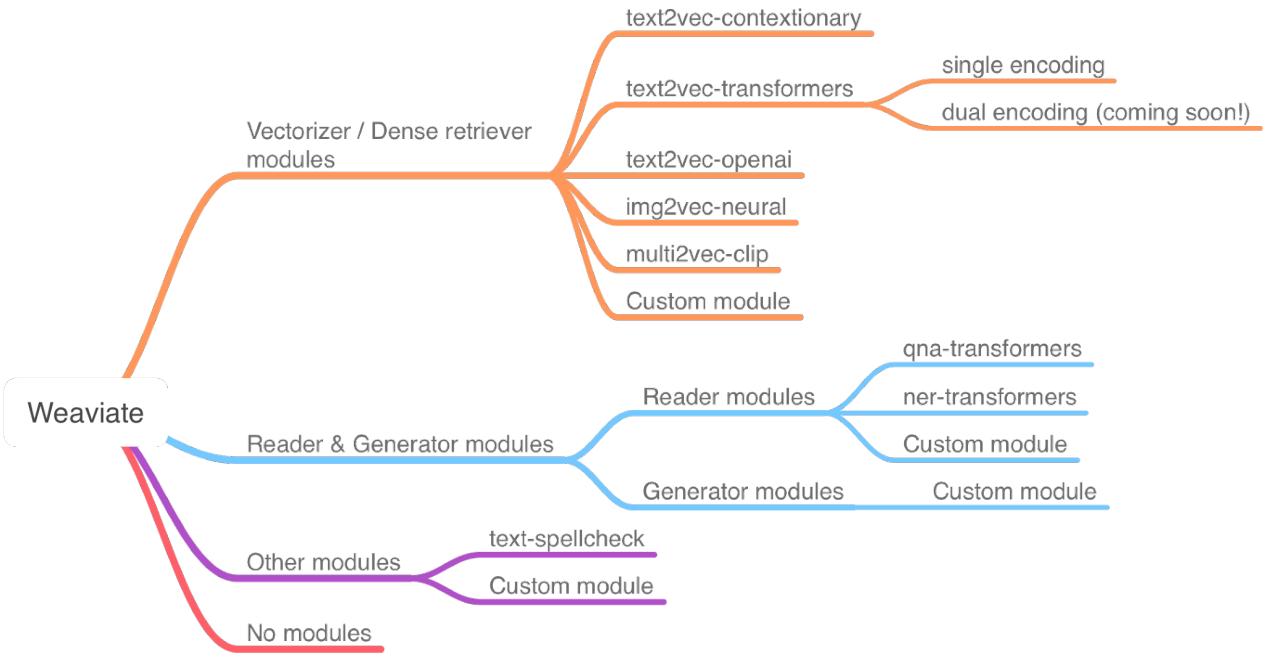
- In ES, **disjunction** of knn and bm25 match.
- The score of each hit is the sum of the knn and query scores. A boost can be specified

```
POST image-index/_search
{
  "query": {
    "match": {
      "title": {
        "query": "mountain lake",
        "boost": 0.9
      }
    }
  },
  "knn": {
    "field": "image-vector",
    "query_vector": [54, 10, -2],
    "k": 5,
    "num_candidates": 50,
    "boost": 0.1
  },
  "size": 10
}
```

