Semantic Search with LLMs

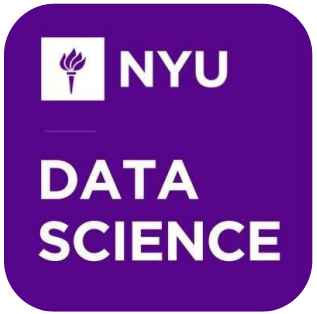
Shaan Khosla

Senior Data Scientist and Researcher

Pearson

My name is **Shaan Khosla**

- Senior Data Scientist at Nebula
- Various research with LLMs, topic models, and legal NLP
- MS in Data Science, New York University
- Writer of *Let's Talk Text* newsletter



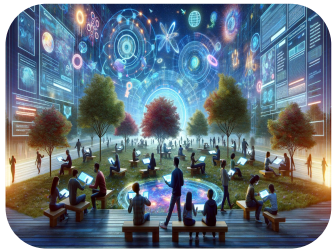
Optimizing LLMs

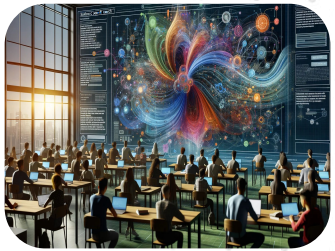
Code:
github.com/shaankhosla/semanticsearch

Follow-up with me:
linkedin.com/in/shaan-khosla

shaankhosla.substack.com **for email newsletter**

shaankhosla.com



Pearson

Poll

What is your current level of experience with semantic search?

- What is semantic search?
- I've used `cos_sim = dot(a, b)/(norm(a)*norm(b))`
- I regularly use semantic search algorithms in projects and work
- I'm up to date with the latest developments

Pearson

Poll

What part of the talk are you most looking forward to?

- Hands-on coding walkthroughs
- Exploring vector databases
- Exploring search algorithms
- Optimization tricks
- More pictures of Mila

Pearson

Schedule

1. Introduction to Large Language Models (45 mins)
2. Q&A (5 mins) & Break (5 mins)
3. How Semantic Search Works (45 mins)
4. Break (10 mins)
5. Optimize Semantic Search Accuracy, Speed, and Cost (60 mins)
6. Summary and Q&A (10 mins)

Pearson

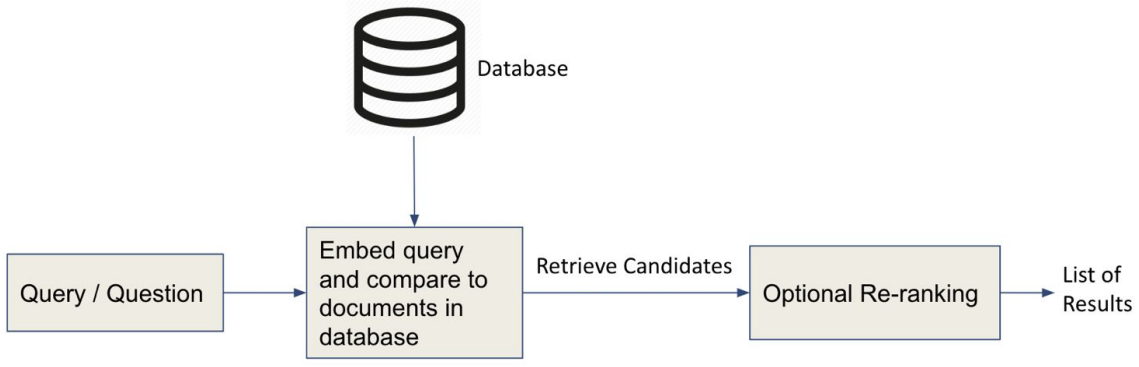
Schedule

1. **Introduction to Large Language Models (45 mins)**
2. Q&A (5 mins) & Break (5 mins)
3. How Semantic Search Works (45 mins)
4. Break (10 mins)
5. Optimize Semantic Search Accuracy, Speed, and Cost (60 mins)
6. Summary and Q&A (10 mins)

Pearson

Find items in a large collection

- Search relies on similarity calculations
- User provides a “query”:
  - Search string
  - Example document
  - Latent representation
  - GPT generated search term
- System returns a list of matching “documents” from the database



Practical applications

Examples

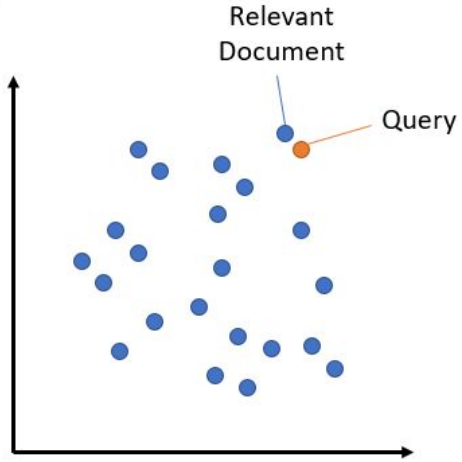
- Text search
  - Search string <> The web
- Recommender system
  - User representation <> Item representation
- Reverse image search
  - Photo <> Library
- Copyright detection
  - Audio <> Collection of songs

