大模型系列 – 数据处理

相似性度量

大模型业务全流程

数据 & 模型算法

2. 数据处理

- 开源数据
- 数据预处理
- 向量数据库

3. 模型算法

- LLM模型架构
- 多模态一切皆Tokens

模型训练 & 微调

4. 模型训练

- 混合精度
- 梯度检查
- 梯度累积
- …
- 5. 分布式并行
- 训练集群稳定性

6. 模型微调

- 全参微调
- 低参微调
- 指令微调

模型验证 & 推理部署

7. 模型验证

- 下游任务
- 测评标准
- bench mark

8. 推理与智能体

- 量化压缩
- 推理加速
- 9. Agent 智能体

集群算力准备

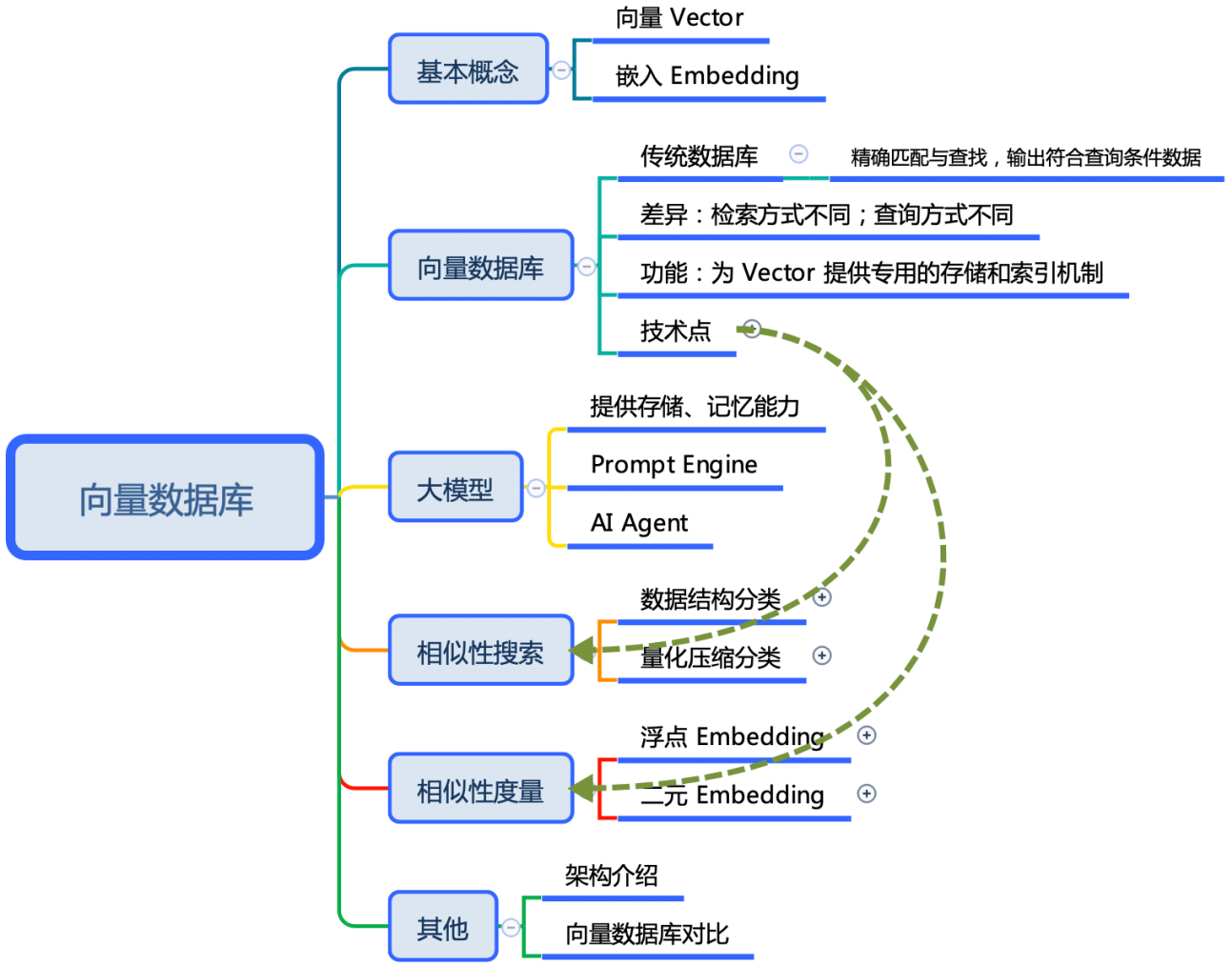
1. AI 集群建设

- 计算、存储、网络
- AI 集群机房建设
- AI 集群上线与运维

大模型系列 – 数据处理之向量数据库

- 具体内容
  - **向量与检索：**向量 Vector 的表示 -- Embedding 原理
  - **向量数据库：**向量数据库原理、功能、特点 -- Vector-DB 应用场景
  - **大模型关系：**向量数据库遇到大模型 – 大模型与 Vector-DB 应用场景
  - **相似性搜索：**K-Means 聚类 -- Faiss 算法 -- PQ 算法 -- IVF 算法 -- HNSW 算法
  - **相似性度量：**欧氏距离（L2） -- 内积（IP） -- 其他度量方式
  - **通用性架构：**通用 Vector-DB 架构 -- KDB 架构示例
  - **对比与小结：**业界向量数据库横向对比 -- Vector-DB 小结

大模型系列 – 数据处理之向量数据库



思考

1. 已经讨论 Vector-DB 搜索算法，还没有讨论如何衡量向量相似性。相似性搜索中，需要计算两向量间距离，根据距离来判断其相似度。
2. 如何计算向量在高维空间距离呢？三种常见相似度算法：欧几里德距离、余弦相似度、点积相似度。