ElasticSearch example of Hybrid Retrieval

Ex from Elastic Doc link

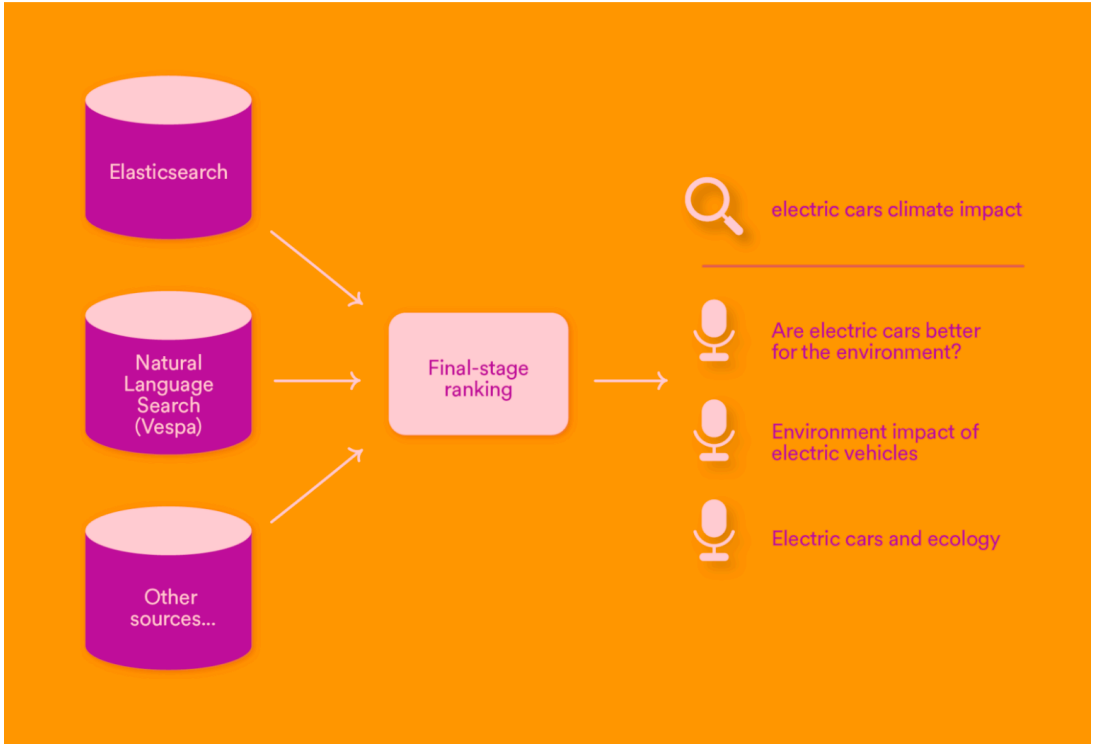
Plugins / Ecosystem



Weaviate Module Ecosystem
From https://weaviate.io/developers/weaviate/current/modules/index.html#module-ecosystem

