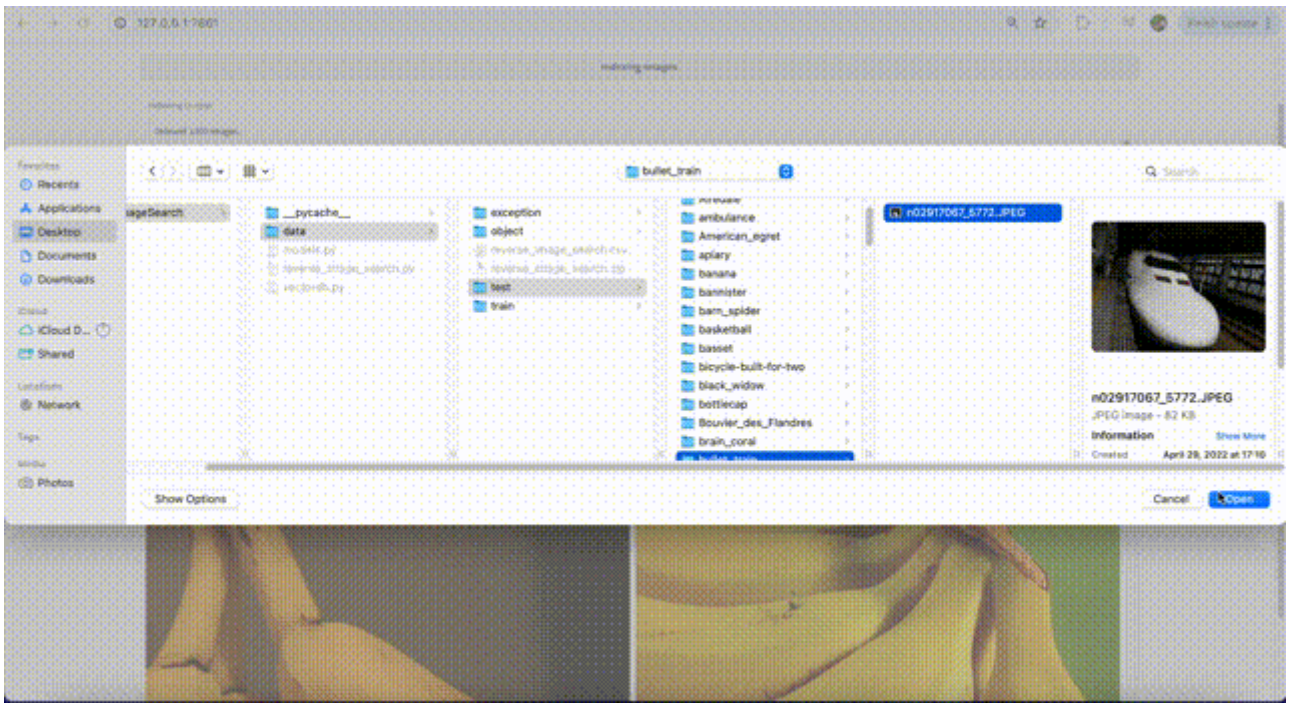
以图识图（反向图片搜索）开发与原理



1. 项目概述

- **Reverse Image Search**
- **Image-Based Search**

1. **寻找图片来源**：确定图片的原始来源或了解图片背后的故事。
2. **版权验证**：检查图片是否被未经授权地使用，或确认图片的版权信息。
3. **相似图片搜索**：寻找风格、内容或主题相似的图片。
4. **产品识别**：识别图片中的产品，并找到购买链接或更多产品信息。
5. **视觉相似性**：在设计、艺术作品或图像识别项目中，找到视觉上相似或匹配的图片。
6. **内容过滤**：帮助过滤或屏蔽不适当或重复的内容。
7. **社交媒体监控**：监控品牌或个人的社交媒体形象，了解如何被公众分享和使用。

