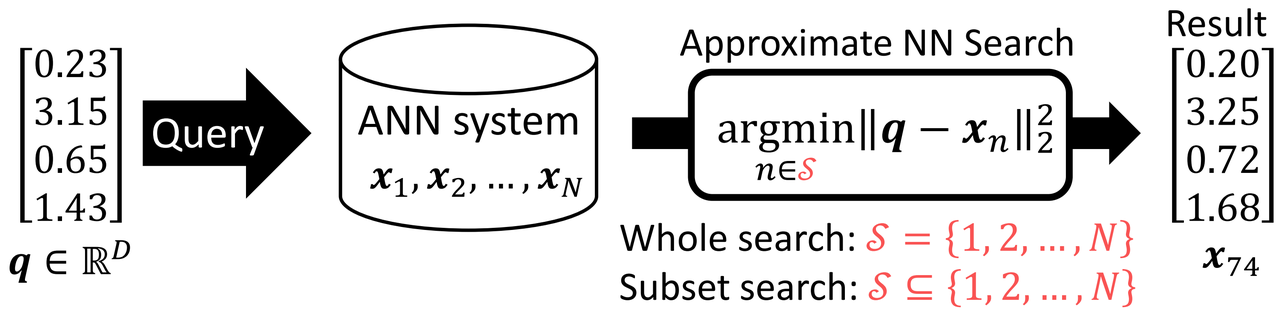
Reconfigurable Inverted Index

Yusuke Matsui (NII) Ryota Hinami (UTokyo) Shin'ichi Satoh (NII)

Summary

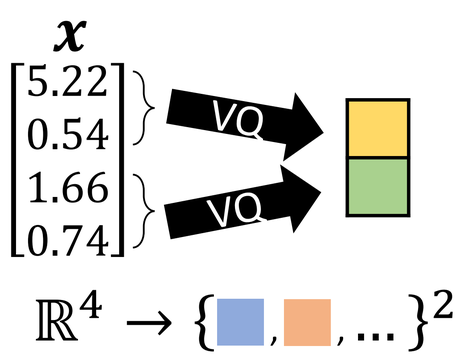
- Approximate nearest neighbor search
- Solved a **subset-search problem**
- Comparative performance with IVFPQ (Faiss); 10 ms for $N = 10^9$

Subset search problem

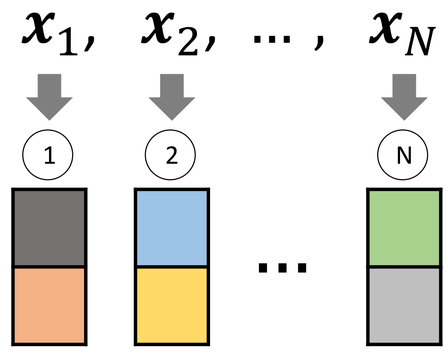


Preliminary 1: Product Quantization (PQ) linear scan [Jégou+, TPAMI 11]

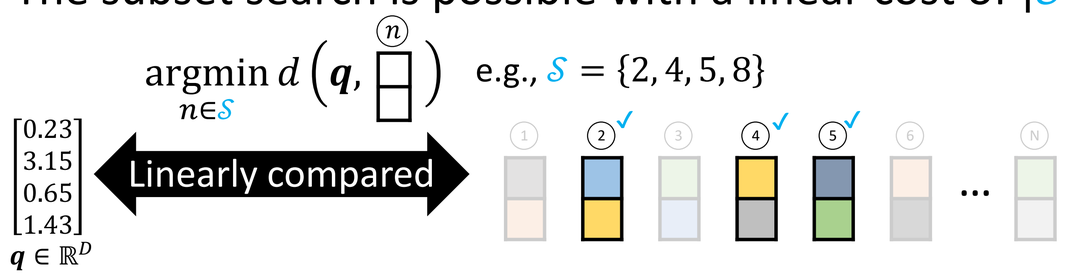
- Compress a vector into a short code



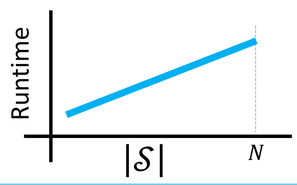
- All database vectors are PQ-encoded beforehand