Conclusion

Multi Source Retrieval

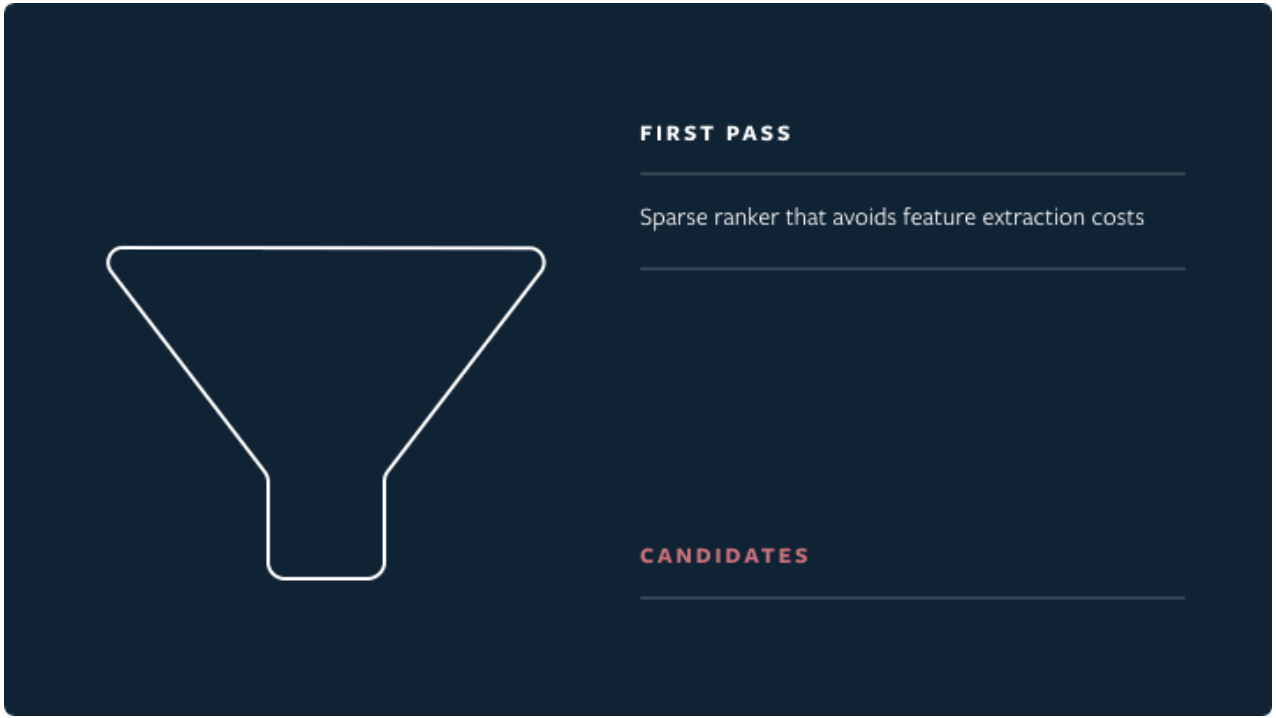


Spotify Podcast Episode Search

Retrieval from ES + ANN + other sources

Tamborrino, Alexandre . “Introducing Natural Language Search for Podcast Episodes - Spotify Engineering.” Introducing Natural Language Search for Podcast Episodes, March 17, 2022. https://engineering.atspotify.com/2022/03/introducing-natural-language-search-for-podcast-episodes/.

