ACM SIGMOD Programming Contest 2023

Team: CantonDwenDwen (Runner-Up) **Advisor:** Dr. Jilian Zhang

Haizhou Ye yeh629@stu2021.jnu.edu.cn

Xuyang Liu liuxuyang@stu2022.jnu.edu.cn

Chenzhao Wang akinouta@stu2022.jnu.edu.cn

Task Overview

Task: to build an **approximate K-NN Graph** for a set of vectors. i.e., for each vector, find its approximate k nearest neighbors in a limited time (30 minutes).

Dataset	Description	Size	K
X	Bing queries encoded by Turing AGI v5	10^7	100

Evaluation Metric: $\text{Recall} = \frac{\text{number of true top 100 nearest neighbors}}{100}$

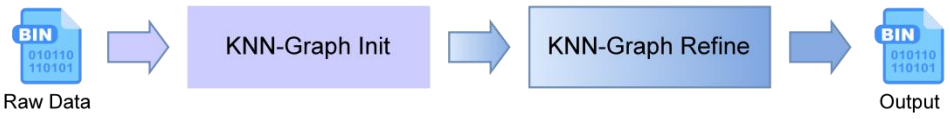
Evaluation Environment: 32 CPU x 2.7 GHz, 64 GB Main Memory, 32 GB Storage, Ubuntu 20.04.5 LTS - **no GPU**

Solution Overview

We first use the **faiss** library [1] to obtain an initial KNN graph, and then use the **pynndescent** library [2] to refine it.

Our solution involved 2 major steps:

1. **Construct an initial KNN graph.**
2. **Refine the initial KNN graph.**



KNN Graph Initialization

Goal: A good trade-off between recall and runtime

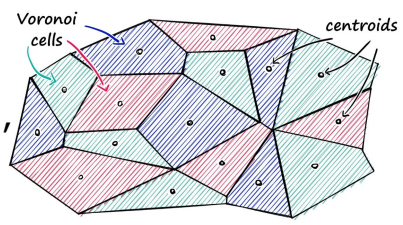
step:

1. **Build:** index=faiss.index_factory(……)
2. **Train:** index.train(X)
3. **Add:** index.add(X)
4. **Search:** D,I=index.search(X, k=340)

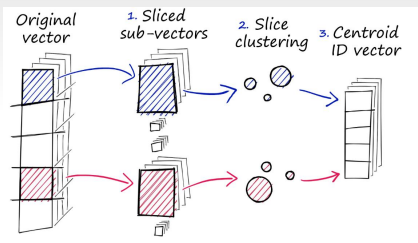
Parameter:

- index_string="IVF1100,PQ100x4fsr,RFlat"
- nprobe=77
- thread=16

• To partition the index into Voronoi cells is a popular approach, which reduces search space of our solution, and produces an approximate answer. (IVF)



• Product Quantization (PQ) [3] is a key optimization technique used, which compresses input vectors and approximates the distance/similarity calculation.



• Specifically, we use PQFastScan (PQfs) [4], which stores the search-time look-up tables in registers.

• In particular, we search for 340NN instead of 100NN. This is because searching more nearest neighbors helps to exploit reverse neighbors. For example, v_1 is one of the current 100-NN of v_0, then v_0 may be one of the true 100-NN of v_1.

KNN Graph Refinement

Simple Principle:
A neighbor of a neighbor is also likely to be a neighbor

step: index = NNDescent(X, ……)

Parameter:

- metric="sqeuclidean"
- thread=16
- init_dist=D
- init_graph=I
- n_neighbors=101
- n_iters=1
- max_candidates=151

Optimization for Pynndescent:

1. **Reverse neighbor utilization:**
   The initial graph generated by faiss is 340NN, and the current nearest neighbor is updated according to the reverse neighbor.
2. **Max heap building:**
   The initial graph constructed by faiss has been sorted by distance and has satisfied the min heap property.
3. **Multithreaded code optimization:**
   Distribute tasks more evenly to each thread.

Time≈9m31s, Recall≈0.980 → (After Optimization) → Time≈**4m18s**, Recall≈**0.984**

Results

#	Graph Size	Recall	Runtime
Initial Graph	**340**-NN	**0.965**	**24**m**9**s
Refined Graph	**100**-NN	**0.984**	**4**m**18**s

Conclusion

1. The technical route of initializing the graph first and then refining it is fast and efficient.

2. The re-ranking of faiss and the searching of 340-NN make many reverse neighbors can be exploited to improve performance.

3. We achieve a good trade-off in the time distribution of initialization and refinement (24~25m on initializing and 4~5m on Refining). The iterative one-round NNDescent algorithm has the highest cost performance.

References:

[1] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. IEEE Trans. Big Data, 7(3):535--547, 2021.

[2] Wei Dong, Charikar Moses, and Kai Li. 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. In WWW. 577--586.

[3] Jegou H, Douze M, Schmid C. Product quantization for nearest neighbor search[J]. IEEE transactions on pattern analysis and machine intelligence, 2010, 33(1): 117-128.

[4] Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. 2015. Cache locality is not enough: High-Performance Nearest Neighbor Search with Product Quantization Fast Scan. Proceedings of the VLDB Endowment (PVLDB) 9, 4 (2015), 288--299.