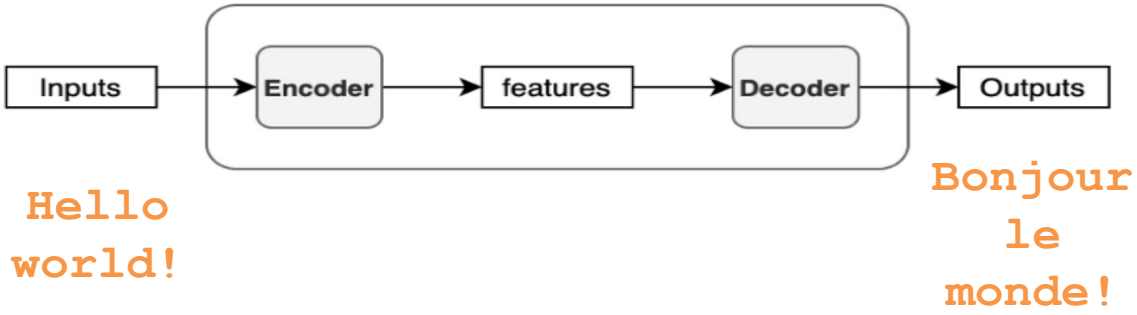
Problems of scale

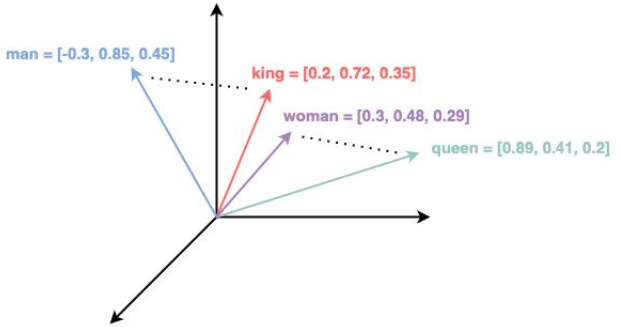
- Size of collection
- Dimensionality of representation
- Complexity of computing similarity
- Storage of representation and metadata
- *Can we do better than o(N) brute force search?*

Overview of text embeddings

- Manual feature encoding
- Word counts: Bag of Words
- Weighted word counts: TF-IDF, LDA
- Neural networks: Word2Vec, GloVe
- Context: RNNs, LSTMs, GRUs
- Attention: BERT, all-MiniLM-L6-v2



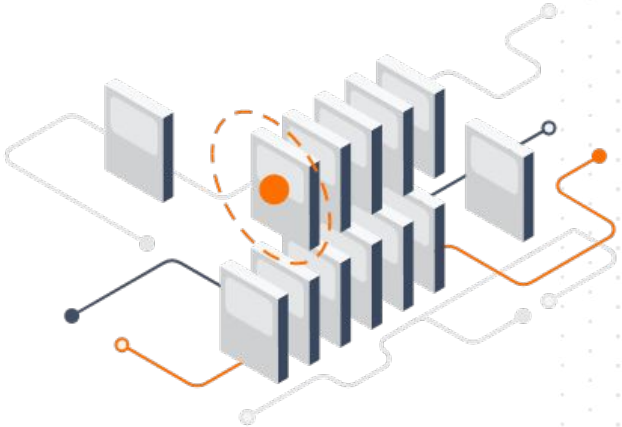




Pearson

Key libraries

- 🤗 **Transformers**
  - Many, many open-source LLMs ready to go
  - Task-ready code that works easily with pre-trained models
  - Easy inference
  - Automatic optimizations
- **Sentence Transformers**
  - Easy text and image embeddings
  - Automatic optimizations
  - Works easily with HuggingFace models



Text embedding models

- Pre-trained open source models
- Domain-specific models
- Proprietary embedding models: OpenAI's *text-embedding-ada-002*

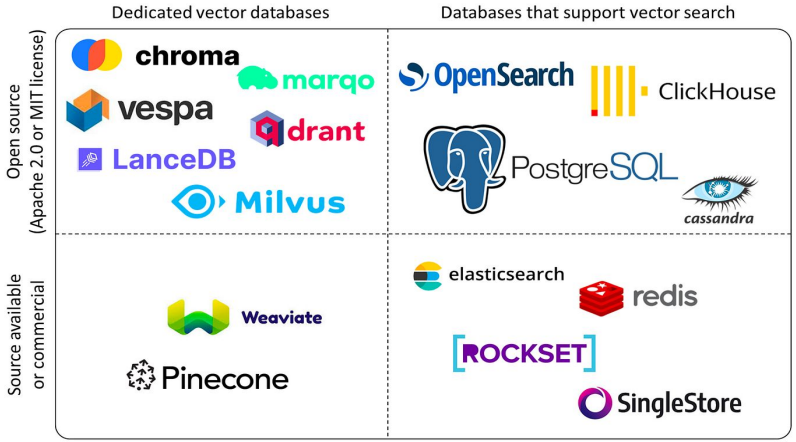
Hands-on code demo:
Semantic_Search_Basics.ipynb