Multi Stage Retrieval and Ranking

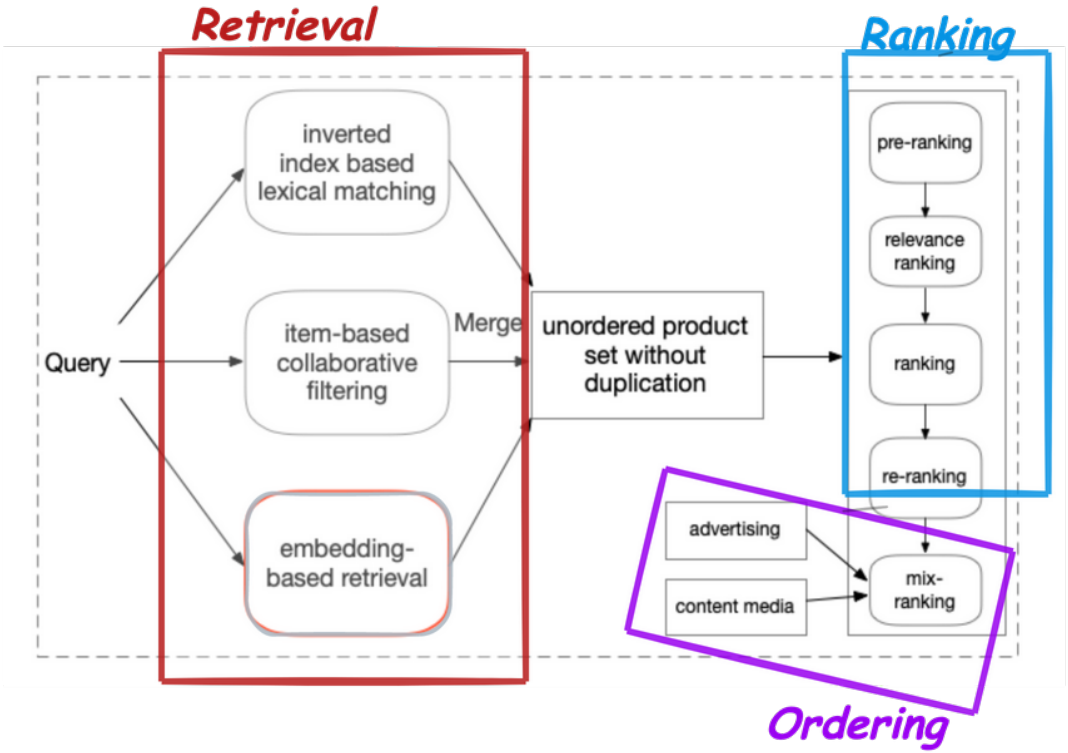


In 2019, Instagram Explore feed was made of

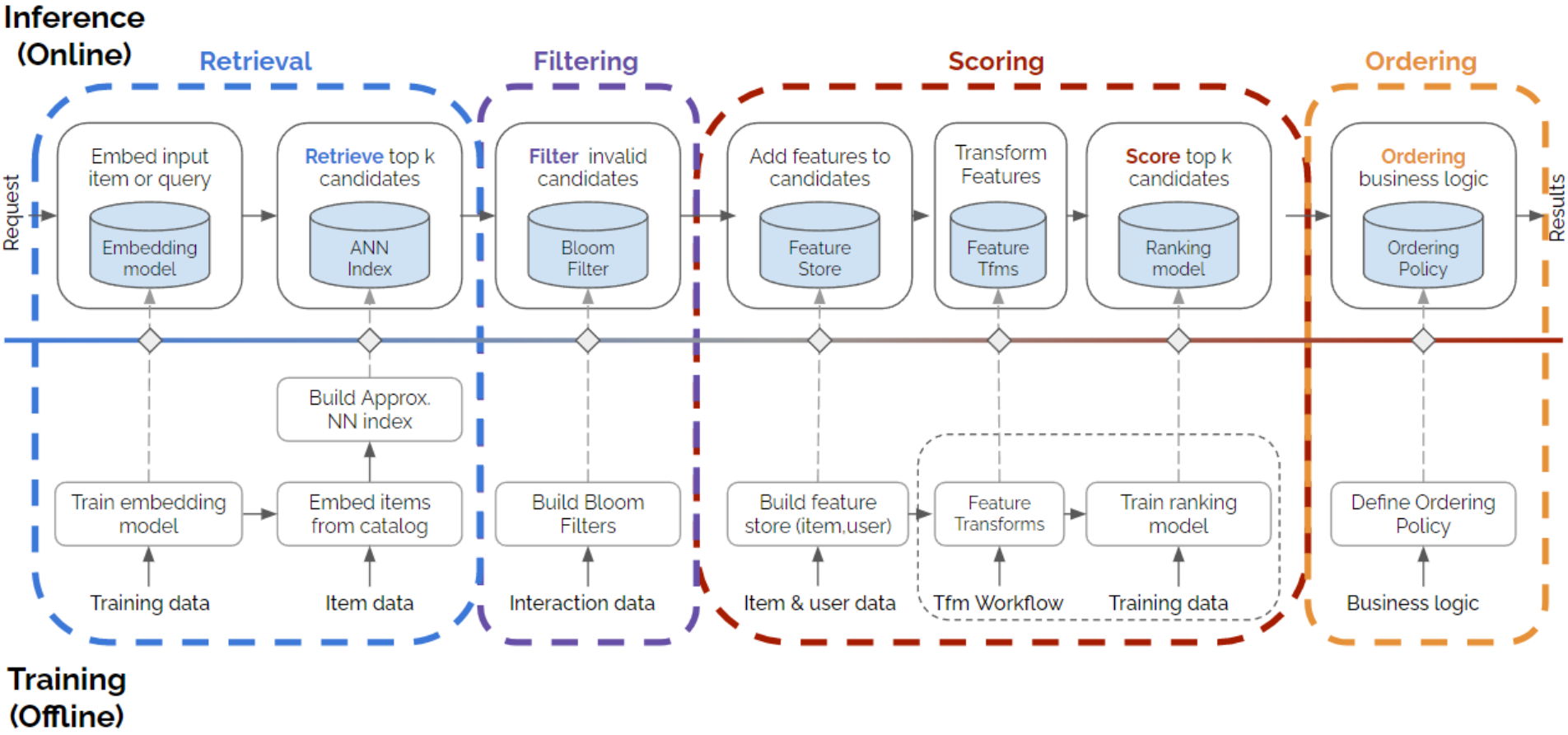
- Sparse Retriever: 500 candidates
- Lightweight NN: 150 ->50
- Deep Neural Network: 50 -> 25