8. **学术研究**：在学术研究中，用于验证图像数据的准确性和来源。

9. **图像数据库管理**：帮助管理和组织大量的图像数据。

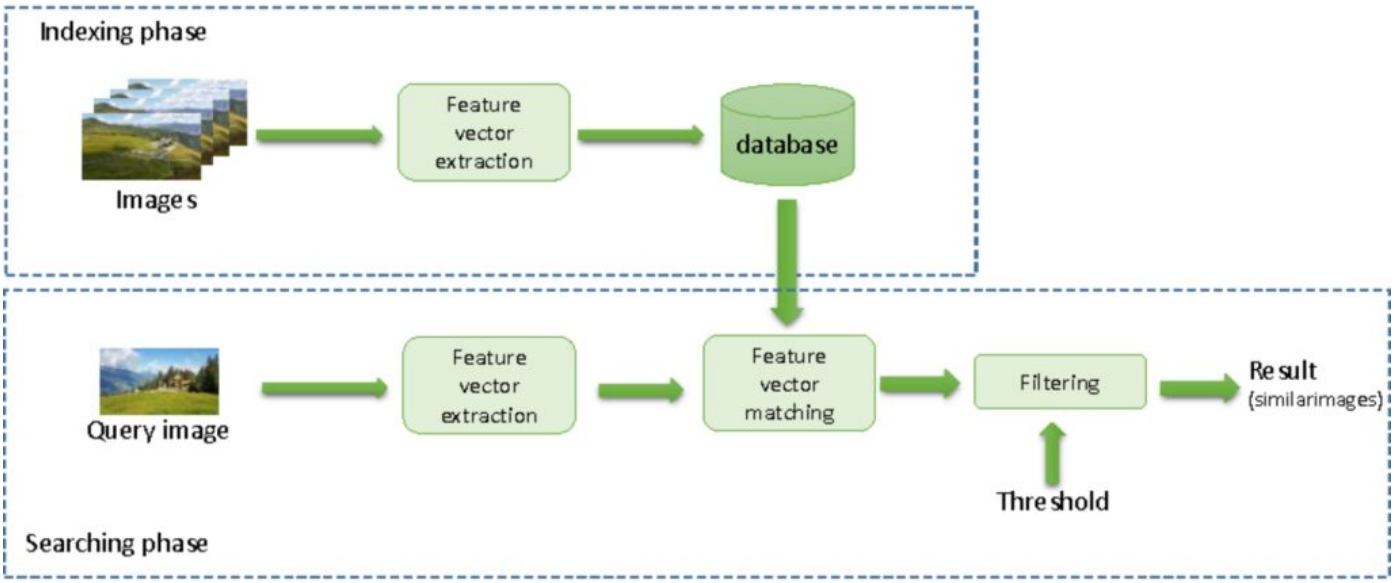
10. **辅助视觉障碍人士**：帮助视觉障碍人士通过图像识别技术了解图片内容。

