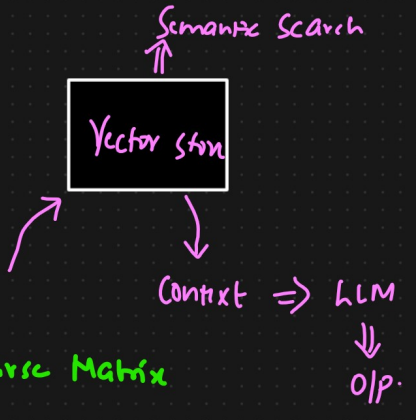
Hybrid Search Strategies

1) Combining Dense And Sparse Retrieval

Hybrid retrieval combines both dense and sparse scores (e.g. using a weighted sum or learning-to-rank methods) to improve recall and relevance.

You get the semantic power of embeddings, and the exact match of keywords — best of both worlds.

Doc ID	Content
D1	"LangChain supports agents and chains"
D2	"The Eiffel Tower is in Paris"
D3	"You can build LLM-powered apps using LangChain"



Text → Sparse Matrix

NLP

Query

BOW, TF-IDF, BM25,

Exact Search

Exact Keyword Search

Sparse Matrix

① Sparse Retrieval : Matches exacts words using techniques such as TF-IDF

Exact Search

Sparse Matrix

I want to have food => [1 0 0 0 1]

② Dense Retrival => Semantic Meaning Using Vector Embedding

FAISS, CHROMA DB

=> O/P Similar Matching Sentences

Exact Search [Sparse Matrix]
+
Semantic Search [Dense Matrix] ←

Hybrid Search

⇓

Retrieval

1) Dense Retrieval (Embeddings + Cosine Similarity)
+
2) Sparse Retrieval (TF-IDF).

$$Score_{hybrid} = \alpha * score_{dense} + (1-\alpha) * score_{sparse}$$

$Score_{dense}$ = Cosine Similarity (Inp, Vectorstore)

$Score_{sparse}$ = TFIDF (Inp, Vector Representation)

α = Weightage = 0.5

Doc ID Text
D1 "LangChain helps build LLM apps"
D2 "Pinecone is used for vector search"
D3 "Eiffel Tower is in Paris"

Your query: "build application using LLM"

Ip: Build application using LLM

Cosine Similarity [Dense Embedding]

D1	0.85
D2	0.40
D3	0.10

BOW, TF-IDF, BM25

D1	0.60
D2	0.20
D3	0.10

⇒ Sparse Matrix.

$\alpha = 0.5$

$D1 = 0.5 \times 0.85 + 0.5 \times 0.6 = \approx 0.82$

$D2 = 0.5 \times 0.40 + 0.2 \times 0.5 = \approx 0.18$

$D3 = 0.5 \times 10 + 0.3 \times 0.5 = \approx 0.04$

1