Medvedev, Ivan, Haotian Wu, and Taylor Gordon. “Powered by AI: Instagram’s Explore Recommender System.” Powered by AI: Instagram’s Explore recommender system, November 2019. https://ai.facebook.com/blog/powered-by-ai-instagrams-explore-recommender-system/

Overview of Taobao Search



Li, Sen, Fuyu Lv, Taiwei Jin, Guli Lin, Keping Yang, Xiaoyi Zeng, Xiao-Ming Wu, και Qianli Ma. 'Embedding-based Product Retrieval in Taobao Search'. arXiv, 2021. https://doi.org/10.48550/ARXIV.2106.09297.

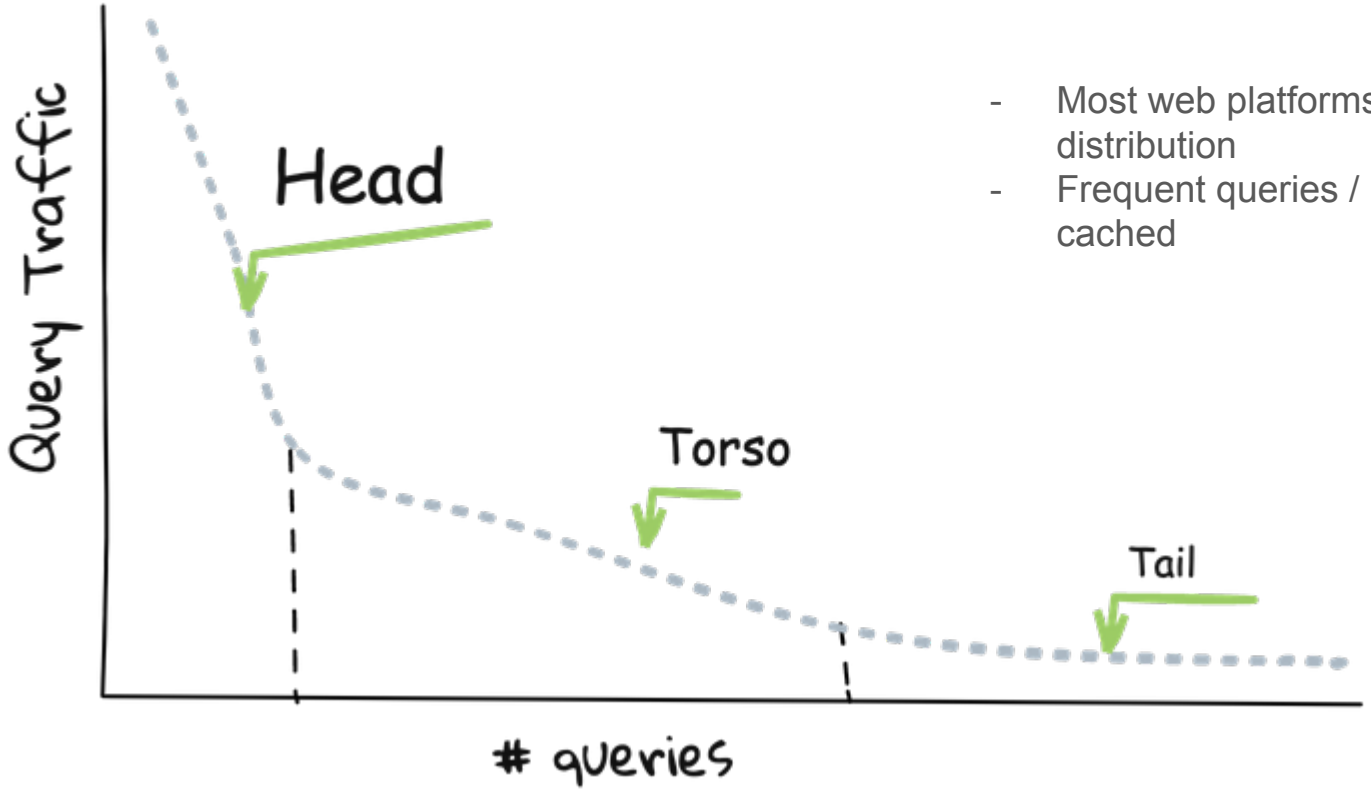


Oldridge, Even. “Recommender Systems, Not Just Recommender Models | by Even Oldridge | NVIDIA Merlin | Medium.” Medium, June 27, 2022. https://medium.com/nvidia-merlin/recommender-systems-not-just-recommender-models-485c161c755e.

	Retrieval	Filtering	Scoring	Ordering
Music Discovery	Find similar songs based on nearest neighbour search	Remove tracks users listened before	Predict likelihood that a user will listen to a song	Trade-Off between score, similarity, BPM, etc
Social Media	Find new posts in user's network	Remove posts from blocked and muted users	Predict likelihood that a user will interact with it	Change order that adjust posts are from different authors