11. **增强现实（AR）和虚拟现实（VR）**：在AR和VR应用中，用于识别现实世界中的物体，提供交互体验。

12. **教育和培训**：在教育领域，帮助学生通过图片找到更多相关信息和学习资源。

13. **娱乐和游戏**：在游戏和娱乐应用中，提供基于图像的互动和搜索功能。

2. 主要技术

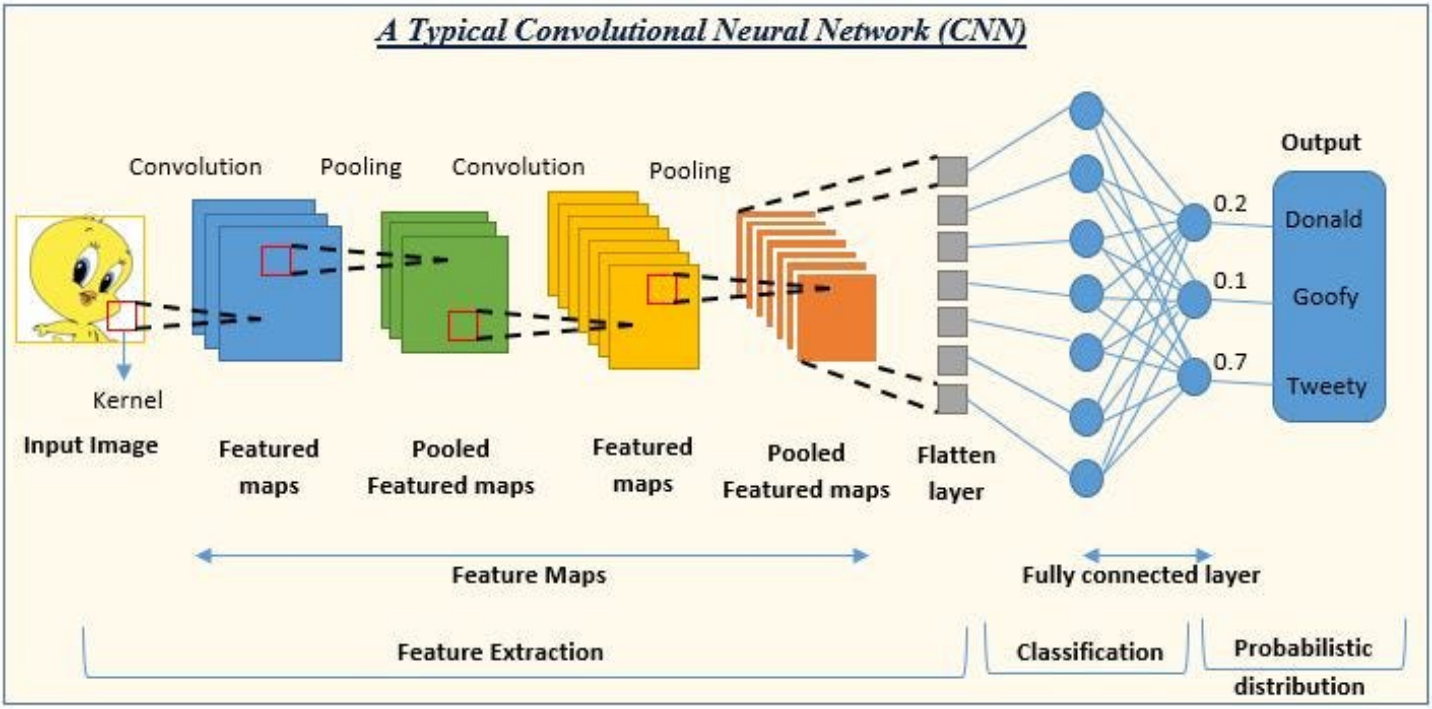


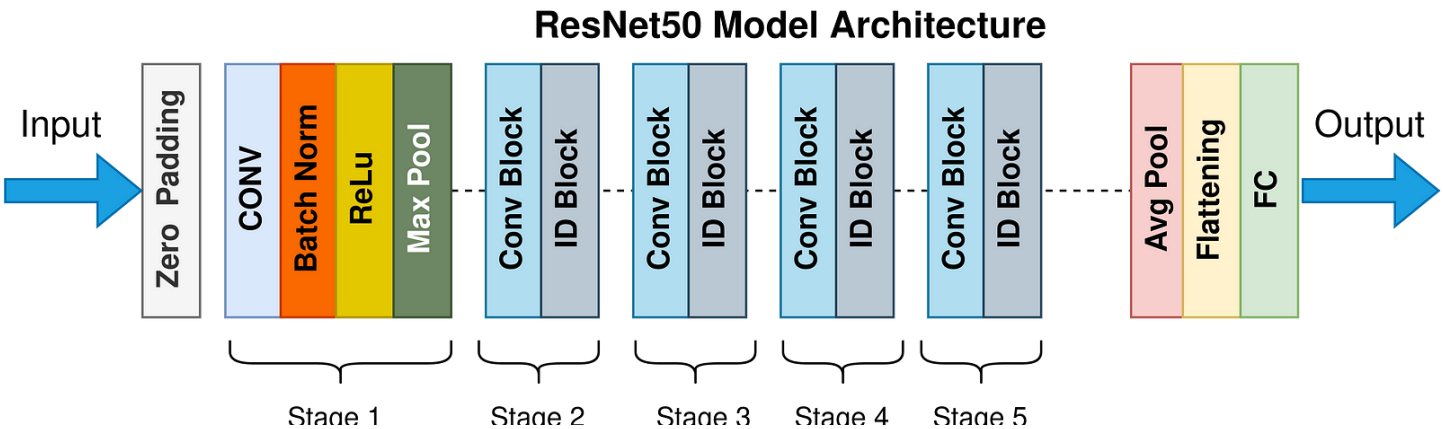
General workflow for reverse image search