Model Name	Performance Sentence Embeddings (14 Datasets)	Performance Semantic Search (6 Datasets)	Avg. Performance	Speed	Model Size
all-mpnet-base-v2	69.57	57.02	63.30	2800	420 MB
multi-qa-mpnet-base-dot-v1	66.76	57.60	62.18	2800	420 MB
all-distilroberta-v1	68.73	50.94	59.84	4000	290 MB
all-MiniLM-L12-v2	68.70	50.82	59.76	7500	120 MB
multi-qa-distilbert-cos-v1	65.98	52.83	59.41	4000	250 MB
all-MiniLM-L6-v2	68.06	49.54	58.80	14200	80 MB
multi-qa-MiniLM-L6-cos-v1	64.33	51.83	58.08	14200	80 MB
paraphrase-multilingual-mpnet-base-v2	65.83	41.68	53.75	2500	970 MB
paraphrase-albert-small-v2	64.46	40.04	52.25	5000	43 MB
paraphrase-multilingual-MiniLM-L12-v2	64.25	39.19	51.72	7500	420 MB
paraphrase-MiniLM-L3-v2	62.29	39.19	50.74	19000	61 MB
distiluse-base-multilingual-cased-v1	61.30	29.87	45.59	4000	480 MB
distiluse-base-multilingual-cased-v2	60.18	27.35	43.77	4000	480 MB

Source: Sentence Transformers

Vector databases

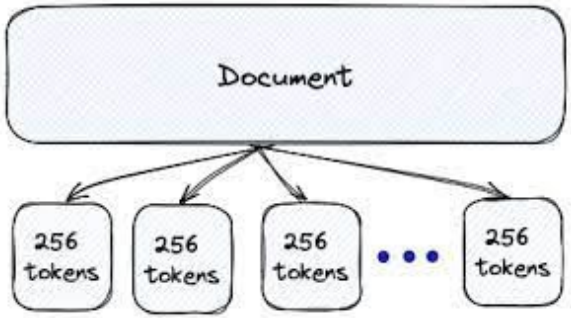
- Elasticsearch
- Pinecone
- Qdrant
- Redis
- Milvus
- Chroma
- Weaviate
- Deep Lake
- LanceDB
- Vespa
- Vald
- Pgvector
- Faiss
- OpenSearch
- Cassandra
- MongoDB Atlas Vector Search



Source: Pinecone

Document chunking

- Challenge: Cannot embed large documents as a single vector
  - Books or research papers
- Solution: Divide the large document into smaller manageable chunks
  - Naive chunking
  - Max token window chunking
    - Challenge: May cut off important text in between chunks, losing context
    - Solution: Use overlapping window
  - Priority splitting with custom delimiters: *"**\n\n**", "**\n**", ".", "**# Title**"*



Hands-on code demo:
Document_Chunks.ipynb

Source: Pinecone

Schedule

1. Introduction to Large Language Models (45 mins)
2. **Q&A (5 mins) & Break (5 mins)**
3. How Semantic Search Works (45 mins)
4. Break (10 mins)
5. Optimize Semantic Search Accuracy, Speed, and Cost (60 mins)
6. Summary and Q&A (10 mins)

Pearson

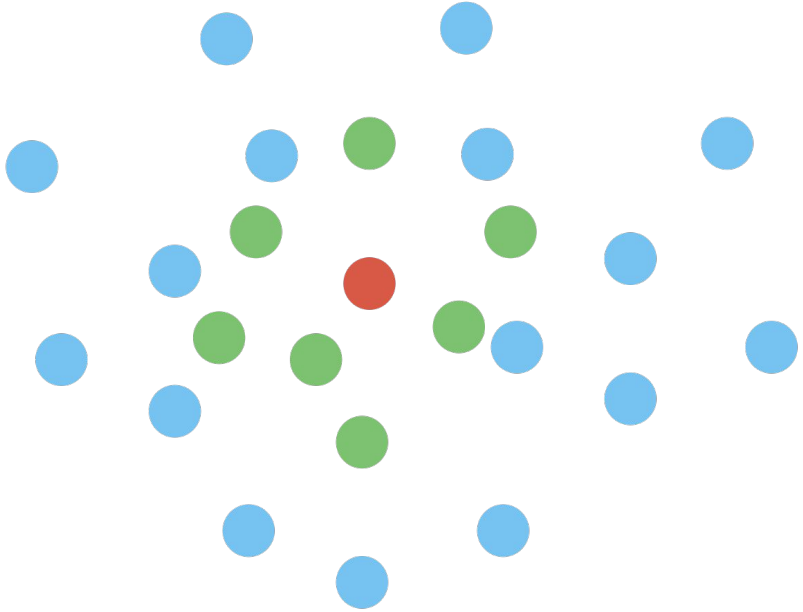
Schedule

1. Introduction to Large Language Models (45 mins)
2. Q&A (5 mins) & Break (5 mins)
3. **How Semantic Search Works (45 mins)**
4. Break (10 mins)
5. Optimize Semantic Search Accuracy, Speed, and Cost (60 mins)
6. Summary and Q&A (10 mins)

