HYPERSPACE

Mastering Hybrid Search

Blending Classic Ranking Functions with Vector Search for Superior Search Relevance

linkedin.com/company/hyperspace-db/

ohadl@hyper-space.io

About Me

Professional:

- Technology builder
- Obsessed by Data, AI and Performance
- Built Data-Intensive applications from zero to 1 million users

Personal:

- Based in Tel Aviv
- Father of 3 amazing daughters
- Love scuba diving, hiking and recently skiing

Started Hyperspace to build the world's fastest search engine

Ohad Levi
Co-Founder and CEO

Hyperspace Intro

Vision Building the world's fastest search engine

- **Based:** Tel Aviv
- **Fundraising:** raised $7.5m to date
- **Team:** 20 employees (mostly R&D)
- **Customers:** running large-scale pilots at domain market leaders

Product

Real time Hybrid search combining

Classic Search + Vector Search

Technology

Designing a virtual chip for search delivering 10x performance, at billion-scale, and at 50% of the compute costs

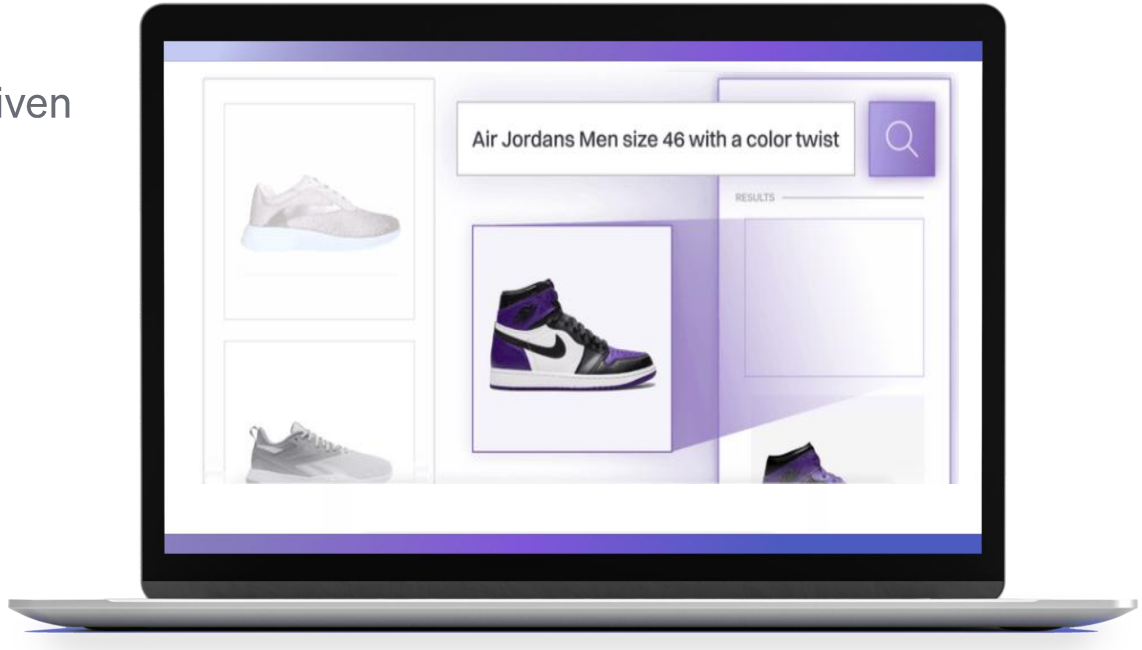
Today’s talk

- Classic search (lexical) - great, but not perfect
- Vector Search – does it comes to replace or compliment?
- Hybrid Search - taking search to the next level
- Tips & Tricks
- From POC to production

Relevance is the key

Relevance is a critical KPI across many industries:

- **E-Commerce** - returning the top products per user search is the difference between a purchase to a frustrated consumers.
- **Fraud Prevention** – finding the top similar transactions is the differentiator between successful fraud identification and false detection (false positive).
- **Recommendation systems** – personalizing the results to a given user is the essence in providing good user experience.



To All the Boys
P.S. I Still Love You
QUEER EYE
NETFLIX
ALTERED
CARBON
NETFLIX
OZARK
THE WITCHER

Classic Search - Recap

The Evolution of Terms Frequency (BM25)

Term-Frequency
(mid 20th century)
Rank according to the query terms frequency in each document

TF-IDF
(late 20th century)
Frequency terms are considered less indicative

Okapi BM25
(late 1990s)
Limit TF impact for common words, adjust ranking by document length

BM25F
(2006)
Include document structure, adjust for varying field importance

BM25 Score - Recap

Repetitions → Higher score

$$BM25\ score(Document, Query) = \sum_{w \in Words} \frac{d_w}{d_w + k_1(1 + b(\overline{DL} - 1))} log\left(1 + \frac{N - m_w + 0.5}{m_w + 0.5}\right)$$

More matched words → Higher score

Long document → Lower score

Common words → Lower score

d_w = No. appearances of W in document (D)

$\overline{DL}$ = normalized document length

N = No. documents in corpus

m_w = No. documents with word W

The Strengths of Keyword Matching

- **Fast and Efficient:** BM25 is fast and efficient, especially for large-scale datasets.
- **Explainable**: It's often simple to explain why a particular result was returned based on keyword matching.
- **Doesn’t Require Training:** BM25 is based on statistical properties of the dataset and doesn't need explicit training.

Keyword Search is great, but not perfect

The Challenges with Lexical Search

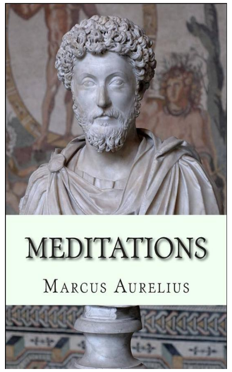
- **Low Recall:** Might miss relevant documents if they don't contain the exact keyword, or ‘vocabulary mismatch’.
- **Unstructured Data:** Struggles with non-textual data like images, voice, or video.
- **Fails to capture context and semantics:** Keyword based search fails to incorporate the effects of word order, sentence structure, etc.
- **Affected by Ambiguity:** Having terms that have different meanings that leads to lower precision - Apple fruit vs Apple stock

BM25 Example - Limited Relevance

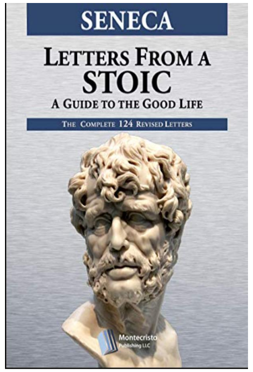
A user is searching for "books similar to writings of Marcus Aurelius."

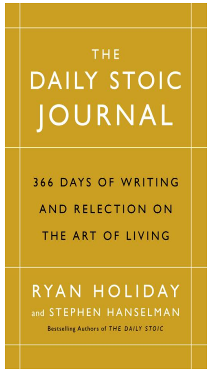
BM25 leans on exact or close keyword matches and would look for books that contain terms like "Marcus“, "Aurelius," "writings," or "similar" in their titles or descriptions.

- ✓ “Marcus”
- ✓ “Aurelius”
- ✓ “writings”



- ✗ Letters from a Stoic
- ✗ Seneca
- ✗ other Stoic Writings





BM25 Example - Ambiguity

A user is interested in fruit import and searches for "Changes in apple stock in USA."