- **图像特征提取**
- **相似度比较**
- **向量数据库**
- **图形界面**

3. 图像特征

- 几何特征
- 形状特征
- 幅值特征
- 。。。。
- HOG特征
- SIFT特征
- Haar特征
- 深度学习特征





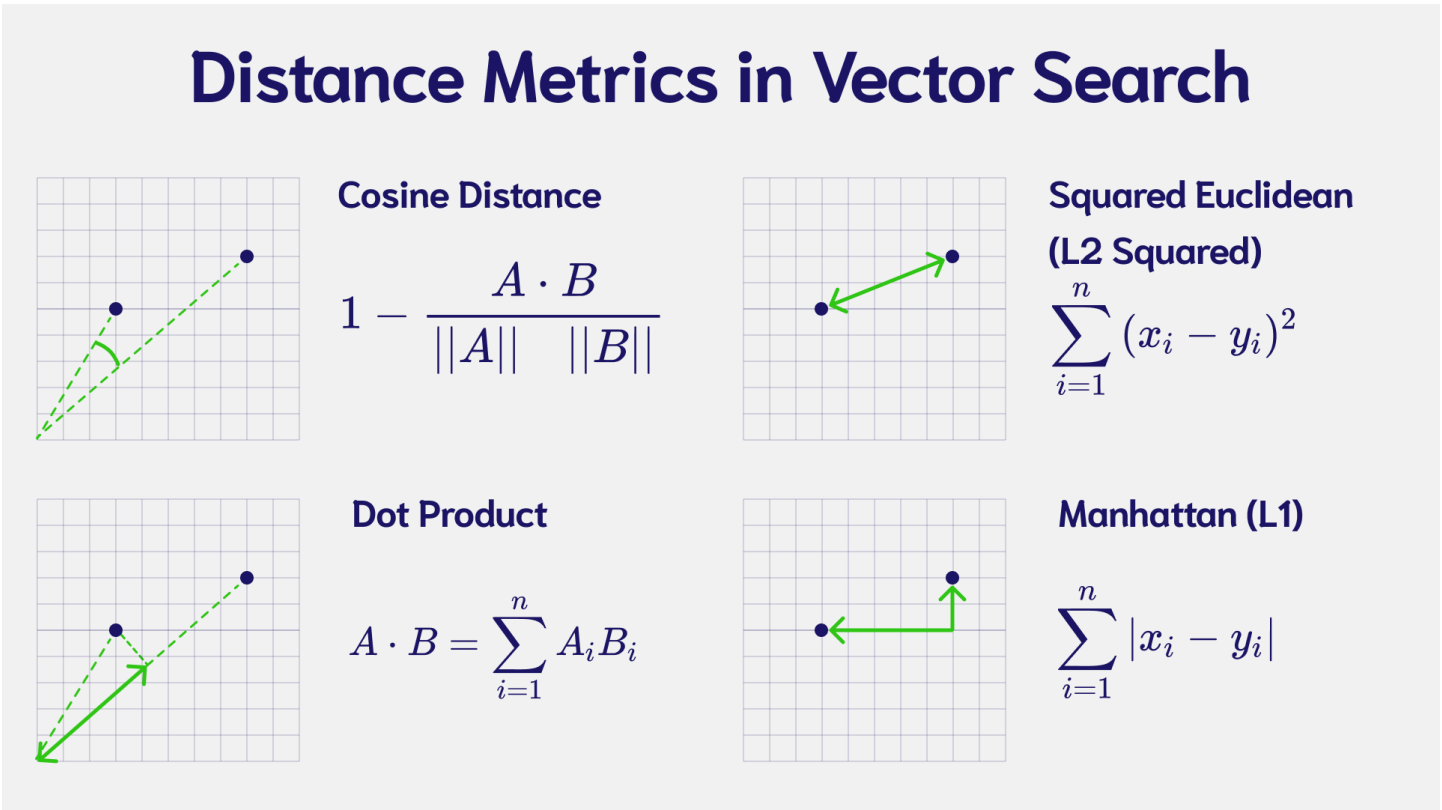
```
array([[ 6.48558010e-01, -7.31574174e-01,  8.34732140e-01,
        -6.95522718e-02, -1.19162460e-01, -3.86208637e-02,
        -1.54963455e+00,  9.96426133e-01, -3.56205319e-01,
         1.39329035e+00, -1.55569899e+00,  1.49137425e+00,
        -2.74209808e+00, -1.27639992e-01, -7.94773369e-01,
         4.41134509e-01, -8.75995218e-01, -9.21835376e-01,
         5.99707214e-01, -1.60583706e+00,  1.18365468e+00,
        -3.20999524e-01,  1.76413408e+00, -1.45934107e+00,
        -1.29465086e+00, -5.50775824e-01, -1.05203143e+00,
        -4.20362294e-02,  2.83038269e-01,  1.51992122e-01,
        -2.48745407e-01,  7.91948494e-01, -4.04546150e-01,
         9.29581175e-02, -1.40017572e-01, -1.07075369e+00,
         6.20790501e-01,  2.40914780e-01,  1.34417580e+00,
         2.47982283e-01,  1.93385242e-01,  8.32165132e-01,
         4.56408677e-01,  7.58521891e-01,  5.27831053e-02,
         1.14158253e+00, -1.68101395e-01, -7.79556683e-02,
        -7.29304970e-01, -8.04259597e-01,  9.25894135e-01,
         1.62594921e+00, -1.14499974e+00,  4.37738882e-01,
        -1.77186139e-01, -1.46990906e-01,  4.40039041e-01,
         1.07757218e+00, -1.53199310e+00, -4.73638682e-01,
         2.76123176e-01, -1.50882365e+00, -3.20495955e+00,
         1.10829683e-01, -3.03008012e-01,  2.86756555e-01,
        -1.40538749e+00,  1.23784241e+00, -7.60325619e-01,
         9.80366468e-02,  1.16398106e+00,  3.02443173e-01,
         1.31030762e+00,  3.64284573e-01,  2.90140038e+00,
...
         4.94936620e-01, -6.98741828e-01, -3.37547593e-01,
        -4.62985357e-02,  6.61291299e-01, -1.32362866e+00,
         6.02243218e-01, -3.49765674e-01, -8.80193185e-01,
        -3.98808821e-01, -2.11313289e-01, -1.31412282e+00,
        -5.58546245e-01]])
```

