

Milvus Paper sharing

Revisiting the Inverted Indices for Billion-Scale Approximate
Nearest Neighbors

godchen fishpenguin

Content

1.Index vs Multi-Index

2.Grouping and pruning

3.Experiments

Index vs Multi-Index

Structure	Inverted Index	Inverted Multi-Index
Candidate lists quality	Medium	High
Query assignment & indexing cost	Medium	Low
Number of random memory accesses during search	Small	Large
Performance increase from large K	High	Small
Memory consumption scalability	$O(K)$	$O(K^2)$

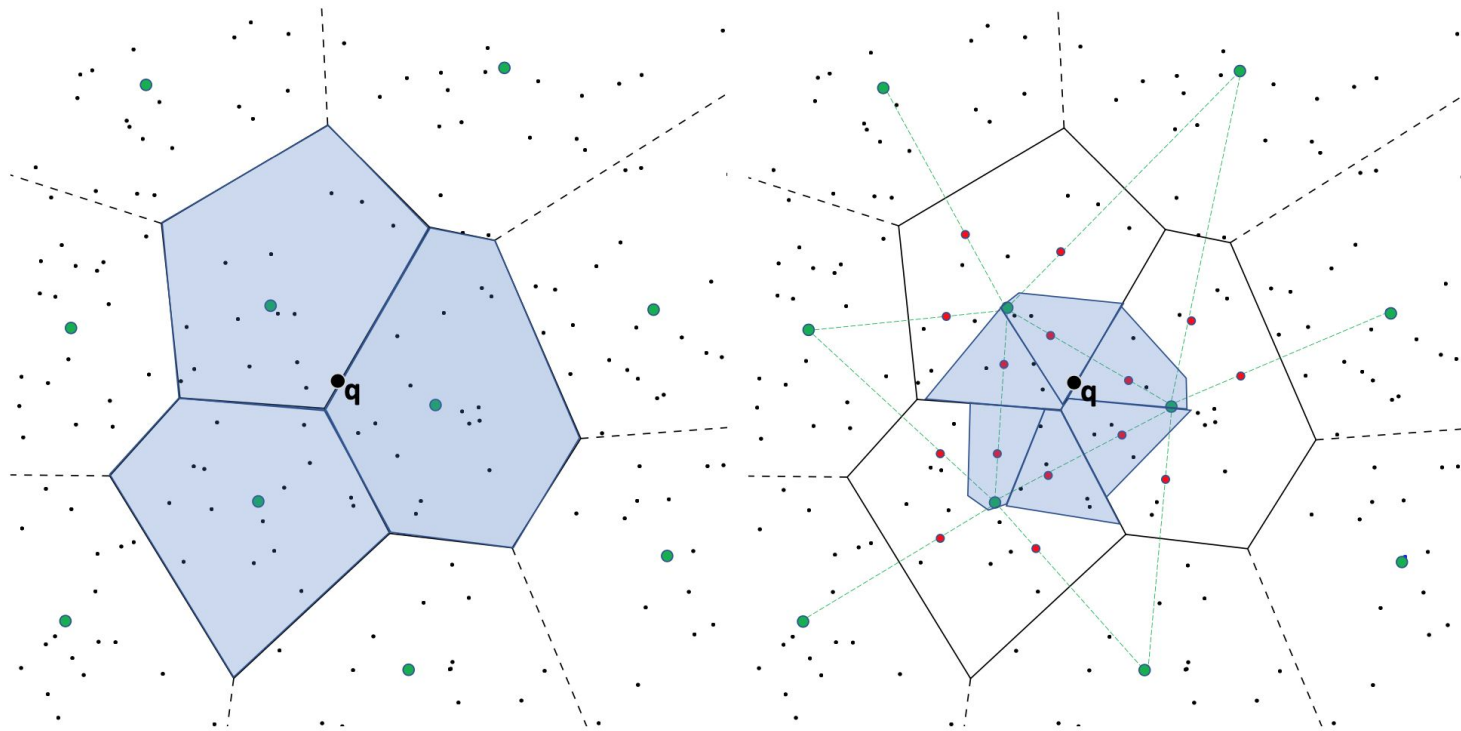
Table 1. Comparison of the main properties of the inverted index and the IMI. K denotes the codebook sizes in both systems. The IMI provides more precise candidate lists and has low indexing and query assignment costs due to smaller codebook sizes. On the other hand, the inverted index requires a smaller number of expensive random memory accesses when searching, and could benefit from large codebooks, while the IMI performance saturates with K about 2^{14} . Moreover, the increase of K is memory-inefficient in the IMI as its additional memory consumption scales quadratically.