Online Store	Find items which are usually co-purchased	Remove items which are out of stock	Predict likelihood that a user will purchase an item	Reorder items based on price points
Streaming Service	Find items based on different rows/shelves/topics	Remove items which are not available for user's country	Predict user's stream time per item	Organize recommendations to fit genre distributions

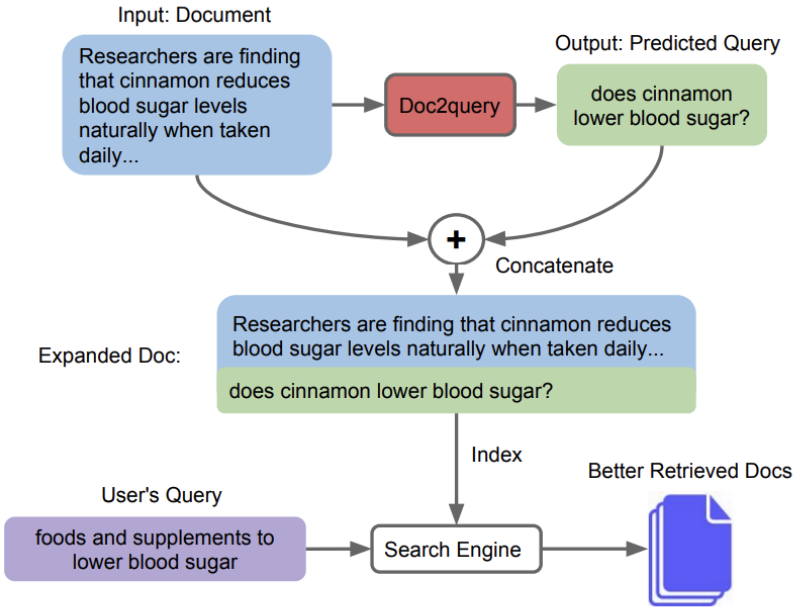
Oldridge, Even. "Recommender Systems, Not Just Recommender Models | by Even Oldridge | NVIDIA Merlin | Medium." Medium, June 27, 2022. https://medium.com/nvidia-merlin/recommender-systems-not-just-recommender-models-485c161c755e.

Zipfian Distribution of Web Traffic



- Most web platforms traffic follows a zipfian distribution
- Frequent queries / head traffic can be cached

Enhancing Sparse Index: Doc2Query / Doc2T5Query



- Use a causal language model to generate additional text to add to documents when indexing.
- At retrieval time, use BM25

Nogueira, Rodrigo, Wei Yang, Jimmy Lin, και Kyunghyun Cho. ‘Document Expansion by Query Prediction’. arXiv, 2019. https://doi.org/10.48550/ARXIV.1904.08375.