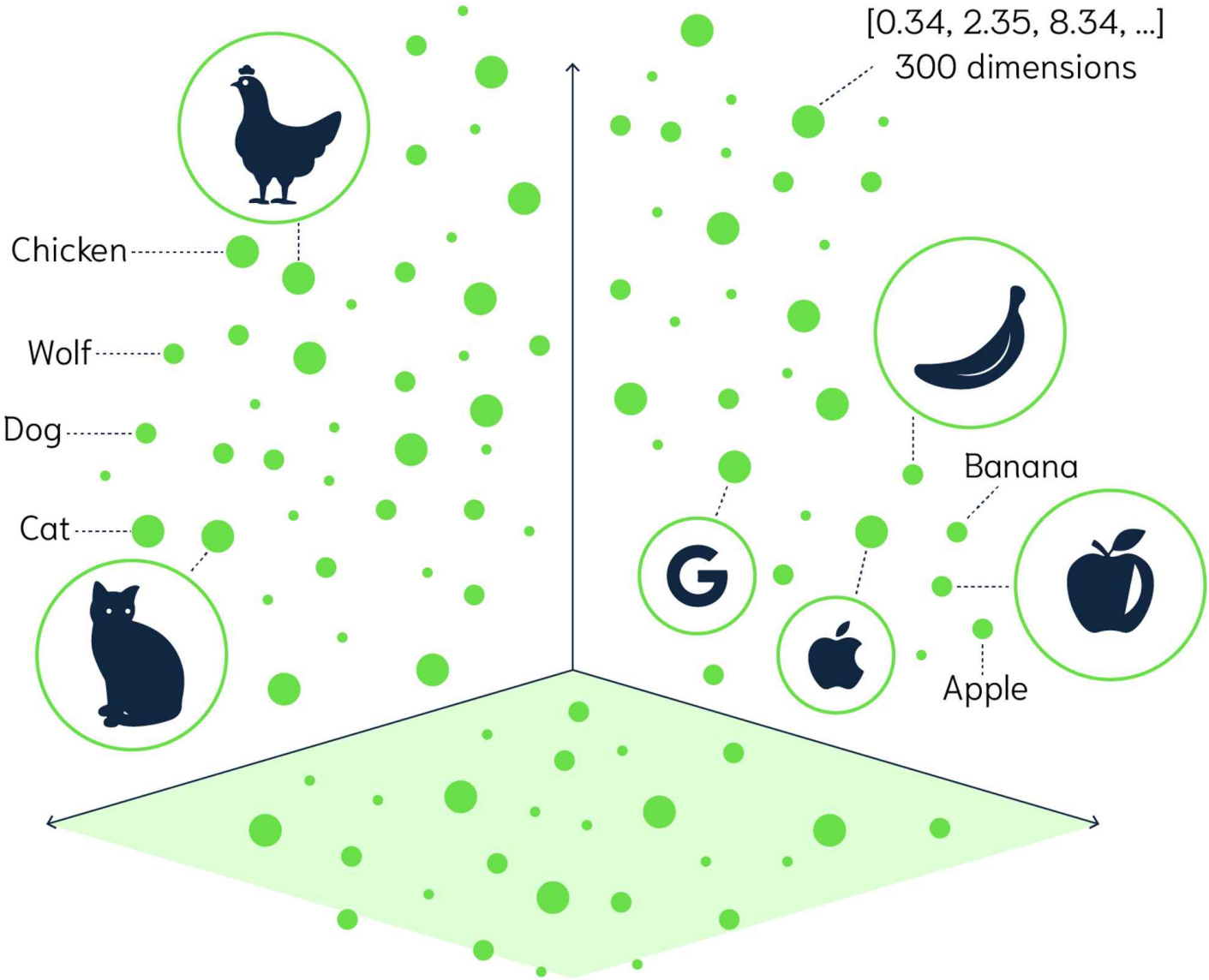
图像特征本质上是一种图像信息的压缩或抽象化。记录了图像中的关键信息，去除冗余，降低维度。

表示为：**[0.81, 0.23, 0.34, 0.56, 0.92, ...]**

4. 相似度比较

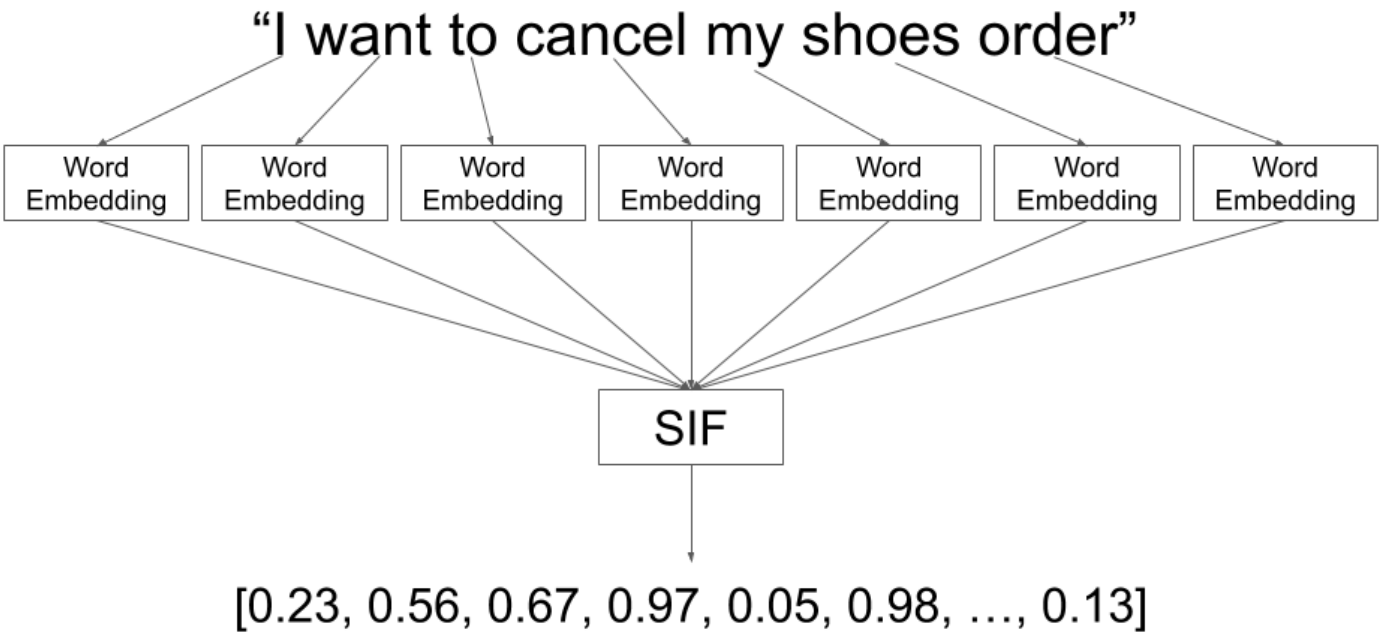
- 图像1: **[0.81, 0.23, 0.34, 0.56, 0.92, ...]**
- 图像2: **[0.50, 0.82, 0.71, 0.21, 0.34, ...]**
- 图像3: **[0.86, 0.21, 0.39, 0.52, 0.89, ...]**
- 如何度量呢?