Pearson

Approximate Nearest Neighbors (ANN)

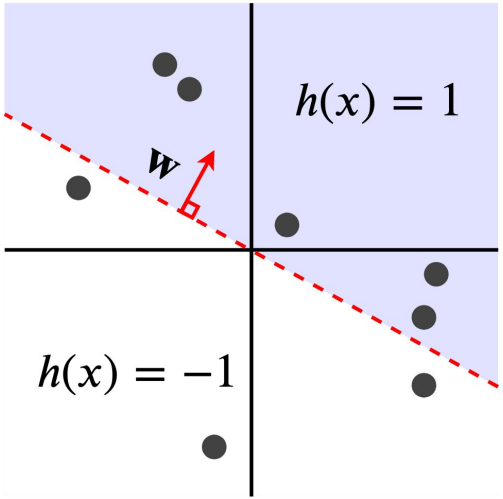
- Search through collection of vectors to retrieve similar objects
- Brute force: check all possible combinations
- Goal: Develop a data structure that allows less than o(N) vector comparisons
- Use an algorithm to identify a subset of candidates

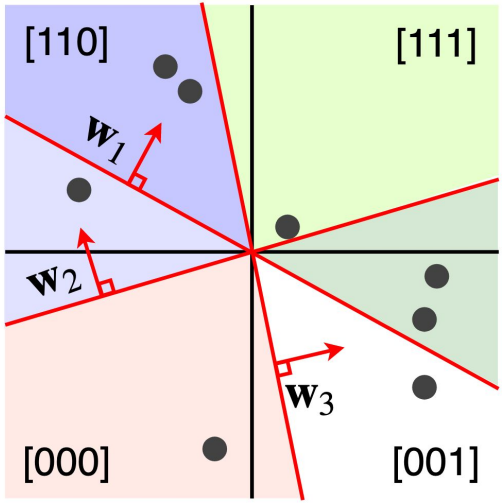


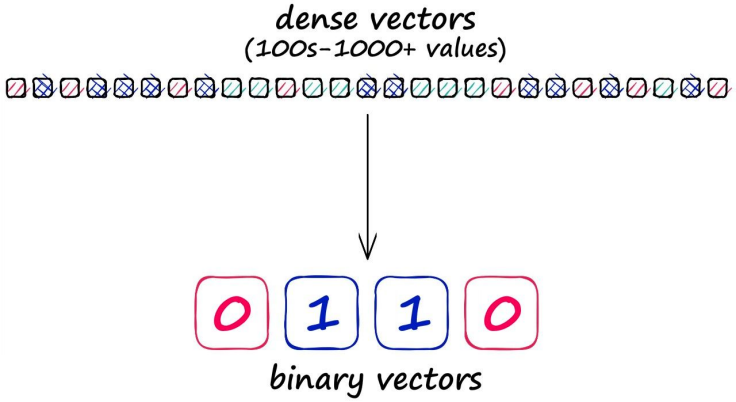
Locality Sensitive Hashing (LSH)

- One of the original search algorithms, used by Google image search
- Steps
  1. Hash all vectors in our collection
  2. Hash search vector
  3. Use vector hashes to identify subset of neighbors