1. 相似性度量

Similarity Metrics

向量相似度

- **相似度度量：**用于衡量向量间相似性。好的距离度量方法可以显著提高向量分类和聚类性能。
- 计算相似度，常用方法是计算两个向量之间的距离，两个向量间距离越近，则两个向量越相似。

向量相似度

- **相似度度量：**用于衡量向量间相似性。好的距离度量方法可以显著提高向量分类和聚类性能。

- **浮点 Embedding**，通常使用欧氏距离和内积：
  1. **欧氏距离（L2）**：常用于计算机视觉领域（CV）。
  2. **余弦（Cos）**：常用于自然语言处理领域（NLP）。
  3. **内积（IP）**：常用于自然语言处理领域（NLP）。

向量相似度

- **相似度度量：**用于衡量向量间相似性。好的距离度量方法可以显著提高向量分类和聚类性能。

- **二元 Embedding**，广泛使用度量标准：

1. **哈明距离**：常用于自然语言处理领域（NLP）。
2. **杰卡德距离**：常用于分子相似性搜索领域。
3. **塔尼莫托距离**：常用于分子相似性搜索领域。
4. **超结构距离**：常用于搜索分子的类似超结构。
5. **亚结构距离**：常用于搜索分子的类似亚结构。

Euclidean Distance 欧几里得距离

- 欧几里得距离指两向量 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 间距离，计算公式为：

$$d(\boldsymbol{A},\boldsymbol{B}) = \sqrt{\sum_{i=1}^{n}(A_i - B_i)^2}$$

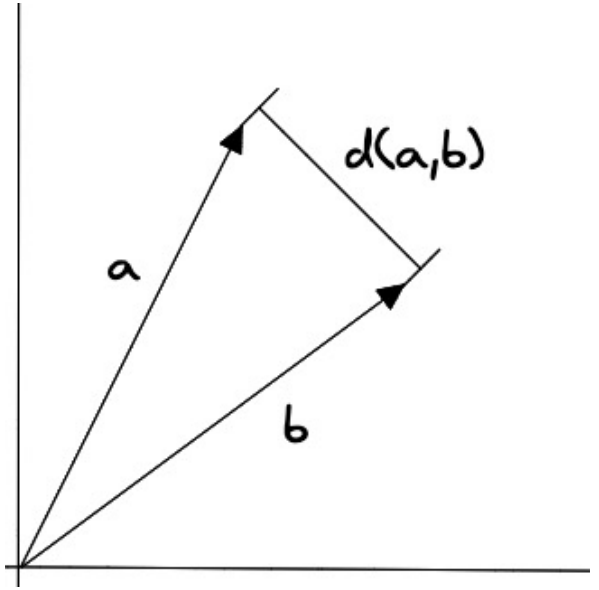
- 其中， $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 表示两个比较的向量，n 表示向量维度。

