[CSED261] Introduction to Data Analysis

Practice #3. Multimodal Search with Image and Text Through Vibe Coding

Abstract
This assignment focuses on developing a multimodal retrieval system with the diskann. The objective is to design a text-to-image and image-to-text search system and implement it in the form of a simple web-based application. Any AI tools may be utilized for "Vibe Coding" as part of this project.

Example demo video: demo-multimodal-retrieval.mov

---

1. Background

- **Multimodal Search**
  Multimodal search refers to retrieving information across different data types, such as text and images. For example, a user can input a text query (e.g., "a cat on a sofa") to retrieve relevant images.

- **Embeddings**
  To enable search across different modalities, data is first converted into *embeddings*. An embedding is a vector representation of data (e.g., text or image) in a high-dimensional space. Items with similar meanings or visual content will have embeddings that are closer together in this space.

- **Approximate Nearest Neighbor Search**
  Since the database may contain thousands of embeddings, a specialized algorithm is used to make retrieval efficient. In this assignment, you will use **DiskANN**.

- **CLIP Model**
  To generate embeddings for queries, we use the CLIP (Contrastive Language–Image Pretraining) model (`openai/clip-vit-base-patch32`). CLIP can map both images and texts into the same embedding space, making it possible to compare them directly. For similarity measurement, CLIP uses the inner product.

2. Deadlines & Deliverables

Code & Demo submission	10/19 23:59	Code: `{student_id}.zip` Demo video: Submit the link to the saved video (e.g., YouTube, Google Drive).
Report submission	10/26 23:59	`{student_id}.pdf`

- No late submissions will be accepted.

3. Problem Statements

3-1. Requirements

- Environments
  - Conda, Docker, etc.
- DiskANN setup
  - Before starting homework, you **must follow the instructions in `example/README.md`** to properly set up the environment and configure DiskANN.
  - This step is mandatory since DiskANN depends on the Milvus library. If the environment is not set up correctly, Milvus will not work.
- Embedding model
  - We use the Hugging Face model: **`openai/clip-vit-base-patch32`** for online query embeddings.
  - Reference: https://huggingface.co/openai/clip-vit-base-patch32

3-2. Provided Materials

Your project files and directories should follow the given structure:

```
data/
   ├── database/
   │      ├── images/
   │      ├── texts/
   │      ├── img_embeddings.npy
   │      └── txt_embeddings.npy
   ├── query/
   │      ├── images/
   │      └── texts/
example/
   └── milvus_diskann_example.py
   └── README.md
   └── start.sh
```

- Embeddings database (`.npy` file)
  - images database (`img_embeddings.npy`)
    - Shape: (5000, 512)
    - 5000: number of items, 512: embedding dimension
    - Each row corresponds to one embedding vector.
  - texts database (`txt_embeddings.npy`)
    - Shape: (25014, 512)
    - 25014: number of items, 512: embedding dimension
    - Each row corresponds to one embedding vector.
- Raw data files
  - Images: the embedding at index `i` corresponds to an image file named `"%06d.jpg" % (i)`
    - e.g., index 42 → 000042.jpg, index 1234 → 001234.jpg

    - Texts: the embedding at index `i` corresponds to a text file named `"%06d.txt" % (i)`
        - e.g., index 42 → 000042.txt, index 1234 → 001234.txt
- Query Data
    - image-to-text query (2 queries)
        - The image data will be provided as the files `animal.jpg`, `baseball.jpg`.


    - text-to-image query (3 queries)
        - *A man riding a bicycle on a city street.*
        - *A group of people eating dinner at a table.*
        - *A cat sleeping on a sofa.*
- Example
    - It contains DiskANN example codes.
    - You should follow the instructions in README.md before starting this homework.

3-3. Process

1. Load the provided `.npy` embedding file and construct a vector database
    - These files are pre-computed embedding data files (`img_embeddings.npy`, `txt_embeddings.npy`), which will serve as the database for retrieval.
    - Construct one image database and one text database from these files.
        - For text-to-image: construct an image database
        - For image-to-text: construct a text database
2. Process query from user
    - Queries for image-to-text and text-to-image are provided in the assignment (see 3-2.Provided Materials section).
    - Compute text or image embedding using the required embedding model.
3. Search with DiskANN
    - Obtain the indexes of the top-3 results per query.
        - For text-to-image: use an image database with a text query
        - For image-to-text: use a text database with an image query
    - Similarity between query and data is defined by the inner product.
4. Retrieve results
    - For text-to-image: return the `.jpg` file matching the index.
    - For image-to-text: return the `.txt` file matching the index.

5. **Demo implementation**
    - A simple web interface should allow users to input queries and view results.
        - The design of the web interface can be arbitrary, and it is acceptable to use either Flask, Streamlit, or anything for the implementation.
    - For each query, the system must display the corresponding **top-3 result**.
    - The demo video must show the system running **one query at a time in sequence**.
        - For each query run:
            1. Input: one query
            2. Output: corresponding top-3 results

4. Grade

- **Please strictly follow the instructions in the "3-3. Process" – especially "5. Demo implementation"**
- Demo evaluation [50 pts]
    - Correctness of retrieval
        - The retrieved results should not be too unrelated to the query.
        - E.g., a text query "a herd of zebras" should not return train images.
    - Scoring
        - Image-to-text: 2 queries, 10 pts each
        - Text-to-image: 3 queries, 10 pts each
        - If any **top-3 outputs** are missing, that query will receive no credit.
- Report [50 pts]
    - After the code & demo submission deadline, analyze your code implementation and query result.
        - Implementation analysis: Explain how you implemented the system
        - Query result analysis: Discuss the results obtained from the provided queries in terms of their relevance
    - Describe what you have learned through this homework.
    - If you use any AI tools, state what/where/how you used them.
        - E.g., prompts that you used