5. 向量数据库

- 文本或句子也可以表示为向量
- 向量无处不在



- 结构化数据 大量出现
- JSON、XML等半结构化数据 大量出现
- 向量数据 大量出现

- 关系型数据库
- NoSQL 数据库
- 向量数据库

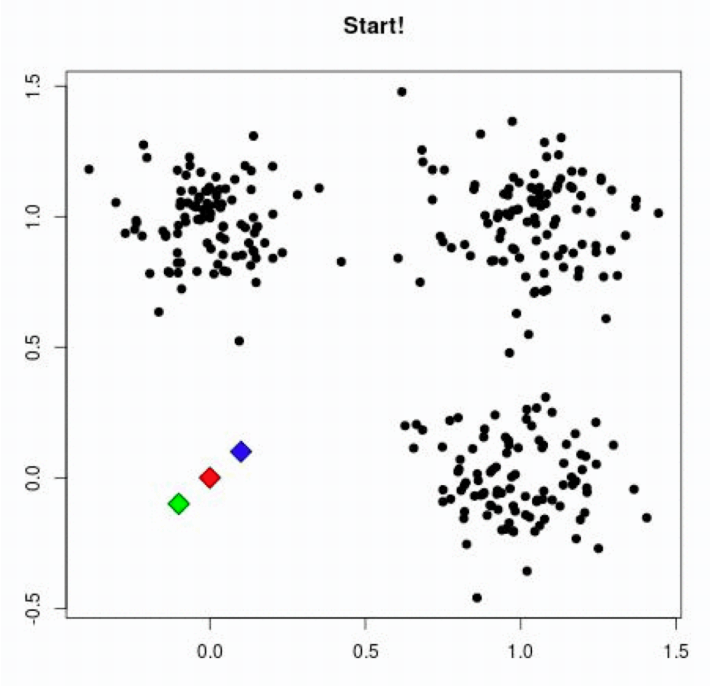
解决大数据量时数据的存储、管理和高效检索的问题，避免brute force。

a) 向量数据库的特点

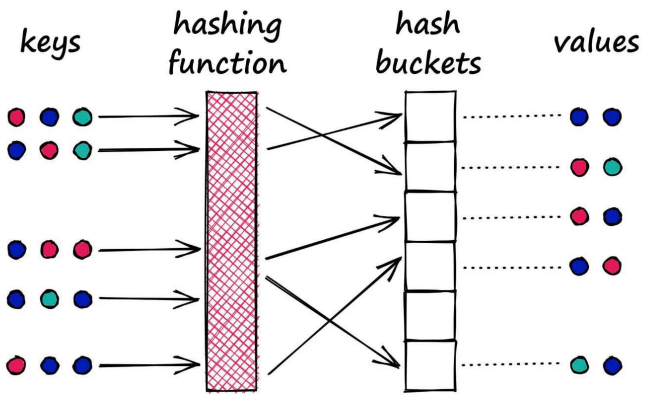
1. 向量数据存储
2. 高效的相似性搜索
3. 支持高维数据
4. 距离度量
5. 索引结构

