BEYOND THE BASICS: ADVANCED VECTOR INDEXING & STORES

HTTPS://GITHUB.COM/INSIGHTBUILDER

SELECTING BEST INDEX FOR YOUR APP

WHAT QUESTIONS WE WILL DISCUSS

1)WHY WE CANNOT USE TRADITIONAL DATABASES FOR VECTOR SIMILARITY SEARCH?

2)WHAT ARE THE BOTTLENECK IN VECTOR STORES? WHY THEY ARE PRESENT

3)WHAT TO LOOK FOR WHEN YOU SELECT A VECTOR INDEX / STORE/ CLIENT

WHO YOU ARE??

1) YOU ARE A DEVELOPER/ PROBLEM SOLVER NEW TO AI / LLM ARENA WHO RECENTLY FOUND ABOUT VECTOR STORES

2) EXPERIENCED SOFTWARE SYSTEM DESIGNER/ ARCHITECT / ADMINISTRATOR TASKED WITH NEW DESIGN AROUND VECTOR STORES

3) BUSINESS LEADER/ SENIOR MANAGER TRYING TO WRAP AROUND THE CONCEPT OF VECTOR STORES

EAGLE VIEW OF VECTOR STORES

- EVERY IDEA THAT SOLVES A EXPENSIVE PROBLEM HAS A BOTTLENECK.

  WHAT IS THE EXPENSIVE PROBLEM?

    FINETUNING LARGE LANGUAGE MODELS WITH CUSTOM / DYNAMIC DATA

- WHAT IS THE IDEA / REPLACEMENT FOR FINETUNING

    STORING THE DATA IN VECTOR STORES, AND USING NECESSARY CONTEXT

- WHAT IS THE BOTTLENECK?

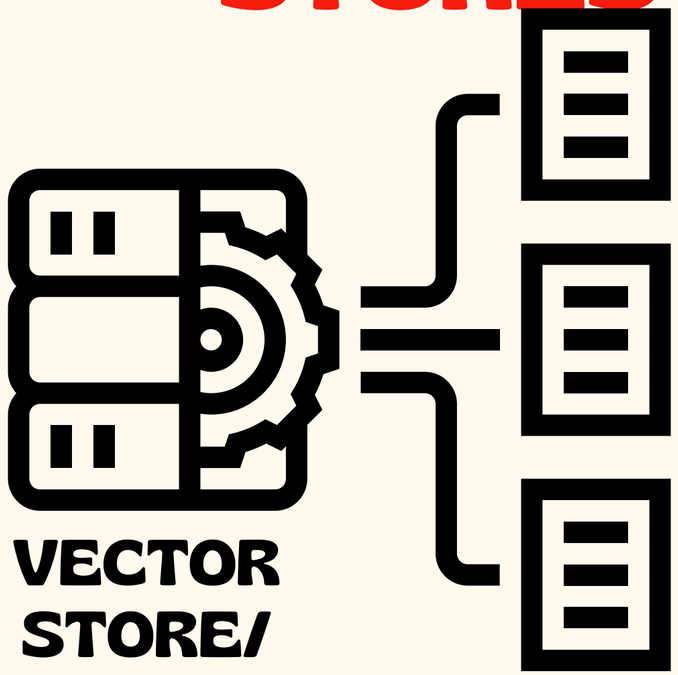
    SPEED OF DATA IMPORT, RETRIEVAL TIME ARE TWO BOTTLENECKS

ON TOP OF THESE YOU MIGHT WANT TO SUPPORT CRUD,

HIGH AVAILABILITY, HORIZONTAL SCALABILITY, CONCURRENT ACCESS, AND

SO ON....

