Faculty of Engineering
Computer Engineering Department

Phase 2: Project Document

Advanced Database Systems

Semester Project

Presented by

Name	Section	BN	ID
Mohammad Yehia Alahmad	2	13	9213024
Moustafa Mohammed Elsayed	2	23	9211194
Mostafa Hani Mostafa	2	24	9211206
Mennatallah Ahmed Moustafa	2	25	9211230

Development Stages

Initial Approach: IVF

At first, we tried the Inverted File Index algorithm (IVF) with Kmeans clustering to partition the vector space. We tested the 1 million databases with k = 256 and obtained the following results:

N_probe = 1	Team Number 14 1M score -235.0 time 0.48 RAM 16.21 MB
N_probe = 2	Team Number 14 1M score -230.33333333333334 time 0.91 RAM 8.89 MB
N_probe = 3	Team Number 14 1M score -207.66666666666666 time 1.32 RAM 9.34 MB
N_probe = 4	Team Number 14 1M score -155.33333333333334 time 1.75 RAM 8.48 MB
N_probe = 5	Team Number 14 1M score -104.33333333333333 time 2.18 RAM 10.22 MB

From this result we observed that increasing **n_probe** improved accuracy but led to higher RAM and longer retrieval times.

Also, we noticed that **Index building time** was very high due to standard Kmeans clustering algorithm

As a result, we replaced **Kmeans** with **MiniBatchKMeans** to reduce index-building time while maintaining similar accuracy. This enabled us to experiment with more parameters.

But the results weren’t that good specifically with the large data sizes, so we tried other approaches.

Second Approach: Multi-Level IVF

A multi-level IVF structure improves indexing efficiency by hierarchical clustering. At the first level, the vector space is partitioned using standard IVF techniques.

Each resulting cluster is then subdivided at the second level using MiniBatchKMeans, with the number of second-level clusters being determined by the square root of the number of vectors in the cluster.

This approach balances between reduced memory usage and higher retrieval accuracy, while maintaining scalable query performance.
However the way for reading vectors from disk differs a lot in time and Ram these are the ways we tried

- Reading one row by one row

```
Team Number 14
1M      score   -135.0  time    2.66    RAM     0.01 MB
10M     score   -269.3333333333333      time    5.93    RAM     0.01 MB
15M     score   -302.6666666666667      time    15.51   RAM     0.01 MB
```

  As we can notice that time is very high but ram is very good

- Reading whole cluster vectors: this made the time decrease but the RAM is very high, So we decided to read the cluster in batches to solve the tradeoff between the time and RAM.

Finally we adjusted each of

1. Batch size in Retrieving of vectors
2. Number of clusters per each size
3. Number of probes in each level

All of these parameters took a long time to reach the best combination that achieve good results with different query seeds and different data sizes
We would like to note that sometimes we had tradeoff between getting score very close to zero and have higher time but we decided to make both time and score with convenient values

Adjusted Parameters:

- number of clusters:

  1 M → 256

  10 M → 5000

  15 M → 4400

  20 M → 10000

- number of probes in level 1:
  1M → 50

10M → 80
15M → 78
20M → 80

- number of probes in level 2:

  1M → 10
  10M → 30
  15M → 40
  20M → 40

Question 2 Answer

First: Build

1. Perform Minibatch K means on the whole database with K = 8000

2. train the kmeans with 1 million records only to speed up building the index

3. predict all the vectors and assign it to clusters

4. save the level 1 index file

5. For each cluster in level 1:

   1- get all the vector that reside in that cluster and perform minibatchKmeans with #clusters = sqrt (length of vectors in the cluster)

   2- do the same for all clusters

   3- Save the level 2 clusters

Second: Retrieval

1. level 1 n_probe = 80

calculate distance between each cluster in level 1 with the query: get the nearest 80 clusters

2. for each cluster of the 80:

   - get all the cluster centers inside that cluster

   - calculate distance between each small cluster with the vector

   - get the 40 small clusters that are closer to the vector - if there are less that 40 clusters inside the big cluster take all of them

   - get each small cluster in batches (500 vectors at a time)

   - put it in a priority queue(heap)

   - return the top_k_heap (list of nearest vectors)

