Similarity Search by Aggregation of Multiple Pivot-Permutation Ranking

David Novak, Pavel Zezula

Masaryk University
Brno, Czech Republic

CEMI meeting, April 16, 2014

Outline of the Talk

1. Approximate Distance-based Similarity Search
2. PPP-Codes Approach
   - Multiple Pivot Space Partitioning
   - Ranking of the Data Objects
   - Indexing and Searching
3. Efficiency of our Approach
4. Summary

Preliminaries

- generic similarity search
- data modeled as metric space $(\mathcal{D}, \delta)$, where $\mathcal{D}$ is a *domain* of objects and δ is a total *distance function* $\delta : \mathcal{D} \times \mathcal{D} \longrightarrow \mathbb{R}_0^+$ satisfying postulates of identity, symmetry, and triangle inequality

Preliminaries

- generic similarity search
- data modeled as metric space $(\mathcal{D}, \delta)$, where $\mathcal{D}$ is a *domain* of objects and δ is a total *distance function* $\delta : \mathcal{D} \times \mathcal{D} \longrightarrow \mathbb{R}_0^+$ satisfying postulates of identity, symmetry, and triangle inequality
- query by example: K-NN(q) returns K objects x from the dataset $\mathcal{X} \subseteq \mathcal{D}$ with the smallest $\delta(q, x)$

Preliminaries

- generic similarity search
- data modeled as metric space $(\mathcal{D}, \delta)$, where $\mathcal{D}$ is a *domain* of objects and δ is a total *distance function* $\delta : \mathcal{D} \times \mathcal{D} \longrightarrow \mathbb{R}_0^+$ satisfying postulates of identity, symmetry, and triangle inequality

- query by example: K-NN(q) returns K objects x from the dataset $\mathcal{X} \subseteq \mathcal{D}$ with the smallest $\delta(q, x)$

- dataset $\mathcal{X}$ may be very large
- distance function δ may be time consuming

Preliminaries

- generic similarity search
- data modeled as metric space $(\mathcal{D}, \delta)$, where $\mathcal{D}$ is a *domain* of objects and δ is a total *distance function* $\delta : \mathcal{D} \times \mathcal{D} \longrightarrow \mathbb{R}_0^+$ satisfying postulates of identity, symmetry, and triangle inequality
- query by example: K-NN(q) returns K objects x from the dataset $\mathcal{X} \subseteq \mathcal{D}$ with the smallest $\delta(q, x)$
- dataset $\mathcal{X}$ may be very large
- distance function δ may be time consuming
- requires approximate search

Motivation

current indexes for large-scale approximate search:

- dataset $\mathcal{X}$ is partitioned
- given query q, the "most-promising" partitions form the candidate set
- the candidate set S_C is refined by calculating $\delta(q, x)$, $\forall x \in S_C$

Motivation

current indexes for large-scale approximate search:

- dataset $\mathcal{X}$ is partitioned
- given query q, the "most-promising" partitions form the candidate set
- the candidate set S_C is refined by calculating $\delta(q, x)$, $\forall x \in S_C$

reading and refinement of S_C form majority of the search costs

- accuracy of the candidate set is key

Our Approach in a Nutshell

1. data space is partitioned multiple-times independently
   - each partitioning is defined by one pivot space

Our Approach in a Nutshell

1. data space is partitioned multiple-times independently
   - each partitioning is defined by one pivot space
2. given query q, multiple ranked candidate sets are generated

Our Approach in a Nutshell

1. data space is partitioned multiple-times independently
   - each partitioning is defined by one pivot space
2. given query q, multiple ranked candidate sets are generated
3. these multiple candidate rankings are effectively merged
   - the merged candidate set is smaller and more accurate

Our Approach in a Nutshell

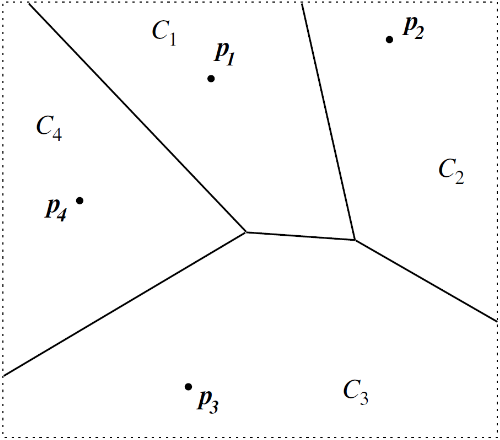
1. data space is partitioned multiple-times independently
   - each partitioning is defined by one pivot space
2. given query q, multiple ranked candidate sets are generated
3. these multiple candidate rankings are effectively merged
   - the merged candidate set is smaller and more accurate
4. the final candidate set is retrieved and refined