b) 常见索引结构

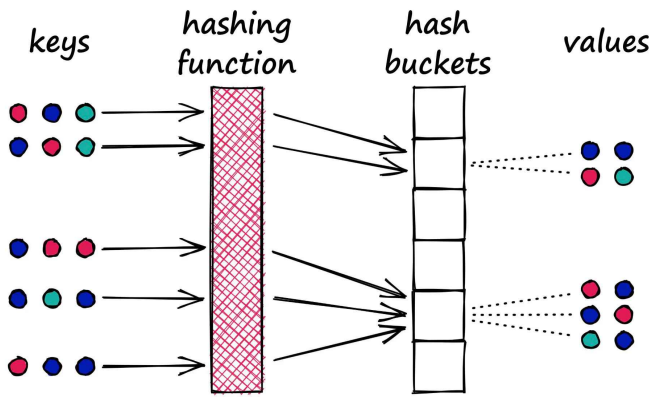
1. **K-means**



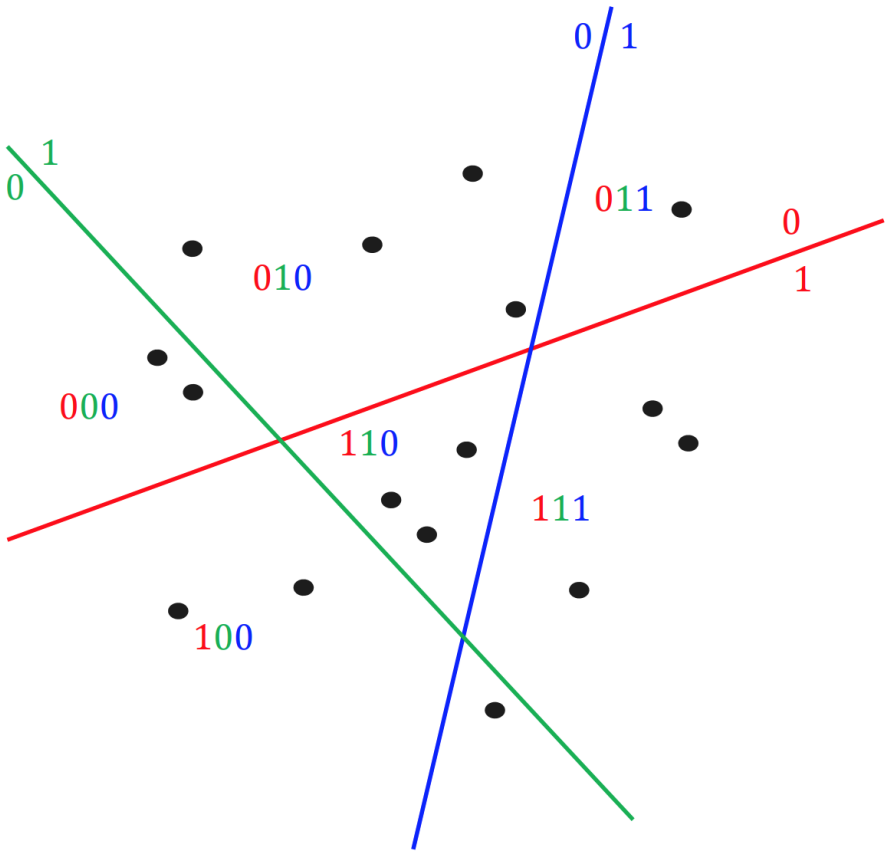
2. LSH (Locality Sensitive Hashing)



A typical hash function aims to place different values (no matter how similar) into separate buckets.



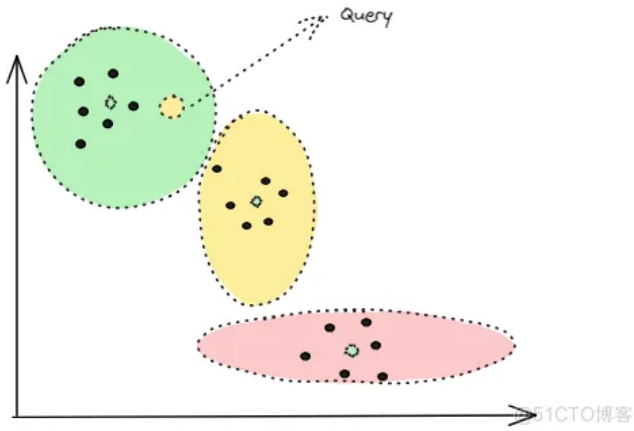
An LSH function aims to place similar values into the same buckets.



随机超平面投影LSH

3. IVF(Inverted File)

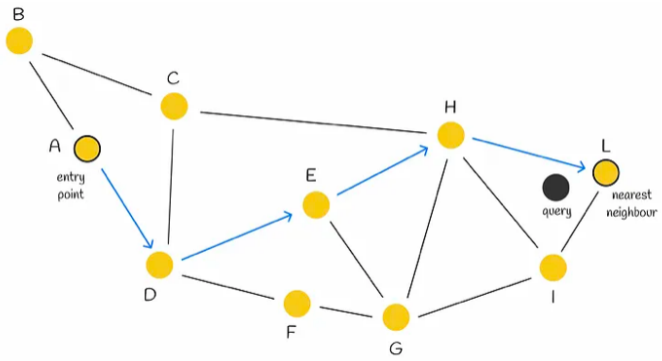
单词ID	单词	文档频率	倒排列表（DocID;TF;<POS>）
1	谷歌	5	(1;1;<1>), (2;1;<1>), (3;2;<1;6>), (4;1;<1>), (5;1;<1>)
2	地图	5	(1;1;<2>), (2;1;<2>), (3;1;<2>), (4;1;<2>), (5;1;<2>)
3	之父	4	<1;1;<3>), (2;1;<3>), (4;1;<3>), (5;1;<3>)
4	跳槽	2	(1;1;<4>), (4;1;<4>)
5	Facebook	5	(1;1;<5>), (2;1;<5>), (3;1;<8>), (4;1;<5>), (5;1;<8>)
6	加盟	3	(2;1;<4>), (3;1;<7>), (5;1;<5>)
7	创始人	1	(3;1;<3>)
8	拉斯	2	(3;1;<4>), (5;1;<4>)
9	离开	1	(3;1;<5>)
10	与	1	（4;1;<6>）