Euclidean Distance 欧几里得距离

- 欧几里得距离指两向量 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 间距离，计算公式为：

$$d(\boldsymbol{A},\boldsymbol{B}) = \sqrt{\sum_{i=1}^{n}(A_i - B_i)^2}$$

- **优点**：反映向量绝对距离，适用于需要考虑向量长度相似性计算。



Cosine Similarity 余弦相似度

- 两个向量间夹角余弦值，其计算公式为：

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{|\mathbf{A}||\mathbf{B}|}$$

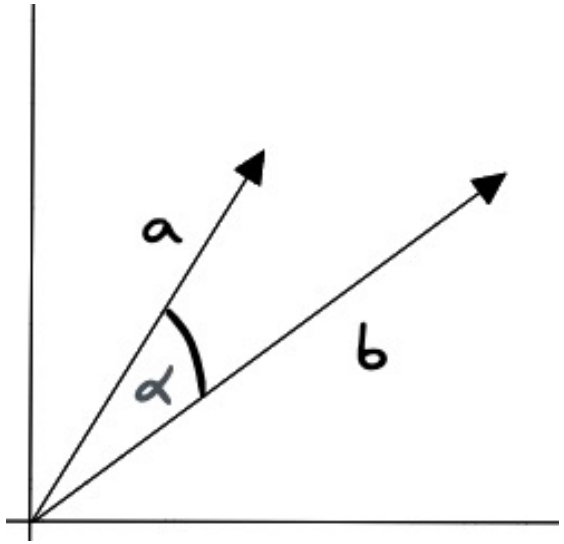
- A 和 B 表示两个比较的向量
- $|\mathbf{A}|$ 和 $|\mathbf{B}|$ 表示两个向量的模长
- $\cdot$ 表示向量点积

Cosine Similarity 余弦相似度

- 两个向量间夹角余弦值，其计算公式为：

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{|\mathbf{A}||\mathbf{B}|}$$

- **优点**：对向量长度不敏感，关注向量方向，适用于高维向量相似性计算。
- **场景**：e.g. 语义搜索和文档分类。



Dot product Similarity 点积相似度

- 指两个向量之间点积值，其计算公式为：

$$\mathbf{A} \cdot \mathbf{B} = \sum_{i=1}^{n} A_i B_i$$

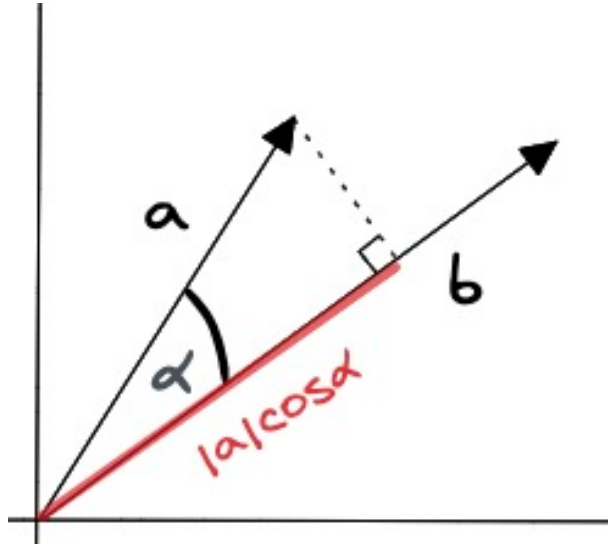
- $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 表示两个比较的向量，n 表示向量维度。

Dot product Similarity 点积相似度

- 指两个向量之间点积值，其计算公式为：

$$\mathbf{A} \cdot \mathbf{B} = \sum_{i=1}^{n} A_i B_i$$

- **优点**：它简单易懂，计算速度快，兼顾向量长度和方向。
- **场景**：适用图像识别、语义搜索和文档分类等。
- **问题**：对向量长度敏感，计算高维向量相似性时会丢失信息。