Benchmark

Model (→)	Lexical	Sparse			Dense				Late-Interaction	Re-ranking
Dataset (↓)	BM25	DeepCT	SPARTA	docT5query	DPR	ANCE	TAS-B	GenQ	ColBERT	BM25+CE
MS MARCO	0.228	0.296‡	0.351‡	0.338‡	0.177	0.388‡	0.408‡	0.408‡	0.401‡	**0.413**‡
TREC-COVID	0.656	0.406	0.538	0.713	0.332	0.654	0.481	0.619	0.677	**0.757**
BioASQ	0.465	0.407	0.351	0.431	0.127	0.306	0.383	0.398	0.474	**0.523**
NFCorpus	0.325	0.283	0.301	0.328	0.189	0.237	0.319	0.319	0.305	**0.350**
NQ	0.329	0.188	0.398	0.399	0.474‡	0.446	0.463	0.358	0.524	**0.533**
HotpotQA	0.603	0.503	0.492	0.580	0.391	0.456	0.584	0.534	0.593	**0.707**
FiQA-2018	0.236	0.191	0.198	0.291	0.112	0.295	0.300	0.308	0.317	**0.347**
Signal-1M (RT)	0.330	0.269	0.252	0.307	0.155	0.249	0.289	0.281	0.274	**0.338**
TREC-NEWS	0.398	0.220	0.258	0.420	0.161	0.382	0.377	0.396	0.393	**0.431**
Robust04	0.408	0.287	0.276	0.437	0.252	0.392	0.427	0.362	0.391	**0.475**
ArguAna	0.315	0.309	0.279	0.349	0.175	0.415	0.429	**0.493**	0.233	0.311
Touché-2020	**0.367**	0.156	0.175	0.347	0.131	0.240	0.162	0.182	0.202	0.271
CQADupStack	0.299	0.268	0.257	0.325	0.153	0.296	0.314	0.347	0.350	**0.370**
Quora	0.789	0.691	0.630	0.802	0.248	0.852	0.835	0.830	**0.854**	0.825
DBPedia	0.313	0.177	0.314	0.331	0.263	0.281	0.384	0.328	0.392	**0.409**
SCIDOCS	0.158	0.124	0.126	0.162	0.077	0.122	0.149	0.143	0.145	**0.166**
FEVER	0.753	0.353	0.596	0.714	0.562	0.669	0.700	0.669	0.771	**0.819**
Climate-FEVER	0.213	0.066	0.082	0.201	0.148	0.198	0.228	0.175	0.184	**0.253**
SciFact	0.665	0.630	0.582	0.675	0.318	0.507	0.643	0.644	0.671	**0.688**
Avg. Performance vs. BM25	- 27.9%	- 20.3%	+ 1.6%	- 47.7%	- 7.4%	- 2.8%	- 3.6%	+ 2.5%	+ 11%	

BM25 is a strong benchmark ✓

cross-attentional re-ranking model generalize to out of domain ✓

Dense Embeddings don't work well for out of domain !

Do models trained on MS MARCO work for different datasets ?

Thakur, Nandan, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, και Iryna Gurevych. 'BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models'. arXiv, 2021. https://doi.org/10.48550/ARXIV.2104.08663.

Cost to Serve

Model	Dimension	Latency (CPU)	Latency (GPU)	Index Size
BM-25	-	20 ms	-	0.4 GB
docT5Query	-	30 ms	-	0.4 GB
Dense Passage Retrieval	768	230 ms	19 ms	3 GB
ColBERT (polyencoder)	128	350 ms	-	20 GB
BM25 + Cross Encoder	-	6100 ms	450 ms	0.4 GB

Estimated average retrieval latency and index sizes for a single query in DBPedia (1 million docs)

Lower Latency and Index Size is preferred

CPU: 8 core Intel Xeon Platinum 8168 CPU @ 2.70GHz

GPU: 1 Nvidia Tesla V100

Thakur, Nandan, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, και Iryna Gurevych. 'BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models'. arXiv, 2021. https://doi.org/10.48550/ARXIV.2104.08663.

FIN