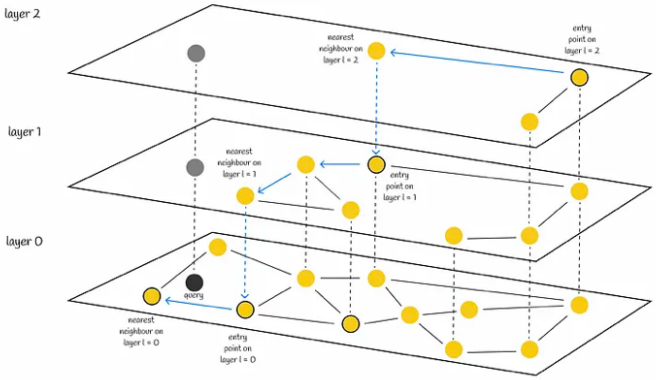
51CTO博客

- 对向量进行k-means聚类，找到聚类中心点
- 每个中心点作为倒排索引的key
- 计算查询向量与索引key之间的距离
- 选取距离最小的key1
- 从倒排索引中检索与key1对应的区域关联的向量

4. HNSW（Hierarchical Navigable Small World）



Greedy search process in a navigable small world. Node A is used as an entry point. It has two neighbours B and D. Node D is closer to the query than B. As a result, we move to D. Node D has three neighbours C, E and F. E is the closest neighbour to the query, so we move to E. Finally, the search process will lead to node L. Since all neighbours of L are located further from the query than L itself, we stop the algorithm and return L as the answer to the query.

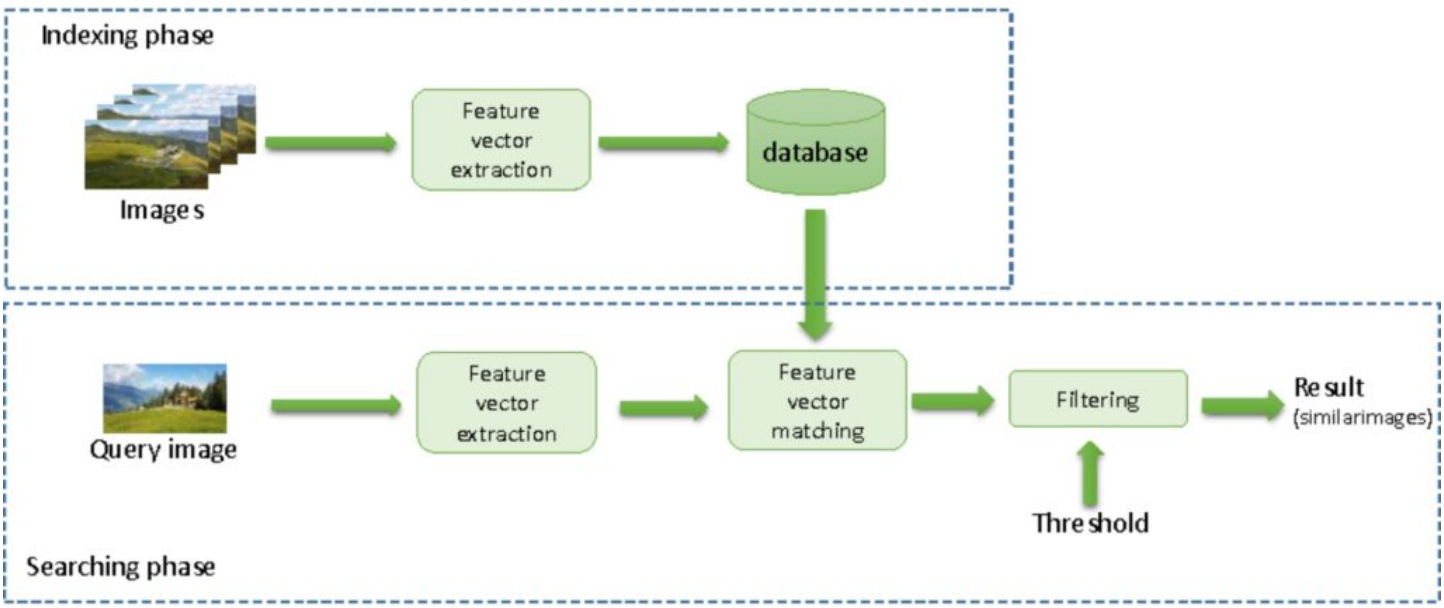


Search in HNSW

6. 图形界面

- **Streamlit**
- **Gradio**
  - 快速创建简单、交互式Web界面
  - 非常简洁的API，无需了解前端
  - 快速部署应用

7. 代码解析



General workflow for reverse image search

Talk is cheap. Show me the code.

a) 安装包

```
pip install gradio pymilvus torch
```

b) 数据准备

```
$ mkdir data
$ cd data
$ curl -L http://github.com/towhee-io/examples/releases/download/data/reverse_image_search.zip -O
$ unzip -q -o reverse_image_search.zip
```

c) 安装向量数据库

```
#安装单机版 Milvus
$ wget http://github.com/milvus-io/milvus/releases/download/v2.0.2/milvus-standalone-docker-compose.yml -O docker-compose.yml
$ docker-compose up -d
$ docker ps
```