2. 过滤

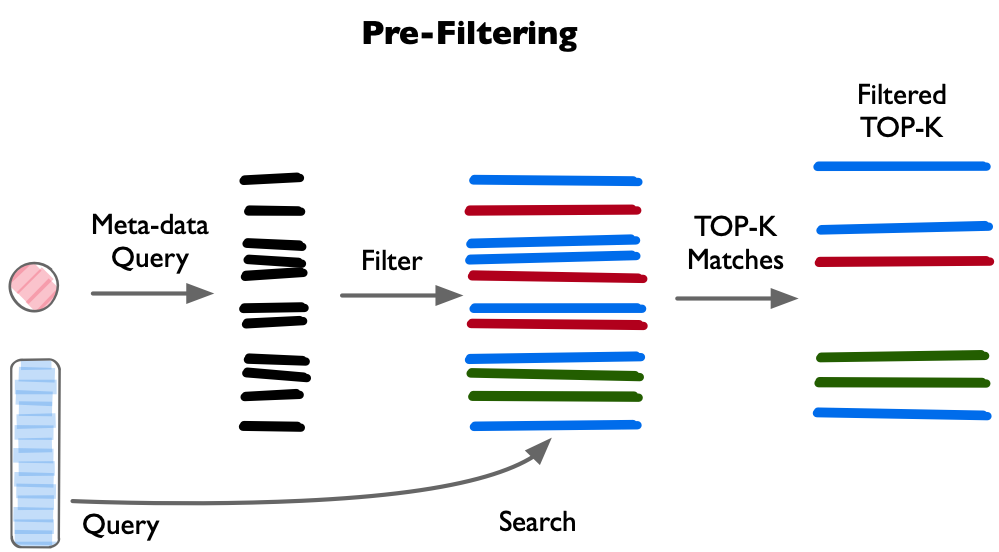
Filtering

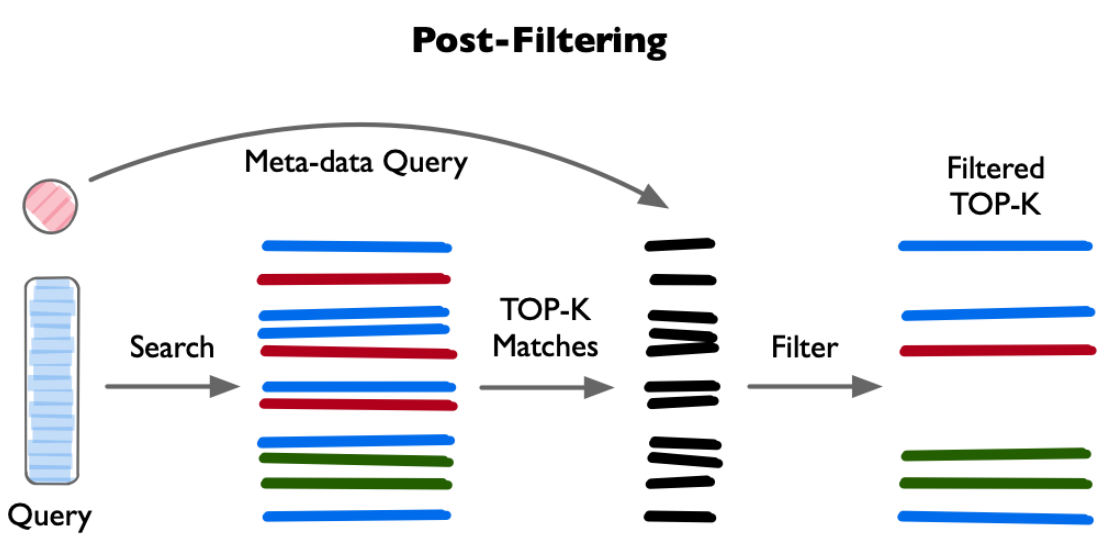
why Filtering ?