BM25 leans on word matching and likely to return results regarding the stock of the "Apple" company

VS

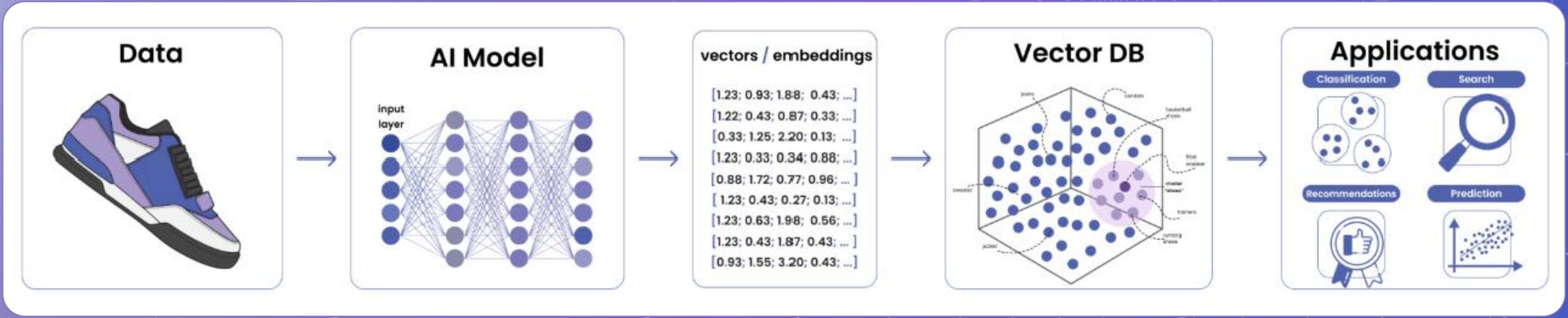
Vector Search

Adding Semantic Understanding into the Equation

Vector Search

→ Representing unstructured data such as text, images, videos as vectors / embeddings

→ Running Search Algorithms such as k-Nearest Neighbor (k-NN) and ANN to find the most similar items



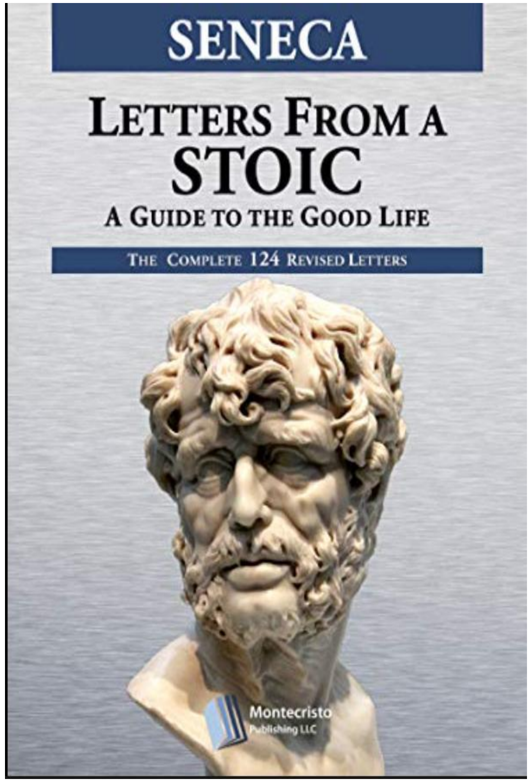
The Strengths of Vector Search

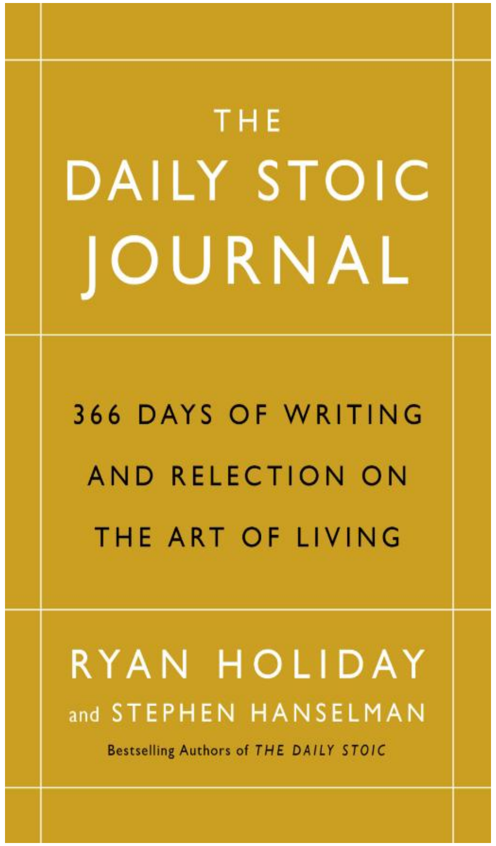
- **Captures Semantics**: Captures the context of words or content, offering results that might not have the exact keywords but are contextually relevant.

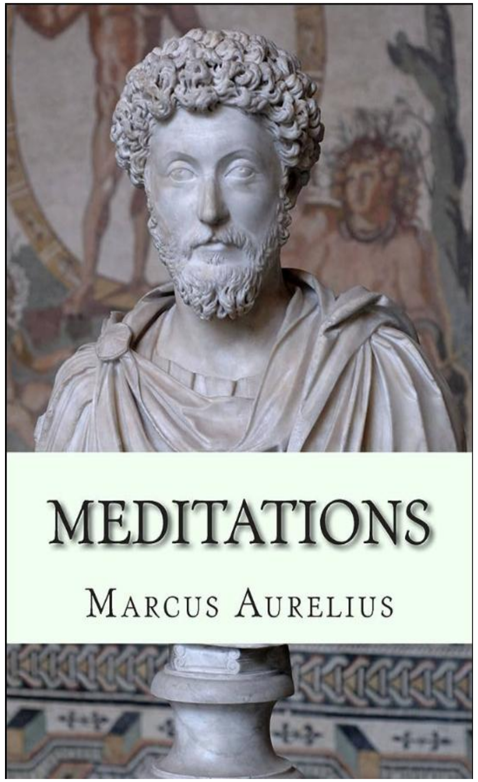
- **Versatile**: Can work with various kinds of data, including text, images, and audio.

- **Easily Personalized**: Can be combined with user behavior vectors to provide personalized search results.

Semantic Understanding for Marcus Aurelius







The Challenges with Vector Search

The Challenges with Vector Search

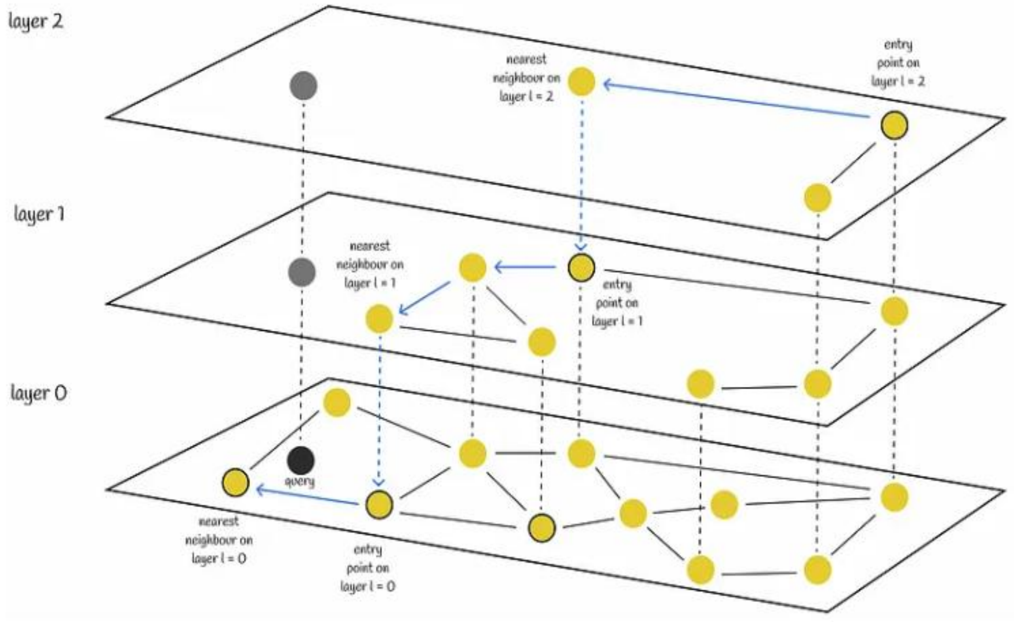
- **Explainability**: The matched results of vector search can be challenging to interpret.
- **Embeddings generation:** LLMs training and inference require expertise and resources
- **Hallucinations** – always return the top K results, even if not so relevant
- **Scalability:** Vector search algorithms are compute-intensive, therefore running large-scale search in real-time may be quite challenging in terms of performance, accuracy and cost.