Index vs Multi-Index

Inverted Index			Inverted Multi-Index		
K	Average distance	Memory	K	Average distance	Memory
2^{18}	0.315	97Mb	2^{13}	0.345	256Mb
2^{20}	0.282	388Mb	2^{14}	0.321	1024Mb
2^{22}	0.259	1552Mb	2^{15}	0.305	4096Mb

Table 2. The indexing quality and the amount of additional memory consumption for the inverted index and the IMI with different codebook sizes on the DEEP1B dataset. The indexing quality is evaluated by the average distance from the datapoints to the closest region centroid. The IMI indexing quality does not benefit from $K > 2^{14}$ while the required memory grows quadratically.

Grouping and pruning



Grouping

1. dataset points: $\{x_1, \dots, x_n\}$

2. assume centroid c

3. let $\{s_1, \dots, s_L\} = \text{NN}_L(c)$, L denote nearest neighbors for c

4. closest subcentroid, $l_i = \text{argmin} \|x_i - (c + \alpha(s_i - c))\|^2$, α is a scalar parameters

5. displacement $x_i - (c + \alpha(s_i - c))$

6. distance estimation: $\|q - c - \alpha(s - c) - [r_1, \dots, r_M]\|^2$

Grouping

$$\begin{aligned} \|q - c - \alpha(s - c) - [r_1, \dots, r_M]\|^2 &= (1 - \alpha)\|q - c\|^2 + \\ &+ \alpha\|q - s\|^2 - 2 \sum_{m=1}^M \langle q_m, r_m \rangle + \text{const}(q) \end{aligned}$$

1. $\|q - c\|^2$ are computed online before visiting the region points.
2. $\|q - s\|^2$ is same as $\|q - c\|^2$
3. The scalar products between the query subvectors and PQ codewords $\langle q_m, r_m \rangle$ are precomputed before regions traversal.
4. The last term is query-independent, and we quantize it into 256 values and explicitly keep its quantized value as an additional byte in the point code.

pruning

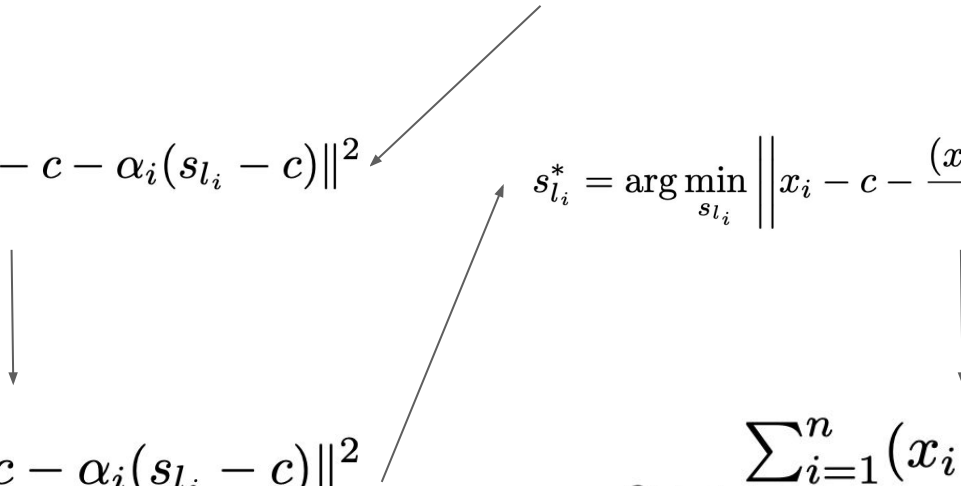
$$\|q - c - \alpha(s_l - c)\|^2 = (1 - \alpha)\|q - c\|^2 + \alpha\|q - s_l\|^2 + \text{const}(q), \quad l = 1 \dots L$$

pruning is a way to skip subregions

- 1.If the search process is set to visit k inverted index regions, total subregions is kL.
- 2.only a certain fraction τ of the closest subregions is visited.

