ADB Project | Final Submission Document

Indexing System & Algorithms Breakdown

Our Proposed Indexing system is IVFPQ (Inverted File with Product Quantization)

- **Mini-Batch K-Means:**

  In this step, we are either dealing with sub-clusters or primary clusters each is handled as follows:

  - **For Sub-Clusters:**
    - We save the centers and compute **Euclidean Distance** manually
  - **For Primary Clusters:**
    - We save the centers and compute **Cosine Similarity**

- **Product Quantization (PQ):**

  We use product quantization for compression

  - Compress the array dimension from the initial dimension 70 to a lower dimension (the sub-vectors range from 5-14 based on the DB size).

How does the algorithm work?

The algorithm consists of two main steps: Training and Retrieving each is explained below as follows:

- **Training phase:**
  1. We cluster the database into a predefined number of primary clusters and assign each tuple in the database to the nearest cluster within these clusters
  2. Then, we divide each tuple into sub-vectors, and within each cluster, we train ***2**n*** bits sub-clusters for all the sub-vectors resulting from the division

- **Retrieval Phase:**
  1. We use **Cosine Similarity** to retrieve the nearest ***n*** clusters to the query vector,
  2. Within each cluster, we search for the ***m*** nearest neighbors using **Euclidean Distance** and compressed vectors(**code-words**)
  3. We filter the resulting **code-words** using **Cosine Similarity** with the original vectors to get the ***nearest k-vectors***.

Describe the detailed steps your system will perform to:

Build the index for the vector column in the database of 20 million records:

- For a 20-million record database, we use 280 clusters to assign the tuples. For each tuple using PQ, we will compress to 7 sub-vectors
- We will use **MiniBatchKMeans** to calculate the ***cluster_centers***, with a ***batch_size = 10000***
- For each cluster, we will compute the sub-vector_estimators using **MiniBatchKMeans** with ***batch_size=1000*** and the ***number of clusters=256 (2**8 bits)***
- Then, we will train the sub_clusters using the vectors of the cluster, and the sub-vector_estimators computed earlier
- Finally, we get the centers of the ***sub-vector_estimators*** and add the ***code-words*** of each index
- After fitting, our final index folder consists of ***cluster_centers***, ***sub_cluster_centers***, ***the indices, and the code-words of each cluster***

Retrieve the nearest vectors to a query vector from the same database.

- First, we will read the ***cluster_centers*** from our index folder, and using **Cosine Similarity** we will get the nearest ***37*** clusters to the query given

- We will load the indices and code_words for the nearest clusters. Then, we search to get the nearest ***200*** neighbors in each ***sub_cluster*** using **Euclidean Distance**
- We compute the new **Cosine Similarities** with the **nearest vectors** to get the nearest ***10*** vectors to the query vector.

Summary of Dot Product Operations

1. **Retrieving Nearest Vectors**:
    - Finding nearest clusters: ***280***
    - Refining search: **37** (**Nearest clusters**) * **200** (**Nearest vectors**) = ***7400***
    - **Total**:

      280 + 7400 = 7680

Our Algorithm Results Using Different *Query_Seeds*

- **Query Seed = 256753**

DB Size	Accuracy	Time	Ram
1M	0.0	0.22	0.12
10M	0.0	2.24	0.38
15M	0.0	2.78	0.00
20M	0.0	9.24	0.12

- **Query Seed = 123**

DB Size	Accuracy	Time	Ram
1M	0.0	0.20	0.00
10M	0.0	2.11	0.25
15M	0.0	2.64	0.23
20M	0.0	6.21	0.25

- **Query Seed = 10**

DB Size	Accuracy	Time	Ram
1M	0.0	0.20	0.00
10M	0.0	2.11	0.50
15M	0.0	2.52	0.62
20M	0.0	8.55	0.00

- **Query Seed = 4**

DB Size	Accuracy	Time	Ram
1M	0.0	0.20	0.25
10M	0.0	2.14	0.25
15M	0.0	2.50	0.24
20M	0.0	5.35	0.25

- **Query Seed = 29**

DB Size	Accuracy	Time	Ram
1M	0.0	0.23	0.38
10M	0.0	2.39	0.36
15M	0.0	2.89	0.25
20M	0.0	7.85	0.00