- 实际业务场景，不需整 Vector-DB 相似性搜索，而通过部分业务字段进行过滤 Filtering 再查询。
- 存储在 Vector-DB 向量还需包含元数据 Mate Data，e.g. User ID、Doc ID 等信息。
- **优点**：搜索时，根据元数据来过滤搜索结果，减少向量检索范围。

Indexing

- 进行相似性搜索前 or 后执行元数据过滤，Vector-DB 一般维护两个索引：
  1. 向量索引 Vector Indexing
  2. 元数据索引 Meta Indexing
- **缺点**：会导致 Vector-DB 查询过程变慢。



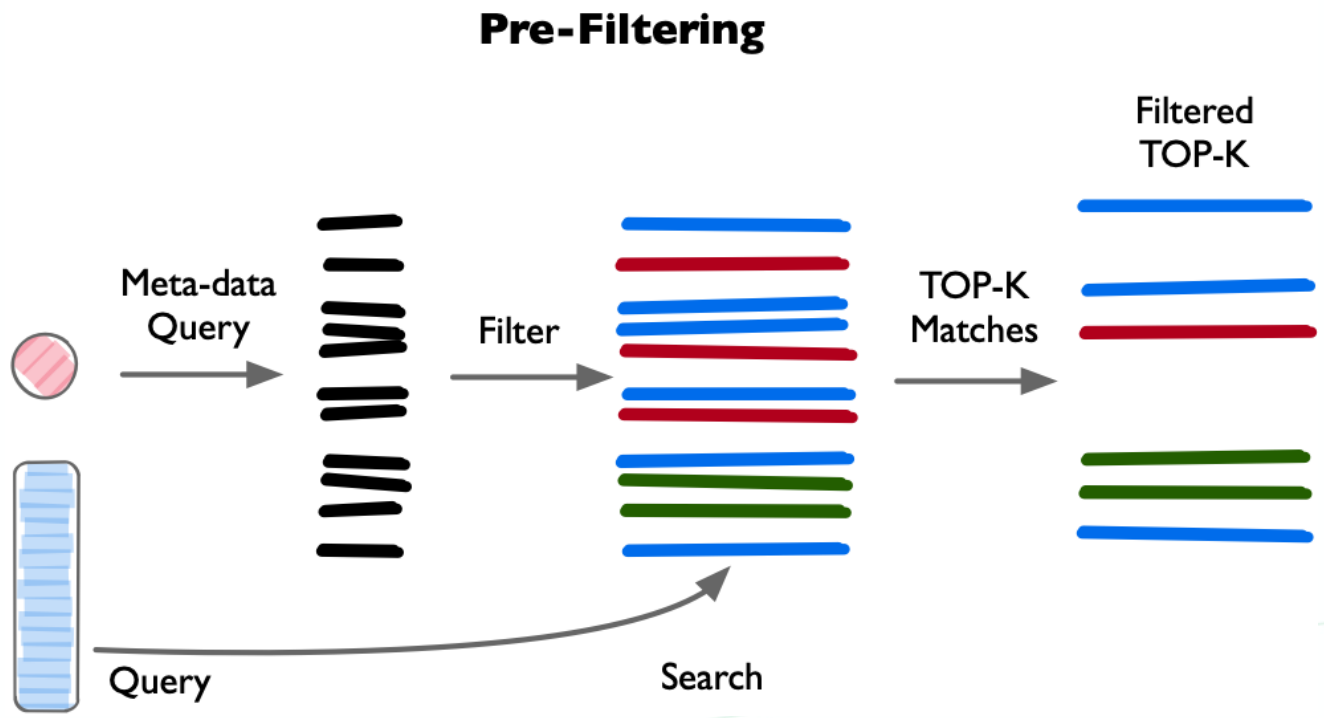


过滤方式

- Filtering 可以在向量搜索前 or 后执行，但每种方法都有其挑战：
  1. **Pre-filtering**：向量搜索前进行 Meta-data 过滤。可以减少搜索空间，但也可能导致系统忽略与 Meta-data 筛选标准不匹配的相关结果。
  2. **Post-filtering**：向量搜索完后进行 Meta-data 过滤。可以确保考虑所有相关结果，搜索完成后将不相关结果进行筛选。