Tackling The Scale Problem - Approximation

ANN (Approximate Nearest Neighbor) methods to trade small reduction in accuracy for a significant improvement in latency.

One common method is HNSW:

- Start from the highest (most coarse) layer and progresses one layer at a time
- The local nearest neighbors are greedily found among each layer nodes.
- Ultimately, the detected nearest neighbor on the lowest layer are the query result.

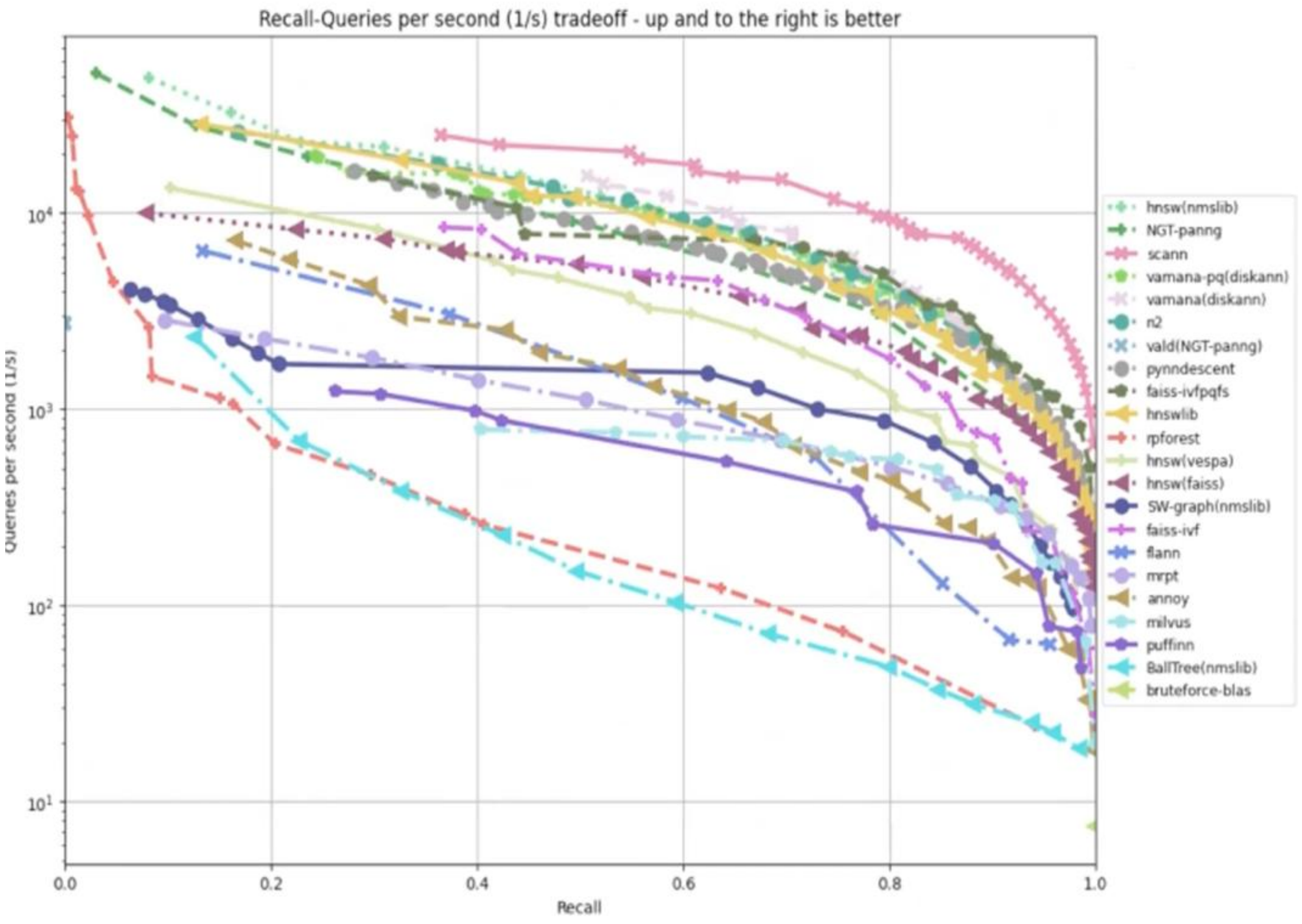


While ANN methods may find, or be very close to, the exact true nearest neighbors, there's no such guarantee. In applications **where exact matches are critical, ANN might fall short**.

Managing the tradeoff: Speed vs Accuracy

Using **sub-optimal approximation methods** to achieve real-time latencies with high QPS

ANN algorithms balance between performance (queries per second) and accuracy (recall)