Hands-on code demo:
LSH.ipynb







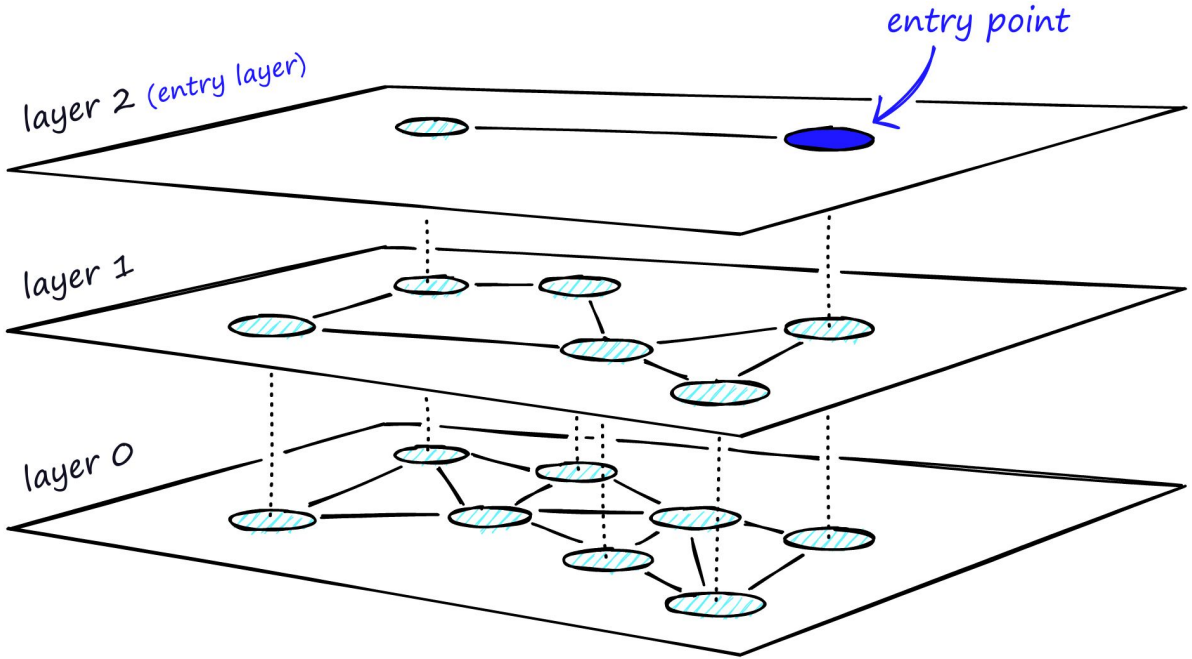
Source: Randorithms

Source: Pineceone

Pearson

Hierarchical Navigable Small Worlds (HNSW)

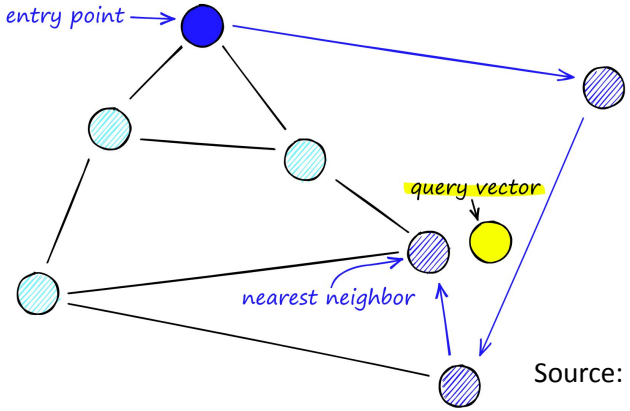
- Is a top-performing index
- Link similar vertices on a series of graphs
- Top layer has the longest edges, lowest has the shortest



Source: Pineceone

Hierarchical Navigable Small Worlds (HNSW) pt. 2

- Graph construction, adding a new node
  1. Start at top layer, find local minimum to *ef* neighbors using similarity
  2. Move down a layer and repeat
  3. At target insertion layer, choose nearest *M* vertices and create edges
- Search
  1. Start at top layer, move vertex to vertex until there are no closer vertices (local minimum)
  2. Move down a layer and repeat
  3. When at bottom layer, return the nearest vertices