Learning the scaling factor α

destination function: $\min_{\alpha \in [0;1]} \sum_{i=1}^n \min_{l_i} \|x_i - c - \alpha(s_{l_i} - c)\|^2$

$$\sum_{i=1}^n \min_{l_i, \alpha_i \in [0;1]} \|x_i - c - \alpha_i(s_{l_i} - c)\|^2$$

$$s_{l_i}^* = \arg \min_{s_{l_i}} \left\| x_i - c - \frac{(x_i - c)^T (s_{l_i} - c)}{\|s_{l_i} - c\|^2} (s_{l_i} - c) \right\|^2$$

$$\min_{\alpha_i \in [0;1], s_{l_i}} \|x_i - c - \alpha_i(s_{l_i} - c)\|^2$$

$$\alpha = \frac{\sum_{i=1}^n (x_i - c)^T (s_{l_i}^* - c)}{\sum_{i=1}^n \|s_{l_i}^* - c\|^2}$$

Experiments

1. DEEP1B dataset[5] contains one billion of 96-dimensional CNN-produced feature vectors of the natural images from the Web. The dataset also contains a learning set of 350 million descriptors and 10,000 queries with the groundtruth nearest neighbors for evaluation.

2. SIFT1B dataset[1] contains one billion of 128-dimensional SIFT descriptors as a base set, a hold-out learning set of 100 million vectors, and 10,000 query vectors with the precomputed groundtruth nearest neighbors.

Attention: Indexing quality of the inverted index does not saturate even with codebooks of several million centroids. As the exhaustive query assignment would be inefficient for large codebooks, we use the approximate nearest centroids search via HNSW algorithm

Experiments

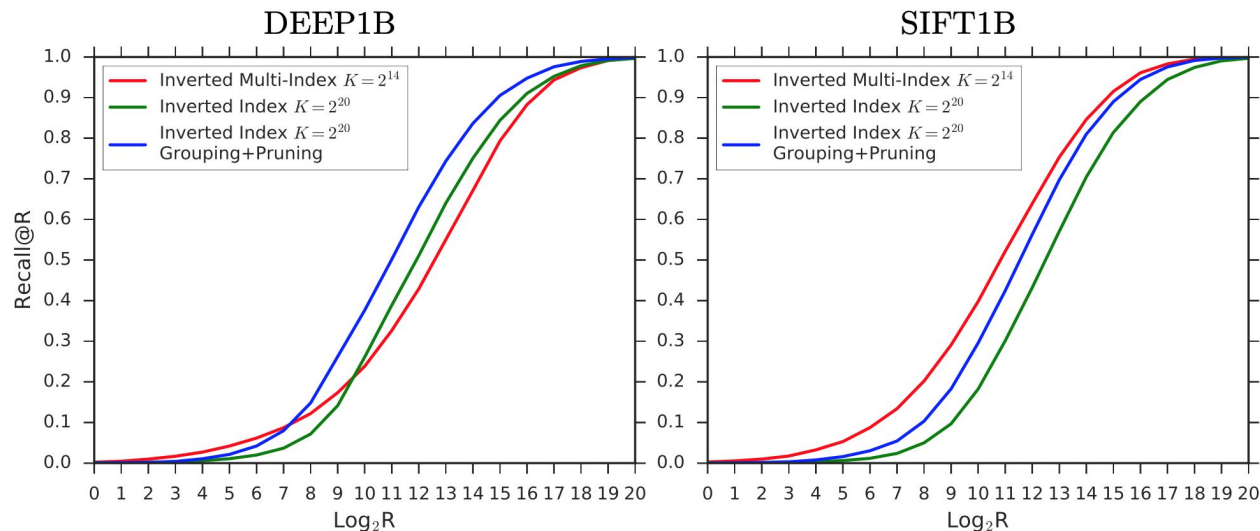
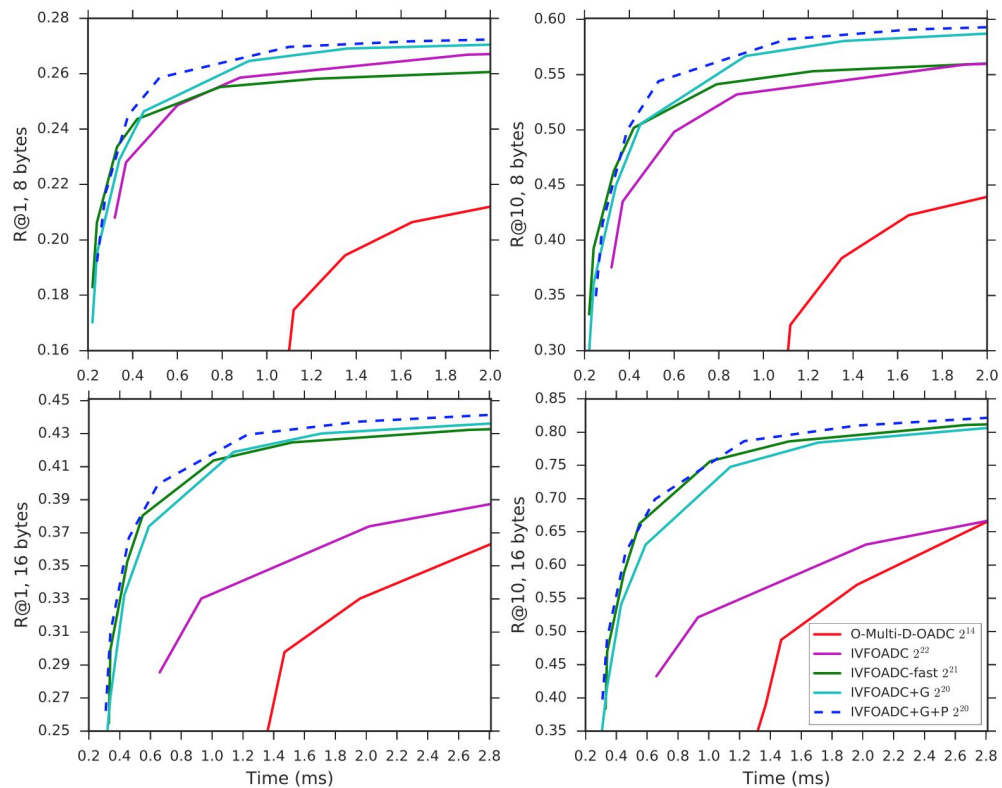