Source: Ann-benchmarks.com

Taking search to the next level: Hybrid Search

Hybrid Search

Combining Classic Search + Vector Search

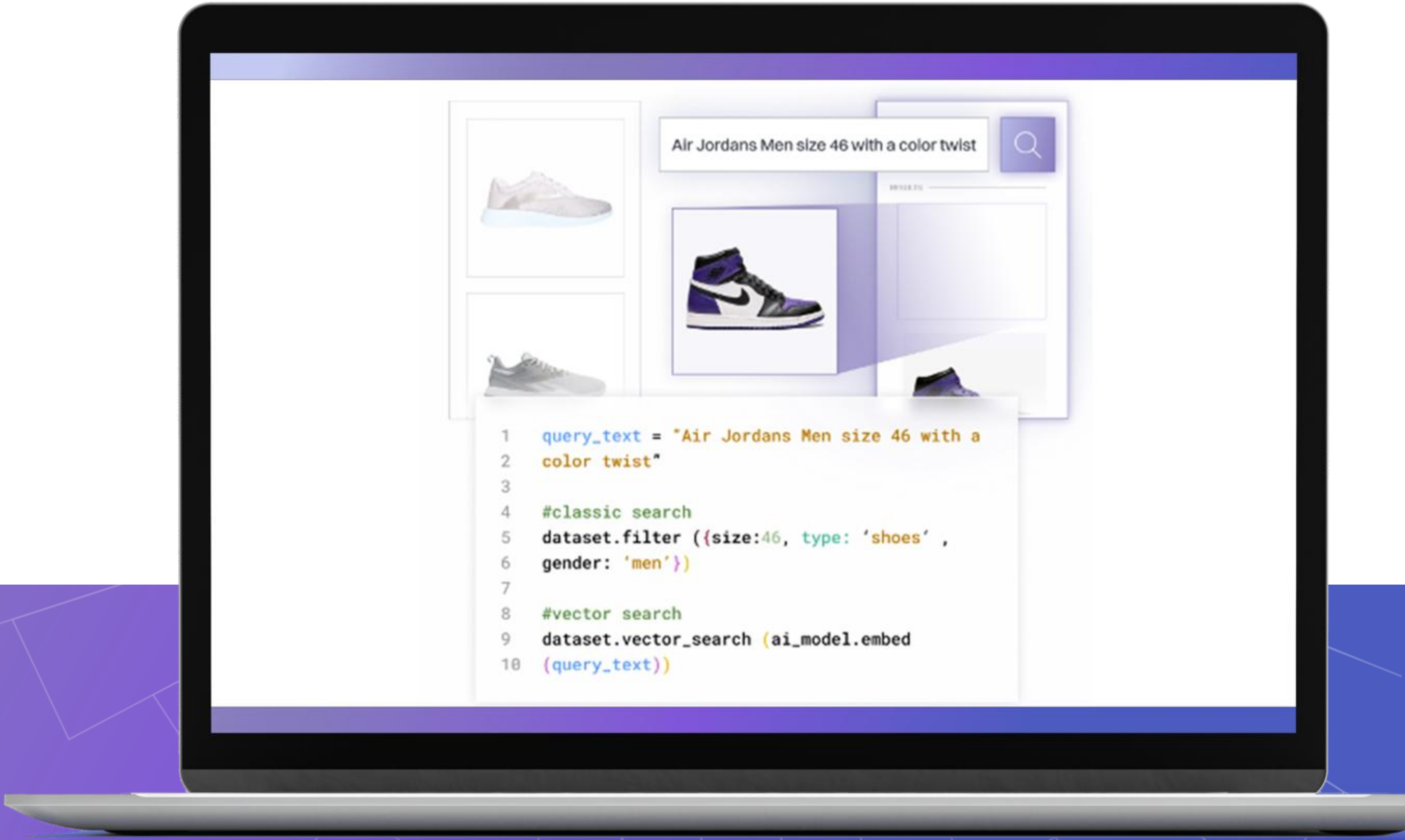
Search like in ChatGPT

Neural Search

- Semantic / Multi-modal
- K-NN / ANN

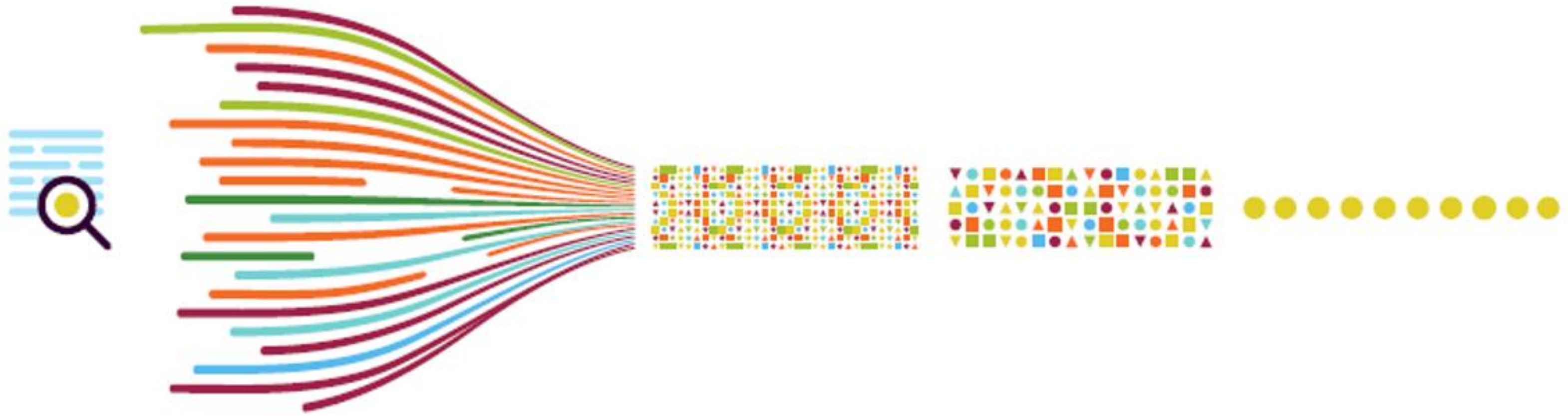
Lexical search

- Metadata Filters
- TF-IDF/BM25
- Aggregations



Business Logic with Metadata Filters

- **Space reduction**: filters can narrow down the vast amount of data to a more focused subset that aligns with specific criteria. This 'candidate generation' phase, ensures that the vector search operates on a more manageable and relevant set of data.
- **Rule based logic**: Explicit filters can ensure a specific policy and handle common cases of ambiguity by narrowing the search to relevant categories, locations, etc.
- **Cost and Performance**: Reducing the load on the search/ranking phase drastically saves latency and costs.