PARTS OF VECTOR STORES



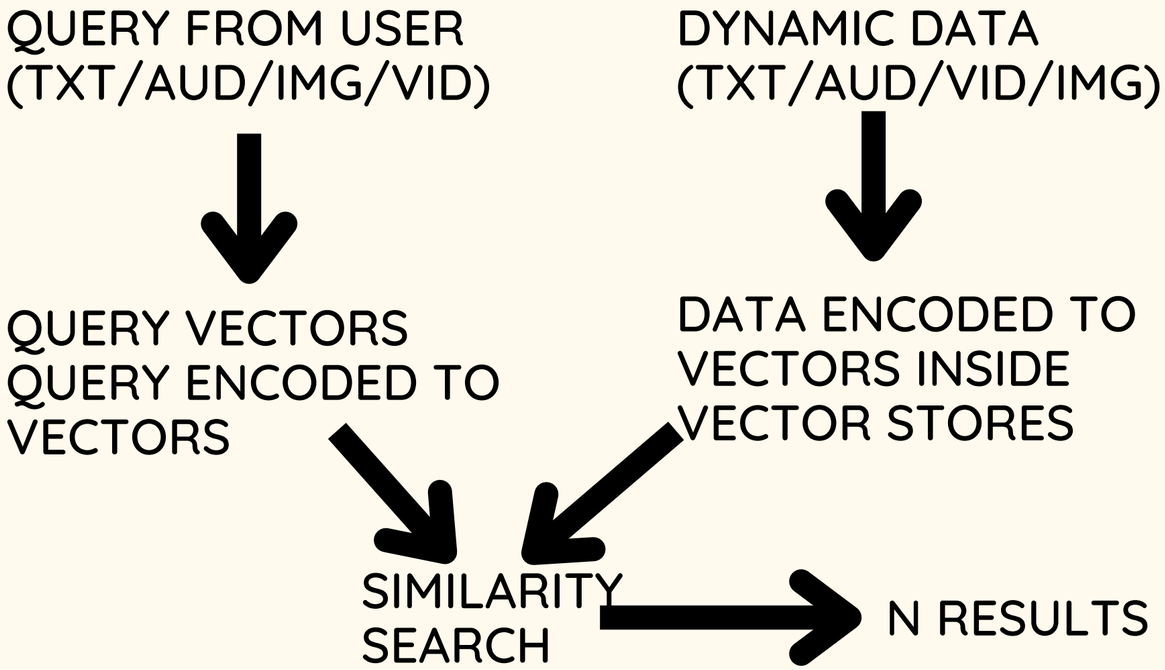

```
[{'class': 'Vim_texts',
  'creationTimeUnix': 1682749756012,
  'id': '28eff942-57f2-4229-8af5-86c9f60f2357',
  'lastUpdateTimeUnix': 1682749756012,
  'properties': {'content': 'The following are related to the activities on splits'},
  'vector': [0.38370237,
   -0.036440857,
   -0.25022212,
   -0.04745609,
   0.36459142,
   0.1071606,
   0.5608019,
   0.26321998,
   -0.49892408,
```


HTTPS://GITHUB.COM/INSIGHTBUILDER

STEPS OF SIMILARITY SEARCH

VECTORS : HAVE MAGNITUDE AND DIRECTION
SCALARS : HAVE ONLY MAGNITUDE

DIRECTION PART OF THE VECTOR IS THE KEY FOR FINDING THE SIMILARITY BETWEEN THE QUERY



COMPARING VECTOR STORES WITH DATABASE

TABLE/ RELATIONS	→	COLLECTIONS/ CLASS
KEYS (PRIMARY / SECONDARY	→	VECTORS
SELECT QUERY	→	NL QUERY + SIMILARITY SEARCH
PYTHON CLIENT LIKE PSYCOPG/ PYSPARK	→	WEVIATE, QDRANT, CHROMADB, FAISS CLIENTS

DEFINING INDEXES & SEARCH ALGORITHMS

INDEXING IS THE PROCESS OF EFFICIENTLY ORGANIZING DATA, AND IT PLAYS A MAJOR ROLE IN MAKING SIMILARITY SEARCH USEFUL BY DRAMATICALLY ACCELERATING TIME-CONSUMING QUERIES ON LARGE DATASETS

APPROXIMATE NEAREST NEIGHBORS SEARCH (ANNS). COMPARED WITH ACCURATE RETRIEVAL, WHICH IS USUALLY VERY TIME-CONSUMING, THE CORE IDEA OF ANNS IS NO LONGER LIMITED TO RETURNING THE MOST ACCURATE RESULT, BUT ONLY SEARCHING FOR NEIGHBORS OF THE TARGET. ANNS IMPROVES RETRIEVAL EFFICIENCY BY SACRIFICING ACCURACY

INDEXES DETAILS

FLAT INDEX : DATA IS STORED IN FLAT ARRAY OF FLOAT/BINARY. NO COMPRESSION. ALL INDEXES ARE DECODED WHEN A QUERY IS SENT.