Fig. 2. Recall as a function of the candidate list length for inverted multi-indices with $K=2^{14}$, inverted index with $K=2^{20}$ with and without pruning. On DEEP1B the inverted indices outperform the IMI for all reasonable values of R by a large margin. For SIFT1B the candidate lists quality of the inverted index with pruning is comparable to the quality of the IMI for R larger than 2^{13} .

Experiments



Experiments

L	R@1	R@10	R@100	$t(ms)$		Inverted Index	Inverted Multi-Index
32	0.417	0.776	0.869	1.22	No grouping	0.282	0.415
64	0.433	0.785	0.878	1.28	With grouping	0.255	0.385
128	0.441	0.791	0.882	1.48	Decrease	10%	7%

Table 3. *Left:* The recall values and the runtimes of the IV-FOADC+Grouping+Pruning system for different numbers of subcentroids per region on the DEEP1B dataset. Here we use the candidate lists of length $30K$ and 16-byte codes. *Right:* The average distances from the datapoints to the closest (sub-)centroids with and without grouping for the inverted index with $K = 2^{20}$ and the IMI with $K = 2^{10}$ on the DEEP1B dataset.

Experiments

		DEEP1B					SIFT1B				
Method	K	R@1	R@10	R@100	t	Mem	R@1	R@10	R@100	t	Mem
O-Multi-D-OADC[24]	2^{14}	0.397	0.766	0.909	8.5	17.34	0.360	0.792	0.901	5	17.34
Multi-LOPQ[4]	2^{14}	0.41	0.79	-	20	18.68	0.454	0.862	0.908	19	19.22
GNOIMI[5]	2^{14}	0.45	0.81	-	20	19.75	-	-	-	-	-
IVFOADC+G+P	2^{20}	0.452	0.832	0.947	3.3	17.87	0.405	0.851	0.957	3.5	18

Table 4. Comparison to the previous works for 16-byte codes. The search runtimes are reported in milliseconds. We also provide the memory per point required by the retrieval systems (the numbers are in bytes and do not include 4 bytes for point ids).

Thanks