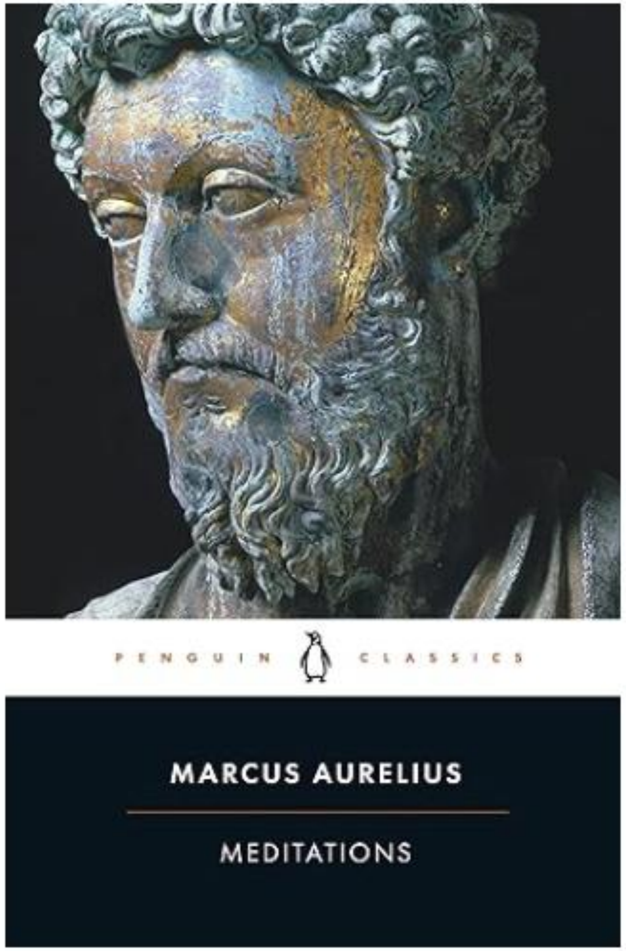
Hybrid Data Structure

Metadata - Structured

- Category
- Sub-category
- Brand
- Price

Embeddings - Unstructured

- Image
- Description

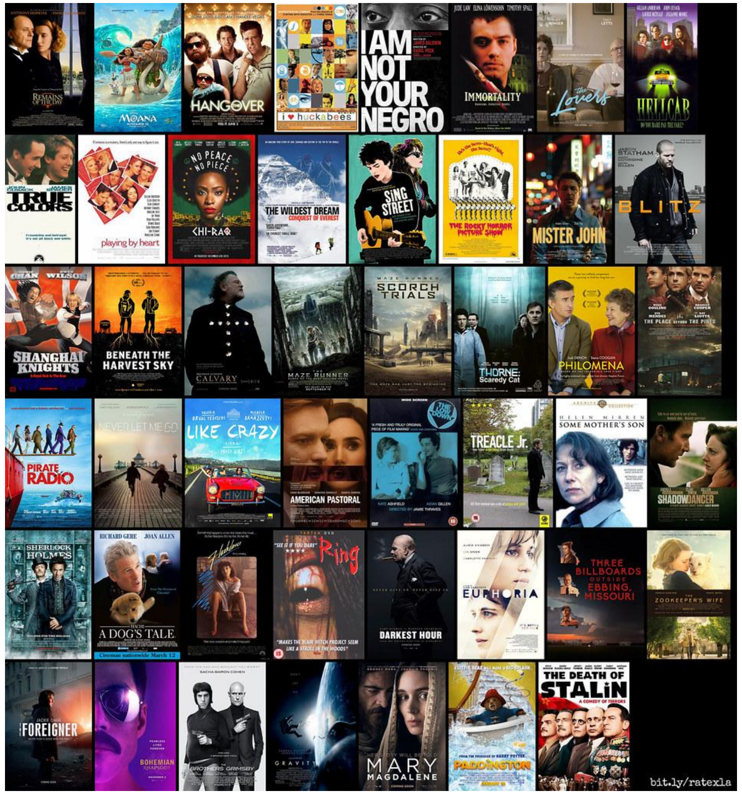


Example - Movie Recommendation System

Consider a recommendation system, based on the MovieLens movies database

Let’s compare two methods:

1. Vector search only - based on word embedding of the description
2. Hybrid search - using metadata filtering



Example - Movie Recommendation System

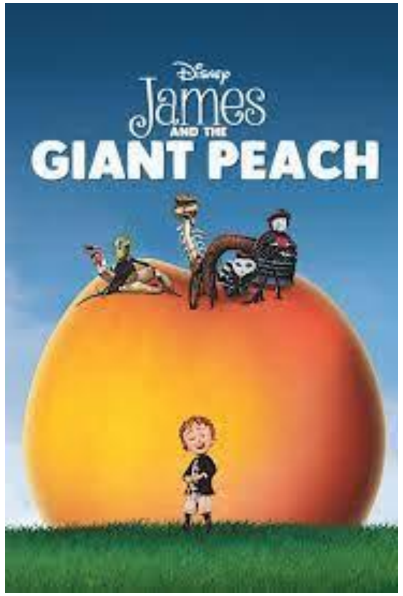
Vector Search

Title: 'James and the Giant Peach'

Description: 'young orphan boy James's spill magic bag crocodile tongue find possession giant peach fly away strange land Adventures big grow tree')

Rank	ID	Title
1	555	James and the Giant Peach
2	36676	Air Mater
3	20863	A Kid for Two Farthings
4	40550	The Famous Box Trick
5	15758	Mystics in Bali
6	23548	Magic Boy
7	9331	Freedomland
8	454	Sleepless in Seattle
9	19950	Gamera vs. Viras
10	16095	Magic Christmas Tree

Not children movies!



Example - Movie Recommendation System

Hybrid Search

Title: 'James and the Giant Peach'

Description: 'young orphan boy James's spill magic bag crocodile tongue find possession giant peach fly away strange land Adventures big grow tree')

```
if match('genres') and not match('title'):
    return true
return false
```

```
                    = {
  'params':                    ,
   "query": {"boost": 1},
  '                        ':{"boost": 10}
}
```

Rank	ID	Title
1	37206	Moana
2	27368	The Good Dinosaur
3	11584	The Tale of Despereaux
4	14703	Mars Needs Moms
5	36753	Trolls
6	30255	Phantom Boy
7	36977	Little Longnose
8	3374	A Monkey's Tale
9	6256	Clifford's Really Big Movie
10	6677	The Chipmunk Adventure

Results are now relevant!

Fraud Prevention Use Case

Opportunity:

- eCommerce companies are estimated to lose $48 billion to fraud each year
- Using Similarity Search as an anomaly detection method in identifying fraudulent credit card transactions

Fraud Prevention Use Case

A new online transaction

- Country of origin
- Shipping address
- Email address
- Purchase amount
- Product
- IP

Similarity Search

Is it a known fraudster?
Is he similar to fraudsters?

Or to legit customers?

Fraud Prevention: Applying Hybrid Search

One of the biggest challenges of fraud detection is to identify and respond to a new fraud methods

Challenges:

- Rigidity: Relies on predefined rules, which means it can miss novel fraud tactics.
- High False Positives: Might flag legitimate transactions that simply match the predefined criteria.
- Latency sensitive: require real-time response (<100ms)

Classic method - use rule-based logic that is similar to known fraud patterns.

```
def                                    (Q, V):
  score = 0
  if (V['                  '] == 'Food' or
     V['                  '] == 'Drinks' or
     V['                  '] == 'Consumables')\
       and V['                  '] == "Alcohol":
     if V['Type'] == 'Whiskey' and V['price'] > 500 * USD:
       score +=                  ('brand')
  return score
```

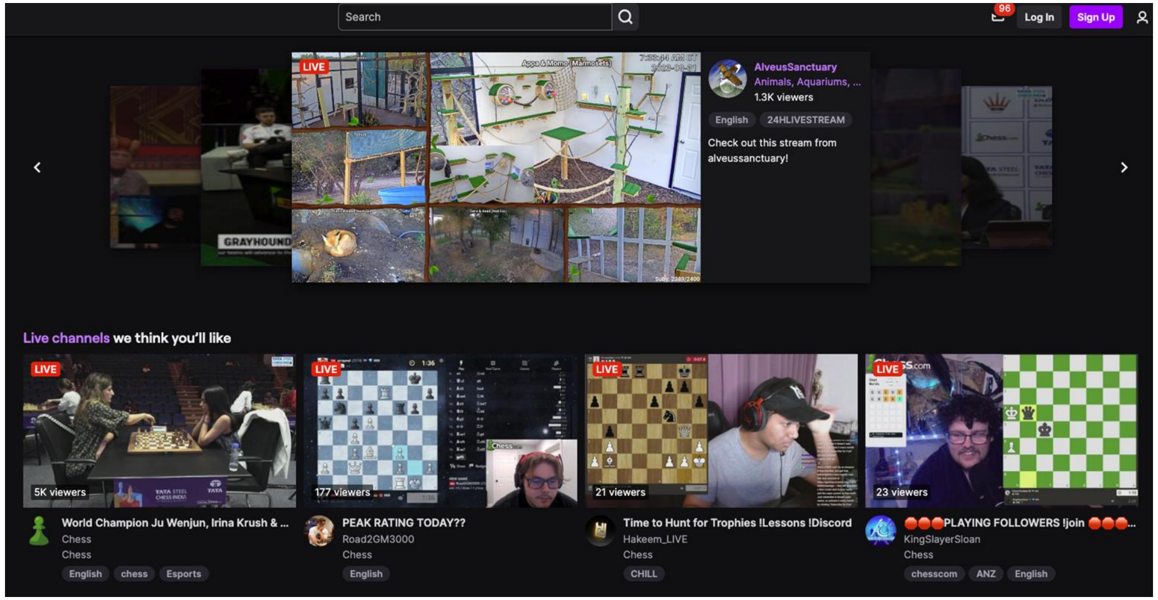
Hybrid method - combine classic search with vector search for fast and efficient scan of similar products

```
model.transform('Glenlivet whiskey aged 15 years price over 300 USD
Shipped New-Mexico')
```