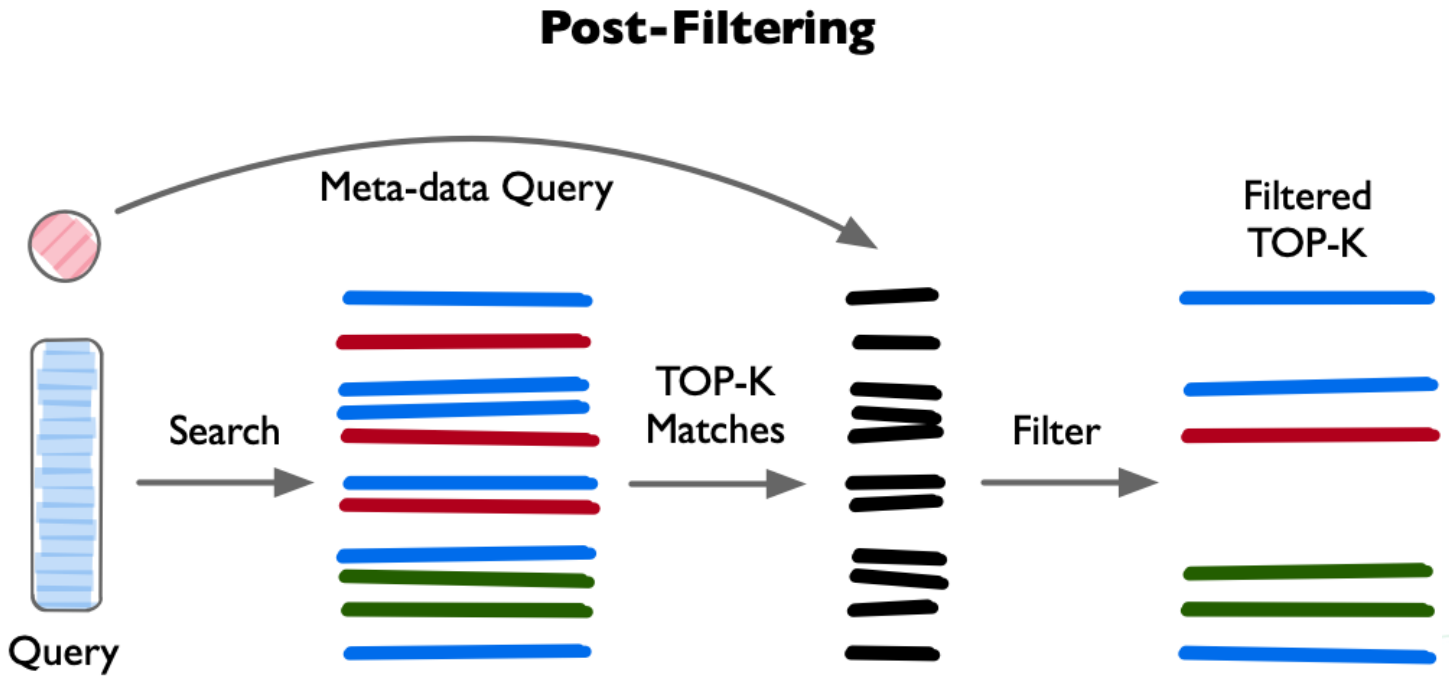
过滤方式

- **Pre-filtering**：向量搜索前进行 Meta-data 过滤。可以减少搜索空间，但也可能导致系统忽略与 Meta-data 筛选标准不匹配的相关结果。



过滤方式

- **Post-filtering**：向量搜索完后进行 Meta-data 过滤。可以确保考虑所有相关结果，搜索完成后将不相关结果进行筛选。



More...

- 为了优化过滤流程，Vector-DB 使用各种技术
  1. e.g. 利用索引算法处理 Meta-data 或使用并行算法来加速过滤任务。
  2. 平衡搜索性能和筛选精度，这对提供高效且相关查询结果至关重要。