Dot Product Calculation

Number of first level clusters = 10000

Average number of second level clusters = root (20,000,000 / 10,000) = 45

First Level n_probe = 80

Second Level n_probe = 40

Average number of vectors in second level cluster = 20,000,000 / (45*10000) = 45

Total number of dot products = Number of first level clusters + Average number of second level clusters * First Level n_probe + Average number of vectors in second level cluster * First Level n_probe * Second Level n_probe

Total number of dot products = 10000 + 45 * 80 + 80 * 45 * 40=157600

Sample Queries

Seed=10

```
Team Number 14
1M      score   -15.0   time    0.77    RAM     6.38 MB
10M     score   -24.0   time    4.09    RAM     9.71 MB
15M     score   -30.333333333333332     time    5.50    RAM     7.74 MB
20M     score   -38.666666666666664     time    6.58    RAM     8.33 MB
```

Seed=0

```
Team Number 14
1M      score   -17.666666666666668     time    0.79    RAM     6.61 MB
10M     score   -34.0   time    3.97    RAM     8.22 MB
15M     score   -12.666666666666666     time    5.32    RAM     10.58 MB
20M     score   -8.666666666666666      time    6.73    RAM     7.95 MB
```

Seed=15

```
Team Number 14
1M      score   -19.333333333333332     time    0.73    RAM     6.19 MB
10M     score   -7.666666666666667      time    3.88    RAM     6.25 MB
15M     score   -26.0   time    5.01    RAM     6.56 MB
20M     score   -20.333333333333332     time    6.26    RAM     6.44 MB
```

Seed=39

```
Team Number 14
1M      score   -14.333333333333334     time    0.76    RAM     6.31 MB
10M     score   -5.333333333333333      time    4.03    RAM     6.38 MB
15M     score   -24.333333333333332     time    5.18    RAM     6.57 MB
20M     score   0.0     time    6.26    RAM     9.05 MB
```

Seed=40

```
Team Number 14
1M      score   0.0     time    0.75    RAM     6.24 MB
10M     score   -24.666666666666668     time    3.89    RAM     6.25 MB
15M     score   -7.0    time    6.47    RAM     6.44 MB
20M     score   -40.0   time    6.56    RAM     6.36 MB
```

Seed=4

```
Team Number 14
1M      score   -22.666666666666668     time    0.75    RAM     6.12 MB
10M     score   -28.333333333333332     time    4.20    RAM     6.55 MB
15M     score   -39.666666666666664     time    5.84    RAM     5.39 MB
20M     score   -17.666666666666668     time    6.19    RAM     6.44 MB
```

Seed=50

```
Team Number 14
1M      score   -17.333333333333332     time    0.73    RAM     6.31 MB
10M     score   -15.0   time    4.07    RAM     6.62 MB
15M     score   -19.0   time    5.31    RAM     6.41 MB
20M     score   -48.333333333333336     time    6.15    RAM     5.59 MB
```

Seed=70

```
Team Number 14
1M      score   -8.333333333333334      time    0.73    RAM     6.31 MB
10M     score   -26.0   time    4.11    RAM     6.38 MB
15M     score   -36.666666666666664     time    5.66    RAM     6.36 MB
20M     score   -34.0   time    6.16    RAM     6.47 MB
```

Seed=100

```
Team Number 14
1M      score   0.0     time    0.73    RAM     6.21 MB
10M     score   -16.0   time    4.08    RAM     6.50 MB
15M     score   -57.0   time    5.10    RAM     6.44 MB
20M     score   -20.0   time    5.97    RAM     6.44 MB
```

Seed=20

```
Team Number 14
1M      score   -16.333333333333332     time    0.74    RAM     6.25 MB
10M     score   -25.666666666666668     time    3.97    RAM     6.34 MB
15M     score   -107.0  time    5.00    RAM     6.31 MB
20M     score   -40.0   time    5.95    RAM     6.34 MB
```

Seed=285616

```
Team Number 14
1M      score   -7.333333333333333      time    0.75    RAM     6.31 MB
10M     score   -40.0   time    3.87    RAM     6.44 MB
15M     score   -14.666666666666666     time    6.61    RAM     6.84 MB
20M     score   -40.333333333333336     time    5.98    RAM     6.50 MB
```