Twitch Use Case

Using Hybrid Search to efficiently find live broadcasters

- “Is Live” pre-filter: Reduces to 150-250k (95% of use cases)
- At this point, KNN proves just as fast, eliminating the need for added indexing overhead
- Specific filters example - "Streams Fortnite in Spanish"

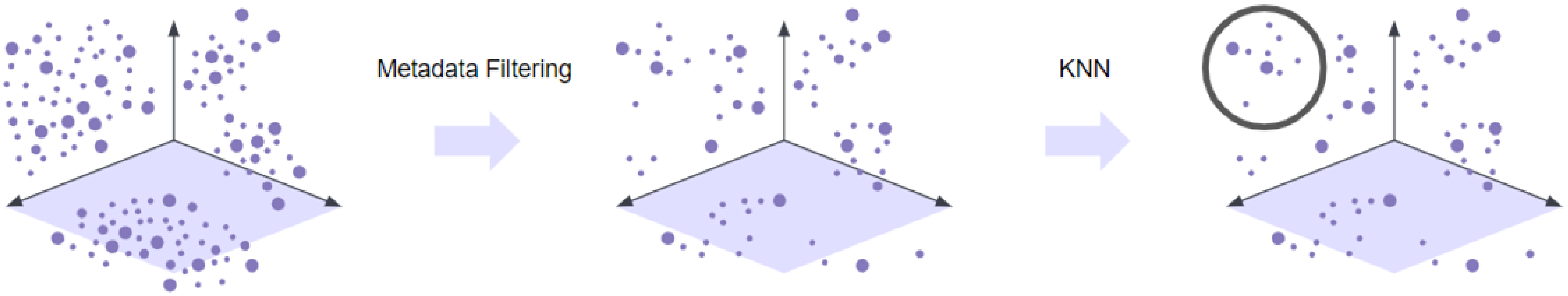


- Average concurrent Twitch viewership is 2.45 million users
- Total Broadcasters: 5-10 million

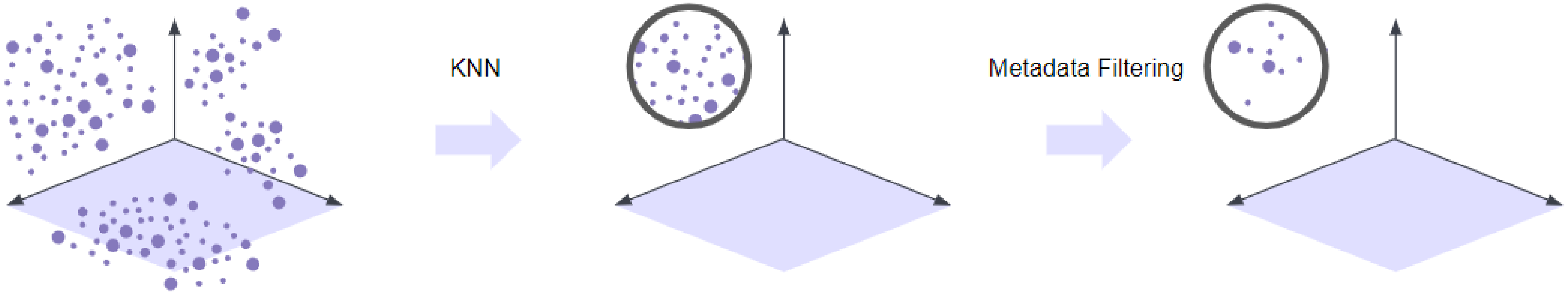
Hybrid Considerations

Pre vs. Post Filtering

Pre-filtering – Filters applied first (e.g. by country) and k-NN is running over the remaining vector space.

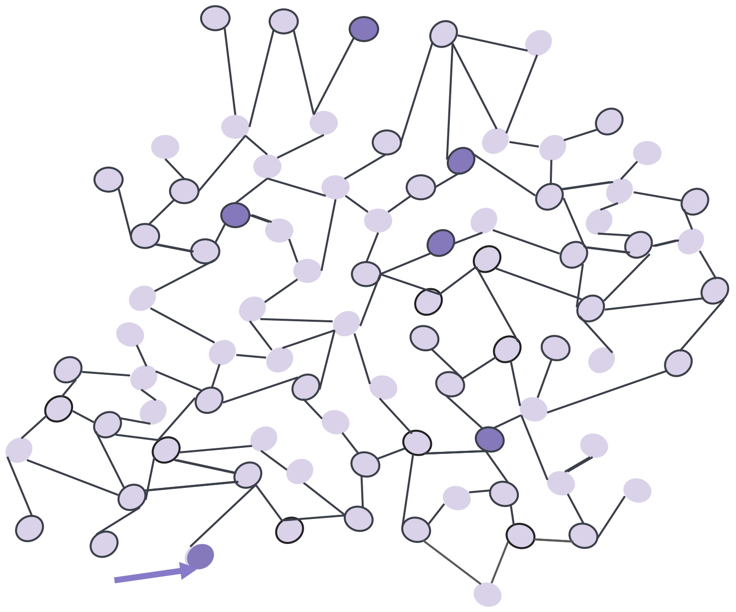


Post-filtering - Vector search is performed over the whole space (e.g. HNSW), and the filters applied over the top K results.



HNSW filtering while traversing the graph

- Layered graph with accumulation of nearest neighbors at each step
- Filters are applied to the candidates while traversing the graph
- Latency sensitive as it might take longer time to achieve top K that match the criteria



First matched candidate

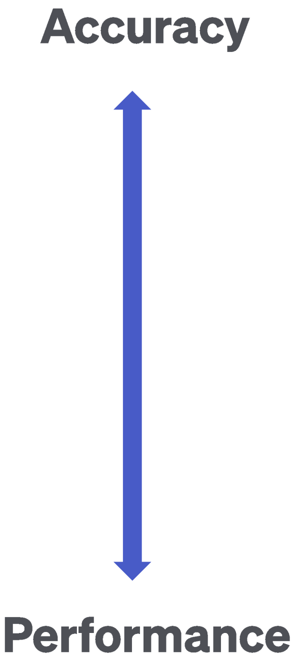
Relevant Results

Passed filter

Filtered out

Metadata filtering

1. Pre filters ➔ k-NN (brute force)
2. HNSW filtering while traversing the graph
3. HNSW ➔ Post filters

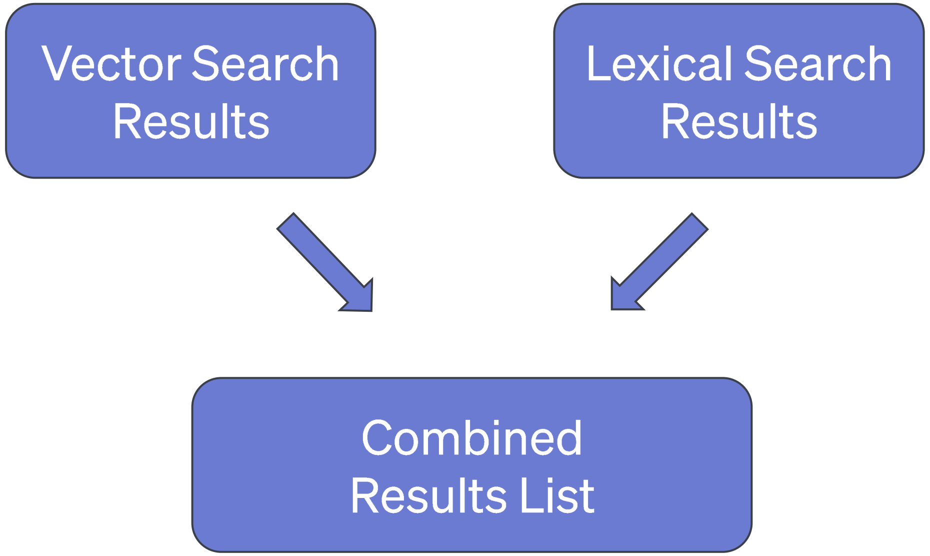


Hybrid Ranking - Combining the Scores

When combining approaches to obtain top K results, we need to merge the results to a single top-K list.