- The subset search is possible with a linear cost of $|\mathcal{S}|$

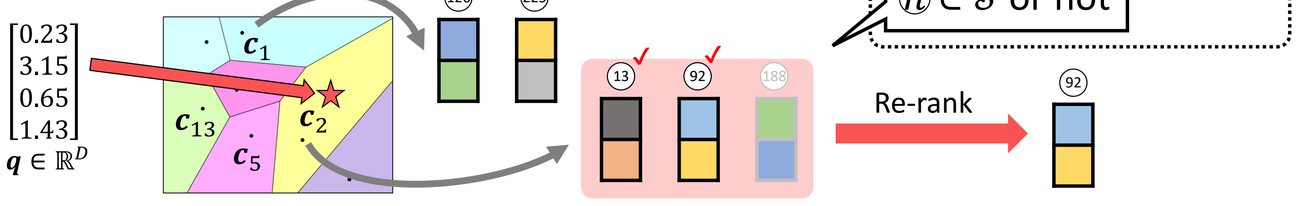


- The search is efficient only if $|\mathcal{S}|$ is small



Preliminary 2: Inverted Index + PQ (IVFPQ) [Jégou+, TPAMI 11]

- Inverted index over PQ-encoded vectors

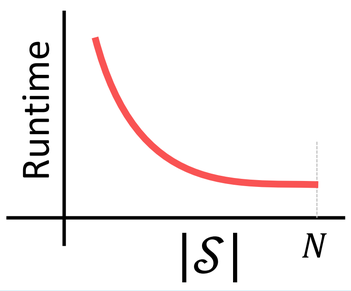


1. Find the closest space: $k^* = \operatorname{argmin}_k \|q - c_k\|_2^2$
2. Focus the k^*th space, filter out items $\notin \mathcal{S}$
3. Re-rank the items via PQ-linear scan

- Subset-search is possible only if $|\mathcal{S}|$ is large

Why is it slow for small $|\mathcal{S}|$?

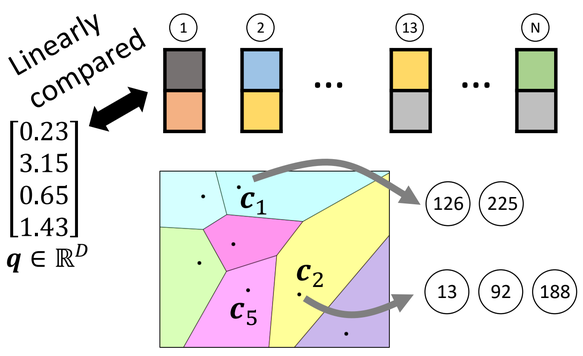
e.g., if $|\mathcal{S}|$ is small and they are far away from the query, we might need to scan all items



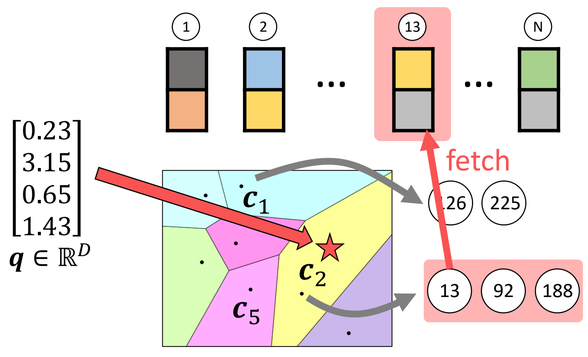
Proposed: Reconfigurable inverted index (Rii)

- Store (1) PQ-codes linearly, and (2) IDs as an inverted index
- Can run either PQ-linear-scan or IVFPQ with a single data structure