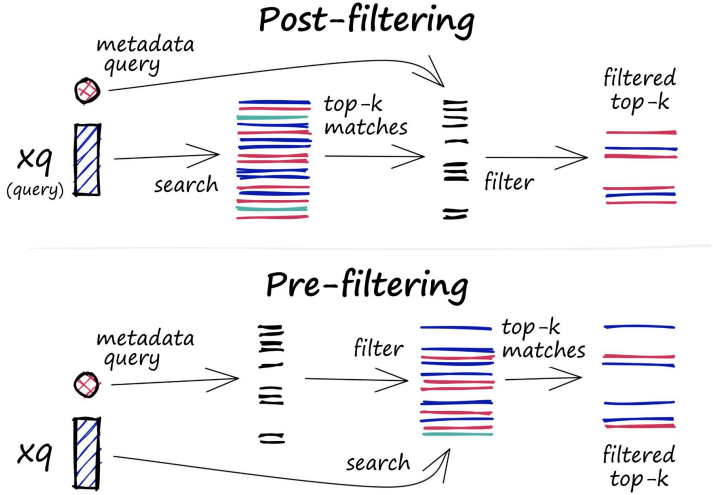


Source: Pineceone

Hands-on code demo:
HNSW Hyperparam Search.ipynb

Filtering

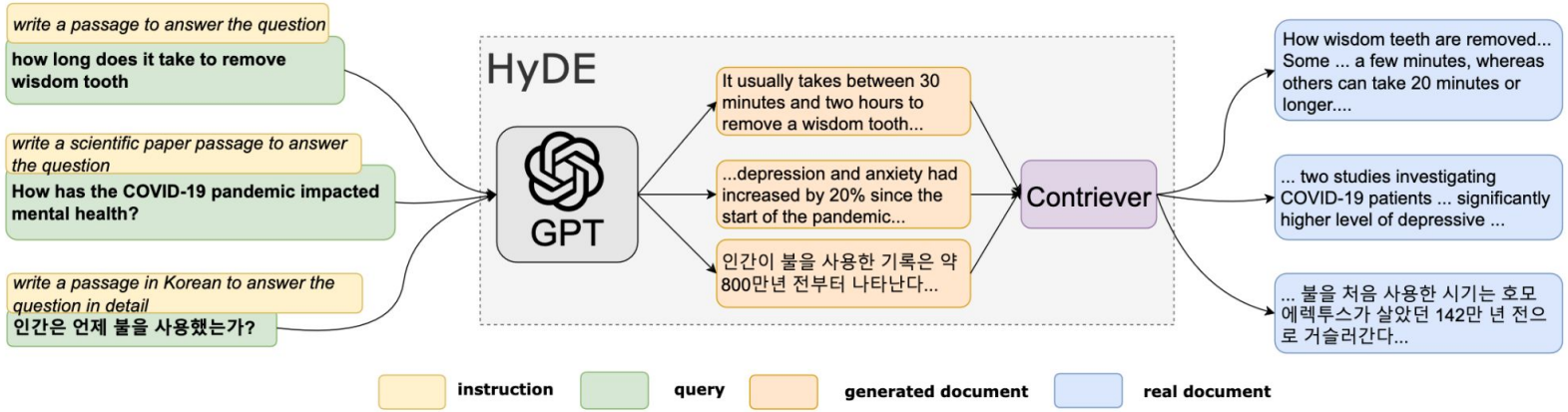
- *"Find all JPEGs with mountains in them"*
- **Pre-filter**
  1. Apply metadata filtering first, then brute force vector search over subset
  2. *"Vector search over only JPEGs"*
- **Post-filter**
  1. ANN vector search first, metadata filtering over remaining subset
  2. *"Select JPEGs from pictures that have mountains in them"*



Source: Pineceone

Asymmetric search

- Imbalance between semantic information in input query and searchable documents
- *"Court cases about land sales"* to search over case law documents
- Rely on learnings on LLMs rather than knowing exactly what needle to search for
  - Select the right model
- Hypothetical Document Embeddings (HYDE)
  - Generative model to construct hypothetical document

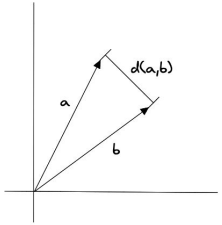


Source: Paper

Metrics

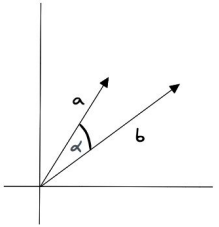
$$d(\mathbf{a}, \mathbf{b}) = \sqrt{(\mathbf{a_1} - \mathbf{b_1})^2 + (\mathbf{a_2} - \mathbf{b_2})^2 + ... + (\mathbf{a_n} - \mathbf{b_n})^2}$$

- **Euclidean distance**
  - Straightforward, literally the distance
  - Natural choice if embedding model not trained with a specific loss function
  - Sensitive to magnitudes



- **Cosine similarity**
  - Measures angle between vectors
  - Not affected by size of vector
  - Between -1 and 1
  - Very popular for semantic search

$$sim(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}|| \cdot ||\mathbf{b}||}$$



- **Dot product**
  - Identical to cosine similarity if vectors are normalized
  - Fastest

Hands-on code demo:
Cosine_vs_Dot.ipynb

Schedule

1. Introduction to Large Language Models (45 mins)
2. Q&A (5 mins) & Break (5 mins)
3. How Semantic Search Works (45 mins)
4. **Break (10 mins)**
5. Optimize Semantic Search Accuracy, Speed, and Cost (60 mins)
6. Summary and Q&A (10 mins)

Pearson

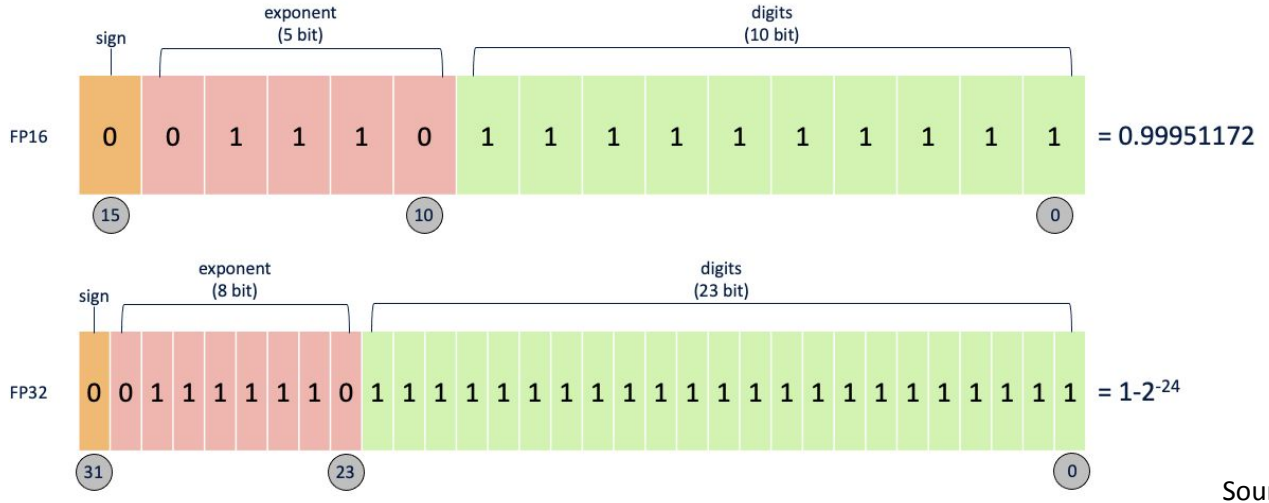
Schedule

1. Introduction to Large Language Models (45 mins)
2. Q&A (5 mins) & Break (5 mins)
3. How Semantic Search Works (45 mins)
4. Break (10 mins)
5. **Optimize Semantic Search Accuracy, Speed, and Cost (60 mins)**
6. Summary and Q&A (10 mins)