The common approaches

1. **Rank Based** - combine the results based on the ranking per method while ignoring the scores
2. **Score Based** - weight the scores to a single score, rank results based on this unified score



Rank Based Approach - Reciprocal Rank Fusion

- A method to combine ranking of multiple search types, by combining the inverse ranking of each search method.
- Ignores the individual scoring, only using the rank, thus removing the need for scaling of scores.
- For hybrid search, it can be written as:

This method makes sense when the different method scores are on different scale and can't be properly normalized

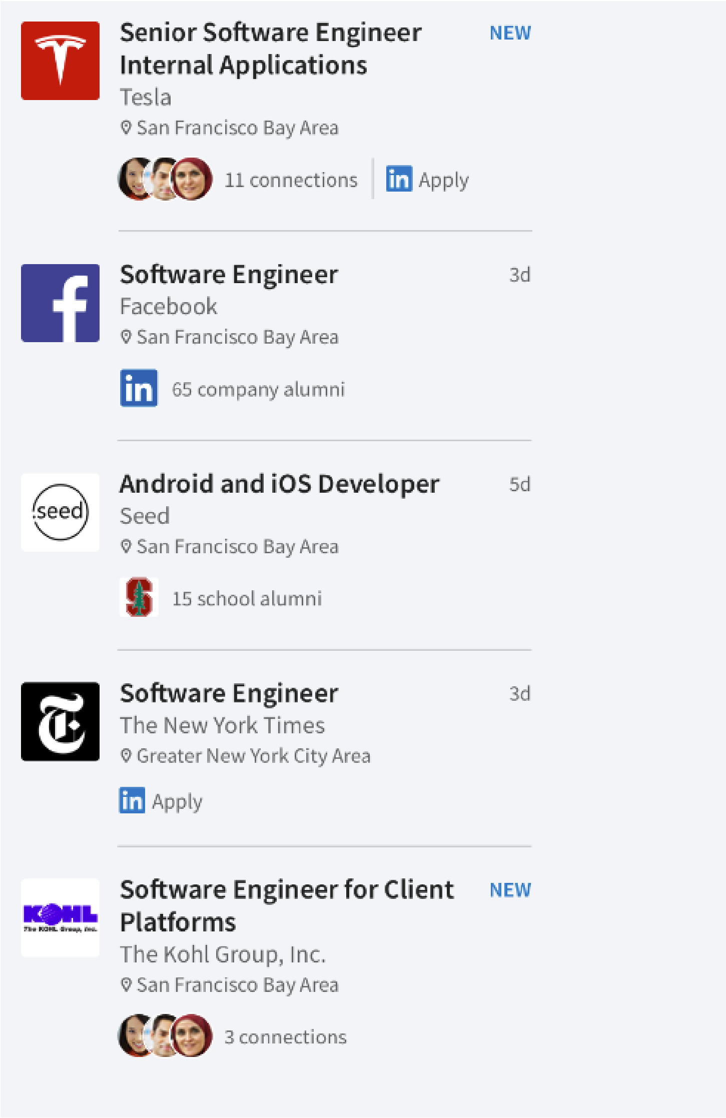
$$\text{Ranking score} = \frac{1}{r(classic)} + \frac{1}{r(vector)}$$

Reciprocal Rank Fusion Example

If a document was ranked **6** (lexical search) and **2** (vector search).

The combined ranking score is then **1/6 + 1/2 = 0.666**

- On LinkedIn, the hybrid score might represent simple keyword-matching search – “Software Engineer” ,”Full Stack”, ”Node JS”
- The vector search can represent a more advanced, contextual understanding of the job listing's content - maybe the candidate was interested in electric car or social media companies



Score Based Ranking

There are two main approaches to this ranking method

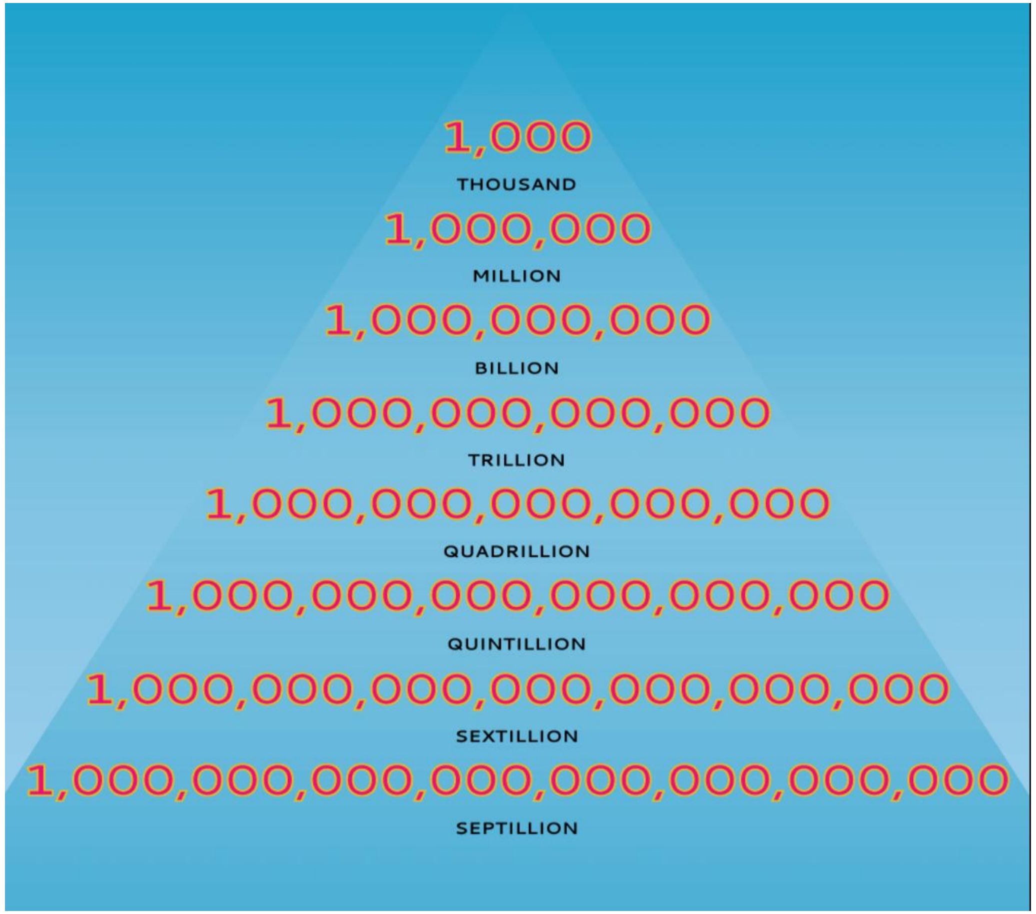
- **Rule based approach** - use weights and linear combinations of scores
  in this cases, all scores should be first normalized to the same scale
- **ML based approach** (i.e. learning to rank) - use an ML model with the different scores as features, will not be discussed here).

$$Score = \quad * \quad + \quad *$$

OMG, how do we get it into production?

