Vector Stores Research

Rithvik Nakirikanti

Understanding Vector Stores

What is a Vector Store?

A vector store is a specialized database optimized for storing and retrieving vector embeddings. Unlike traditional databases that rely on exact keyword matching, vector stores utilize approximate nearest neighbor (ANN) search techniques to find semantically similar documents based on high-dimensional vector representations.

How Vector Stores Work

Vector stores work by converting text into embeddings using machine learning models and then indexing those embeddings for fast similarity searches. Retrieval is performed based on distance metrics such as:

- Cosine Similarity: Measures the angle between two vectors, determining their similarity.
- Euclidean Distance: Measures the absolute distance between two points in vector space.
- Dot Product Similarity: Used in some neural models to evaluate relevance.

Why Vector Stores Matter in AI-Powered Search

Vector stores enable semantic search, making them essential for RAG-based applications, where the goal is to retrieve contextually relevant information from large datasets. Key advantages include:

- Scalability: Handles millions of high-dimensional embeddings efficiently.
- Fast Retrieval: Performs approximate searches in logarithmic time complexity.
- Integration with LLMs: Provides structured knowledge for generative AI.

Application of Vector Stores

Role of Vector Stores in Our Project

The project utilizes a vector store to enhance search functionality within the Boston Public Library's digital resources. By embedding metadata and abstracts from the Digital

Commonwealth API, the system enables natural language queries to return relevant document matches.

Our pipeline follows these steps:

1. Embedding Creation: Converts metadata and abstracts into vector representations.
2. Indexing: Stores the embeddings in a vector database for fast retrieval.
3. Retrieval: When a query is entered, the system fetches similar vectors using ANN techniques.
4. Re-ranking: Retrieved results are further refined using BM25 for better ranking.
5. Response Generation: The LLM generates an answer based on the retrieved documents.

Current Vector Store Implementation

The current implementation of LibRAG utilizes Pinecone as the vector database due to its cloud scalability and API-driven approach. Our configuration is designed for optimal performance and efficiency. We use all-MiniLM-L6-v2 as our embedding model, generating 384-dimensional vector embeddings. To optimize storage, only title and abstract embeddings are retained, reducing overhead while maintaining retrieval accuracy. For each query, the system retrieves the top 100 nearest neighbor vectors, ensuring relevant document matches. The results are then refined using BM25-based filtering, enhancing relevance and search precision.

Limitations and Potential Improvements

Despite the effectiveness of our current vector store setup, there are several areas for improvement:

Storage Constraints and Cost Optimization

- Issue: Pinecone's cloud-based vector storage incurs high costs, especially as our dataset grows.
- Improvement: Implement a hybrid retrieval approach by storing frequently used embeddings in a local FAISS/Chroma database and using Pinecone only for large-scale queries.
- Alternative: Experiment with vector quantization techniques like Product Quantization (PQ) to reduce storage overhead.

Reducing Retrieval Latency

- Issue: Queries currently take 25-70 seconds, mainly due to metadata fetching from the Digital Commonwealth API.

- Improvement:
  - Preload metadata into a structured SQL or NoSQL database to eliminate API delays.
  - Parallelize metadata retrieval with asynchronous calls to speed up processing.

Query Alignment and Semantic Precision

- Issue: The quality of retrieval depends heavily on query wording.
- Improvement:
  - Implement query preprocessing to extract structured elements (e.g., names, dates) and align them with metadata fields.
  - Use multiple vector stores for different metadata categories (e.g., title, author, year) and merge results.

Long-Term Scalability

- Issue: As dataset size increases, query efficiency may degrade.
- Improvement:
  - Investigate self-hosted alternatives like Milvus or Weaviate for long-term cost reduction.
  - Optimize indexing strategies by testing different ANN search methods (HNSW, IVF, PQ).

Resources

1. Facebook AI Similarity Search (FAISS)
   - https://arxiv.org/pdf/1702.08734.pdf
2. Pinecone Blog on Vector Databases
   - https://www.pinecone.io/learn/vector-database/
3. Retrieval-Augmented Generation (RAG) - Deep Dive by Meta AI
   - https://research.facebook.com/publications/retrieval-augmented-generation-for-knowledge-intensive-nlp-tasks/
4. Vector Indexing Methods Explained (HNSW, IVF, PQ)
   - https://towardsdatascience.com/vector-search-in-a-nutshell-6f0a44a5ddaa