Pearson

Scalar quantization

- 1M dense vectors can require several GBs of memory
- Numbers are typically stored in 32 bits
- Instead, compress some values to FP16 or FP8
- Makes search faster and reduces memory footprint
- Increase in error is usually small



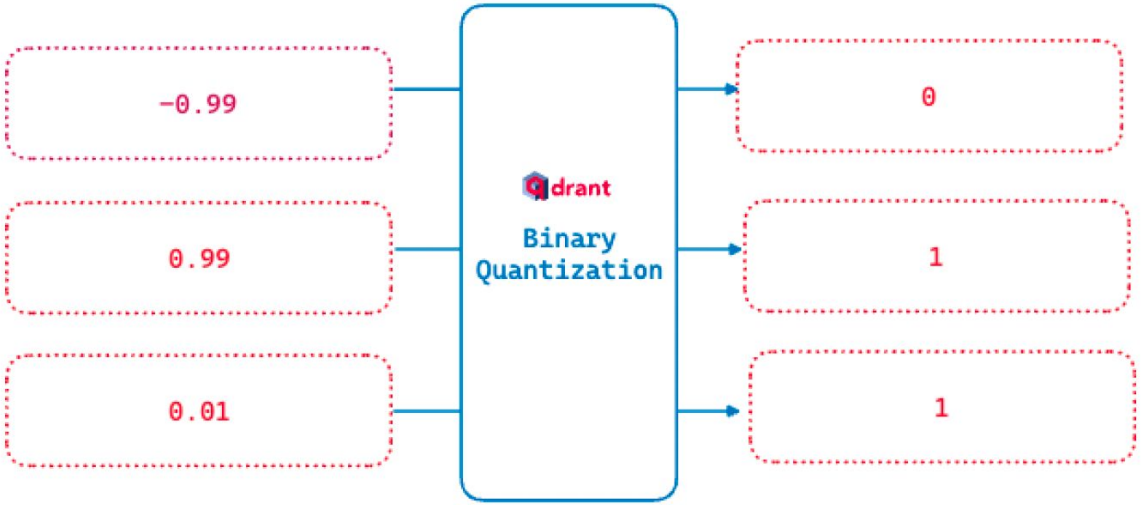
Source: Jonathan Davis

Pearson

Binary quantization

- Extreme version of scalar quantization
- Each number is represented as a single bit: either 0 or 1
- Boolean operations are very fast
- Can drastically decrease accuracy with small vectors

Hands-on code demo:
Scalar_and_Binary_Quantization.ipynb



Source: Qdrant

Embedding size

- Reduce dimensionality of data representations
- **Accuracy**: Larger embeddings often capture more details, improving search accuracy
- **Storage Cost**: Increased embedding size leads to higher storage requirements
- **Speed**: Larger embeddings can slow down search and retrieval processes
- **Memory Usage**: Higher memory demand for processing large embeddings
- **Trade-Off**: Balance between accuracy (larger embeddings) and efficiency (smaller embeddings)
- **Application Context**: Optimal size depends on specific use cases and resource constraints.

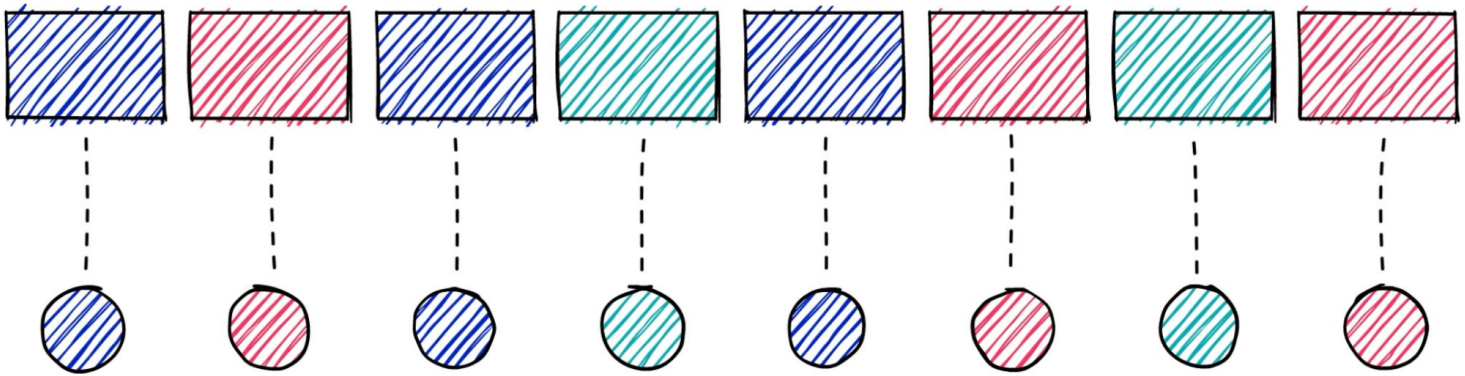
Hands-on code demo:
Embedding_Size.ipynb

