Vector Database 101

Amber Liu

2023/06
Funding to Vector DB Takes Off

Until 4/27/2023
Agenda

- Vector data
- Vector Index
- Vector Database
- Vector DB Nowadays
Why We Need **Vector DB**?

Unstructured data

> 80%

- social media posts
- images
- video
- audio
Query for Unstructured Data?

<table>
<thead>
<tr>
<th>Type</th>
<th>Color</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>Yellow</td>
<td>Small</td>
</tr>
<tr>
<td></td>
<td>Brown</td>
<td>Fat</td>
</tr>
</tbody>
</table>

[0.12, 0.45,..., 1.2, -0.4]
[0.24, 0.56,..., 2.0, 1.1]
Vector Database

Index and Store vector embeddings
For fast retrieval and similarity search

1. Vector embedding generation
2. Vector Indexing
3. Vector database
A. Vector Embedding

Audio
→ Audio model
→ Audio vector embeddings

Text
→ Texts model
→ Text vector embeddings

Videos
→ Videos model
→ Video vector embeddings
A. Vector Embedding

1. word2vec
2. GloVe
3. FastText
4. Model-based
   • Contrastive pre-training | OpenAI

---

Text and Code Embeddings by Contrastive Pre-Training

- Arvind Neelakantan
- Tao Xu
- Raul Puri
- Alec Radford
- Jesse Michael Han
- Jerry Tworek
- Qiming Yuan
- Nikolas Tezak
- Jong Wook Kim
- Chris Hallacy
- Johannes Heidecke
- Pranav Shyam
- Boris Power
- Tyna Eloundou Nekoul
- Girish Sastry
- Gretchen Krueger
- David Schnurr
- Felipe Petroski Such
- Kenny Hsu
- Madeleine Thompson
- Tabarak Khan
- Toki Sherbakov
- Joanne Jang
- Peter Welinder
- Lilian Weng
1. Initialize Transformer encoder with GPT
2. Select M example pairs
   a. Within each pair: semantically similar
   b. Across pairs: negative examples
3. Calculated similarity
4. Minimize loss
   a. Increase similarity within each pair
   b. Decrease similarity across pairs
5. Output: last hidden layer

Text and Code Embeddings by Contrastive Pre-Training
B. Vector Index

1. **Product Quantization (PQ)**
2. Hierarchical Navigable Small World (HNSW)
3. Locality-Sensitive Hashing (LSH)
4. …
Efficiently **compress** high-dimensional data while minimizing information loss for fast **similarity search**.
Product Quantization II

Data dimension = 1024
Data size = 50k
# sub-vector = 8
Dimension of sub-vector = 128
# centers in subspace = 256
Distance approximation

Fig. 2. Illustration of the symmetric and asymmetric distance computation. The distance $d(x, y)$ is estimated with either the distance $d(q(x), q(y))$ (left) or the distance $d(x, q(y))$ (right). The mean squared error on the distance is on average bounded by the quantization error.
Product Quantization III
Metrics about Vector Index

1. Query Latency, Indexing Time, Index Size, Recall, Precision
2. **Scalability** w.r.t size and dimension
3. **Update/Insertion/Deletion Efficiency**
4. Robustness to Data Distribution
5. Support for Different Distance Metrics
C. Vector Database Systems

1. Distributed System Design: Horizontal scalability; availability
2. Memory Management: Memory-efficient data storage, caching, resource utilization
3. Security and Access Control:
4. Flexible Interface
Why Vector DBs are so **HOT**?

1. Enable LLM with **long-term memory**
Benchmark

https://objectbox.io/vector-database/
https://qdrant.tech/benchmarks/