The Billion Scale Challenge

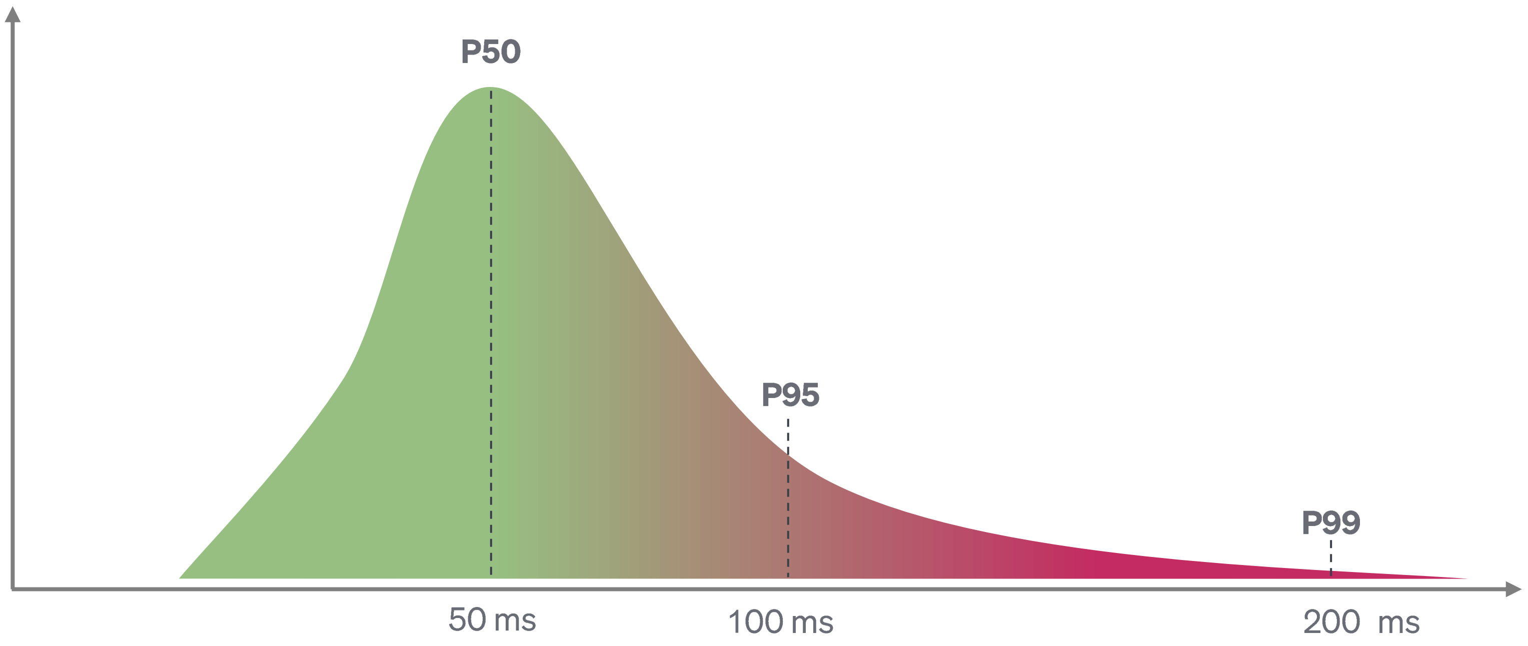
- Considering the amount of data coming via **streams or batch processing** these days, one key element of vector databases is to achieve billion-scale.
- However, **indexing** billions of vectors in a **cost effective** while achieving **low latency** is incredibly difficult task.
- Breaking this barrier requires a technological step function with a **purpose-built solution**.



Real-time at scale

User experience dictates **real-time search to be < 200ms**

Any additional latency significantly **impact the user engagement and conversion**



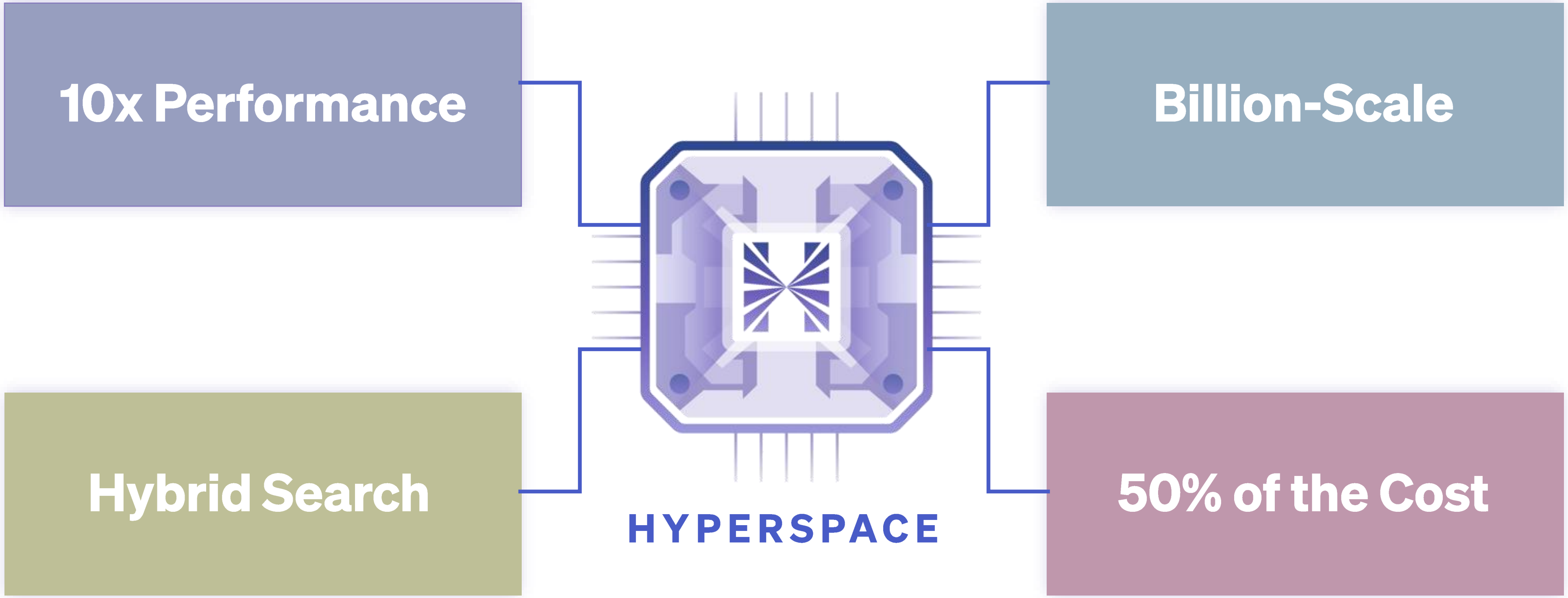
[Every 100ms in Added Page Load Time Cost 1% in Revenue]

[Increasing web search latency by 100ms reduces the daily number of searches per user by 0.2%]

[An increase of 30% in latency costs about 0.5% in conversion rates]

Designing a Search Processing Unit (SPU®)

Leverage domain specific computing to deliver unmatched search performance in data-intensive applications, surpassing the limitations of traditional software-based solutions.



Summary

- The importance of relevancy
- An overview of classic search and vector search: strengths and weaknesses
- Taking search to the next level with Hybrid Search
- Best practices of combining lexical and neural search to improve relevancy
- Achieving Real-time at Scale

HYPERSPACE

Let’s Connect

https://www.linkedin.com/in/ohad-levi/

ohadl@hyper-space.io