IVF: INVERTED FILE IS QUANTISATION BASED. VECTORS ARE DIVIDED INTO CLUSTERS (NLISTS). ONLY THE CENTER OF THOSE CLUSTERS IS COMPARED TO QUERY VECTOR. LESSER RECALL RATE DEPENDING ON NUMBER OF RETURNS(NPROBE)

IVF SQ8: (SCALAR) EACH VECTOR IS PLACED IN A UNIT BASED ON IVF. EACH VECTOR IS CONVERTED FROM 4-BYTE FLOAT TO 1 BYTE UNSIGNED INTEGER. THE VECTORS ARE COMPRESSED, BUT ACCURACY COMPROMSED

IVF SQ8H: SAME LIKE IVF SQ8 WITH COARSER QUANTIZER INVOLVING CPU AND GPU TO SPEED UP PROCESS

IVF PQ:(PRODUCT) DECOMPOSES AND THEN QUANTIZES THE ORIGINAL VECTOR INTO M DIM X N BITS VECTOR. THE TARGET IS ALSO PQ-IZED AND THEN DISTANCE TO THE CENTER OF THE CLUSTER IS CALCULATED

RNSG (REFINED NAVIGATING SPREADING-OUT GRAPH) IT SETS THE CENTER POSITION OF THE WHOLE DATA AS A NAVIGATION POINT, & USES EDGE SELECTION STRATEGY TO CONTROL THE OUT-DEGREE.

HNSW (HIERARCHICAL SMALL WORLD GRAPH) IS A GRAPH-BASED INDEXING ALGORITHM. IT BUILDS A MULTI-LAYER NAVIGATION STRUCTURE FOR AN IMAGE ACCORDING TO CERTAIN RULES

ANNOY (APPROXIMATE NEAREST NEIGHBORS OH YEAH) IS AN INDEX THAT USES A HYPERPLANE TO DIVIDE A HIGH-DIMENSIONAL SPACE INTO MULTIPLE SUBSPACES, AND THEN STORES THEM IN A TREE STRUCTURE.

REFER TO THS WIKI : HTTPS://GITHUB.COM/FACEBOOKRESEARCH/FAISS/WIKI/

TYPES OF INDEXES: PRO N CON

[0.57,0.62....]	FLAT	N/A	Relatively small datasetRequires a 100% recall rate
[0.78,0.89....]	IVF_FLAT	Quantization-based index	High-speed query Requires a recall rate as high as possible
[0.79,0.21....]	IVF_SQ8	Quantization-based index	High-speed query Limited memory resources Accepts minor compromise in recall rate
[0.51,0.64....]	IVF_PQ	Quantization-based index	Very high-speed query Limited memory resources Accepts substantial compromise in recall rate
[0.98,0.52...]	HNSW	Graph-based index	High-speed query Requires a recall rate as high as possible Large memory resources
[0.26,0.25....]	ANNOY	Tree-Based Index	Low-dimensional vectors

VECTOR SPECS, METRICS & DECISIONS

IMPORTANT PARAMETERS TO COMPARE THE INDEX ARE AS FOLLOWS

- RECALL : TIME TAKEN FOR THE RESULTS AFTER QUERY
- QPS : QUERY PER SECOND
- IMPORT TIME : TIME TAKEN FOR THE DATA TO BE INSERTED INTO INDEX

ABOVE PARAMETERS DEPEND ON THE DISTANCE METRICS USED

- EUCLIDEAN DISTANCE : SIMPLES TO UNDERSTAND
- INNER PRODUCT : USEFUL TO FIND VECTOR DIRECTION
- JACCARD DISTANCE : MEASURES SIMILARITY B/W 2 SETS
- HAMMING DISTANCE: NUMBER OF BIT POSITION AT WHICH THE 2 STRINGS ARE DIFFERENT
- SUPER & SUB STRUCTURE: MEASURES SIMILARITY BETWEEN CHEMICAL SUPER & SUB STRUCTURE

DECISION MAKING & TRADEOFF:

- YOUR APP USER PROFILE WILL DECIDE THE TYPE OF INDEX, SIMILARITY METRICS & VECTOR STORE IT USES.
- THE PYTHON CLIENTS HAVE ALREADY ABSTRACTED THE ABOVE DETAILS AND PROVIDE API END POINTS
- BENCHMARKS ARE AVAILABLE FOR MANY OPEN SOURCE.

THANKS FOR WATCHING

HTTPS://GITHUB.COM/INSIGHTBUILDER