Reference 引用&参考

1. Maximizing the Potential of LLMs: Using Vector Databases (ruxu.dev)
2. Maglott D， Ostell J， Pruitt KD， Tatusova T. Entrez Gene: gene-centered information at NCBI. Nucleic Acids Res. 2005 Jan 1;33 (Database issue):D54-8.
3. Wang Y， Xiao J， Suzek TO， Zhang J， Wang J， Zhou Z， Han L， Karapetyan K， Dracheva S， Shoemaker BA， Bolton E， Gindulyte A， Bryant SH. PubChe m's BioAssay Database. Nucleic Acids Res. 2012 Jan;40 (Database issue):D400-12.
4. Torng W， Altman RB. 3D deep convolutional neural networks for amino acid environment similarity analysis. BMC Bioinformatics. 2017 Mar 16;18 (1):302.
5. Zheng S， Shao W， Chen L. UniVec: a database of gene expression vectors for PCA based gene similarity search. BMC Genomics. 2017 Dec 6;18 (Suppl 10):918.
6. Manning CD， Raghavan P， Schütze H. Introduction to Information Retrieval. Cambridge: Cambridge University Press， 2008.
7. Mikolov， Tomas， et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).
8. Andoni， Alexandr， and Piotr Indyk. "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions." Communications of the ACM 51.1 (2008): 117-122.
9. Jégou， Hervé， et al. "Product quantization for nearest neighbor search." IEEE transactions on pattern analysis and machine intelligence 33.1 (2010): 117-128.
10. Ge， Tiezheng， et al. "Optimized product quantization." IEEE transactions on pattern analysis and machine intelligence 36.4 (2013): 744-755.
11. Babenko， Artem， and Victor Lempitsky. "The inverted multi-index." IEEE transactions on pattern analysis and machine intelligence 37.6 (2014): 1247-1260.
12. Datar M， Immorlica N， Indyk P， Mirrokni VS. Locality-sensitive hashing scheme based on p-stable distributions. Proceedings of the twentieth annual symposium on Co mputational geometry. 2004 Jun 8:253-62.
13. Muja M， Lowe DG. Fast approximate nearest neighbors with automatic algorithm configuration. VISAPP (1). 2009 Feb 4:331-40.
14. Jégou， Hervé， et al. "Product quantization for nearest neighbor search." IEEE transactions on pattern analysis and machine intelligence 33.1 (2011): 117-128.
15. Chen， Zhenjie， and Jingqi Yan. "Fast KNN search for big data with set compression tree and best bin first." 2016 2nd International Conference on Cloud Computing an d Internet of Things (CCIOT). IEEE， 2016.
16. Dehmamy， Nima， Albert-László Barabási， and Rose Yu. "Understanding the representation power of graph neural networks in learning graph topology." Advances in Neural Information Processing Systems 32 (2019).
17. Babenko A， Lempitsky V. The inverted multi-index. IEEE transactions on pattern analysis and machine intelligence. 2014 Jun 7;37 (6):1247-60.

Reference 引用&参考

1. https://www.bilibili.com/video/BV11a4y1c7SW
2. https://www.bilibili.com/video/BV1BM4y177Dk
3. https://www.pinecone.io/learn/vector-database/
4. https://github.com/guangzhengli/ChatFiles
5. https://github.com/guangzhengli/vectorhub
6. https://www.anthropic.com/index/100k-context-windows
7. https://js.langchain.com/docs/
8. https://www.pinecone.io/learn/series/faiss/locality-sensitive-hashing/
9. https://www.pinecone.io/learn/series/faiss/product-quantization/
10. https://www.pinecone.io/learn/series/faiss/locality-sensitive-hashing-random-projection/
11. https://www.youtube.com/watch?v=QvKMwLjdK-s&t=168s&ab_channel=JamesBriggs
12. https://www.pinecone.io/learn/series/faiss/faiss-tutorial/
13. https://www.youtube.com/watch?v=sKyvsdEv6rk&ab_channel=JamesBriggs
14. https://www.pinecone.io/learn/vector-similarity/
15. https://github.com/chroma-core/chroma
16. https://github.com/milvus-io/milvus
17. https://www.pinecone.io/
18. https://github.com/qdrant/qdrant
19. https://github.com/typesense/typesense
20. https://github.com/weaviate/weaviate
21. https://redis.io/docs/interact/search-and-query/
22. https://github.com/pgvector/pgvector

Thank you

把AI系统带入每个开发者、每个家庭、每个组织，构建万物互联的智能世界

Bring AI System to every person, home and organization for a fully connected, intelligent world.

ZOMI

Course chenzomi12.github.io

GitHub github.com/chenzomi12/DeepLearningSystem