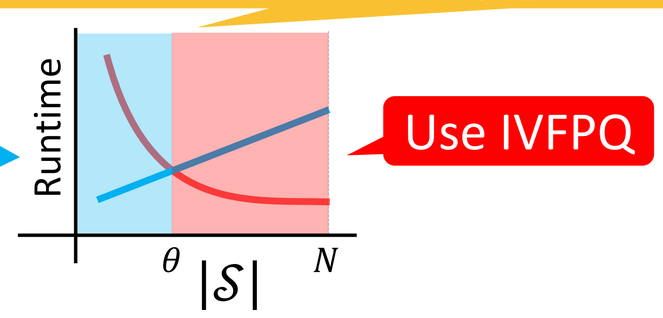
- If $|\mathcal{S}|$ is small, run PQ-linear scan



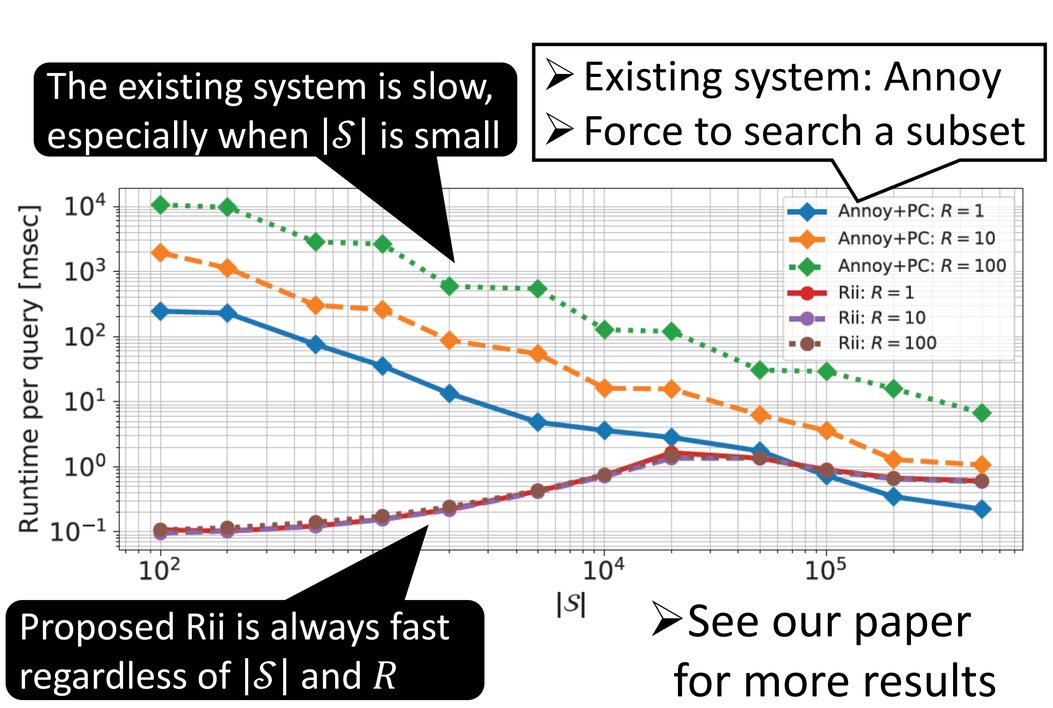
- If $|\mathcal{S}|$ is large, run IVFPQ



- Set a threshold θ
- Key: Switch two methods based on $|\mathcal{S}| \lessgtr \theta$



Evaluation (SIFT1M, top-R search)



- See our paper for more results

Codes: https://github.com/matsui528/rii

```
$ pip install rii
```

```
import rii
import nanopq

# Prepare a PQ/OPQ codec with M=32 sub spaces
codec = nanopq.PQ(M=32).fit(vecs=Xt)  # Trained using Xt

# Instantiate a Rii class with the codec
e = rii.Rii(fine_quantizer=codec)

# Add vectors
e.add_configure(vecs=X)

# Search
ids, dists = e.query(q=q, topk=3, target_ids=S)
print(ids, dists)  # e.g., [7484 8173 1556] [15.0 15.3 16.1]
```