Hands-on code demo:
Postgres Vector Length

Product quantization

1. Divide high-dimensional vectors into smaller chunks
2. Cluster all chunk 1's, cluster all chunk 2's, etc
3. Replace each chunk with index of nearest centroid in cluster
4. Store these indices instead of full vector, reducing storage size

Hands-on code demo:
Product_Quantization.ipynb



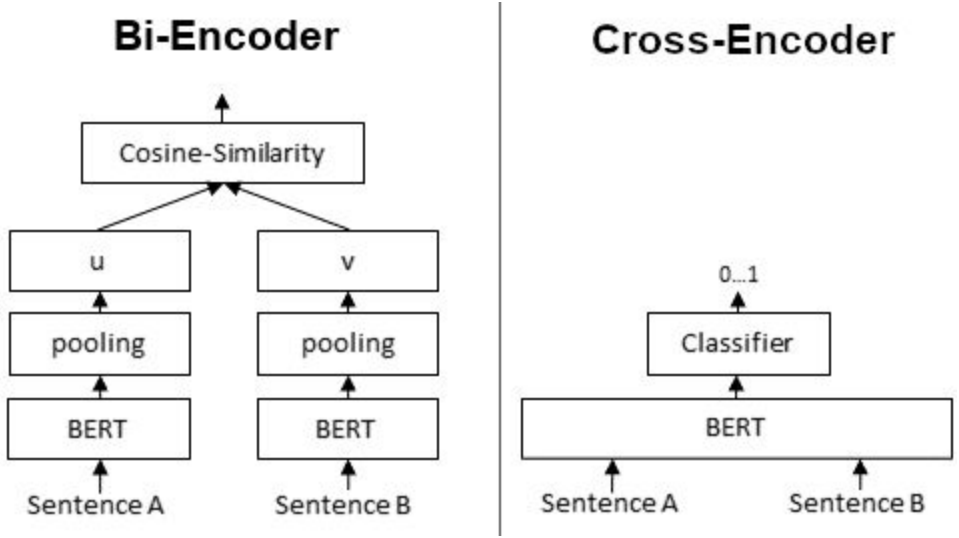
Bi-encoder vs Cross-encoder

Bi-encoder

- Text -> Embedding
- More compute efficient
- Not as accurate

Cross-encoder

- (Text, Text) -> Score
- Difficult to use at scale
- More accurate



Hands-on code demo:
CrossEncoder.ipynb

Source: Sentence Transformers

Retrieval Augmented Generation (RAG)

- Combine information retrieval with generative model
- Reduce hallucination
- Broader knowledge base
- Improve contextual understanding
- Incorporate up-to-date information
- Dynamic content generation

Hands-on code demo:
Mistral_7b_RAG.ipynb

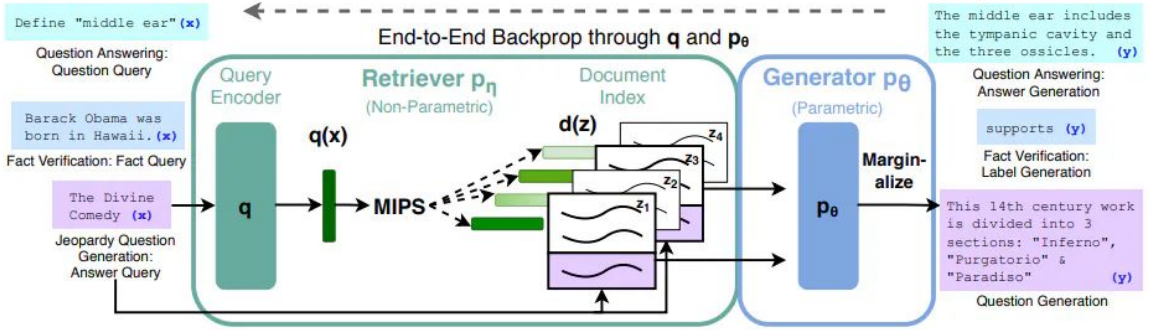


Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder + Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i. For final prediction y, we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

Source: Paper

Schedule

1. Introduction to Large Language Models (45 mins)
2. Q&A (5 mins) & Break (5 mins)
3. How Semantic Search Works (45 mins)
4. Break (10 mins)
5. Optimize Semantic Search Accuracy, Speed, and Cost (60 mins)
6. **Summary and Q&A (10 mins)**

Pearson

What’s next?

- Developing RAG models with improved accuracy speed
- Expanding data sources for retrieval: “embed anything”
- Integration with industry-specific needs
- Real time learning

Pearson