Voronoi Partitioning & Pivot Permutations

Pivot space is defined by a set of k pivots $\{p_1, \ldots, p_k\} \subseteq \mathcal{D}$

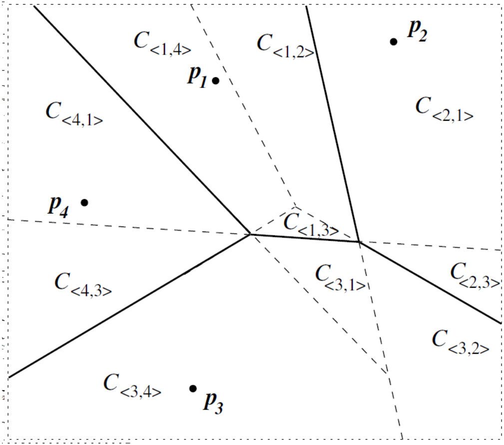
Voronoi Partitioning & Pivot Permutations

Pivot space is defined by a set of k pivots $\{p_1, \ldots, p_k\} \subseteq \mathcal{D}$



Voronoi Partitioning & Pivot Permutations

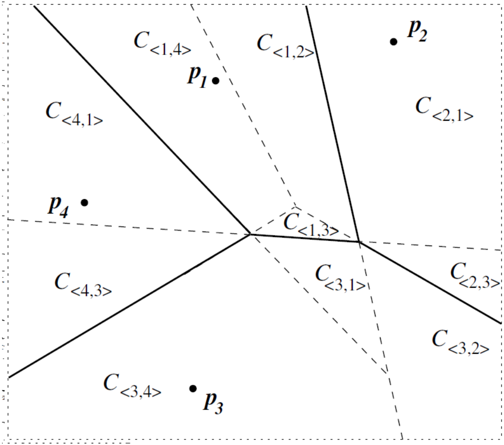
Pivot space is defined by a set of k pivots $\{p_1, \ldots, p_k\} \subseteq \mathcal{D}$



Voronoi Partitioning & Pivot Permutations

Pivot space is defined by a set of k pivots $\{p_1, \ldots, p_k\} \subseteq \mathcal{D}$

Formally:
object $x \in \mathcal{X}$ is mapped to its pivot permutation (PP):
Π_x on $\{1, \ldots, k\}$ such that $\Pi_x(i)$ is the i-th closest pivot from x

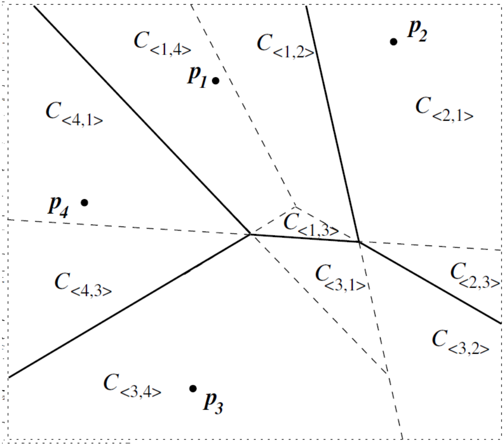


Voronoi Partitioning & Pivot Permutations

Pivot space is defined by a set of k pivots $\{p_1, \ldots, p_k\} \subseteq \mathcal{D}$

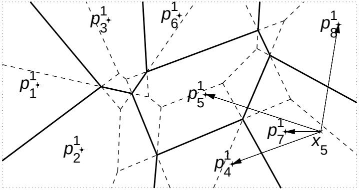
Formally:
object $x \in \mathcal{X}$ is mapped to its pivot permutation (PP): Π_x on $\{1, \ldots, k\}$ such that $\Pi_x(i)$ is the i-th closest pivot from x

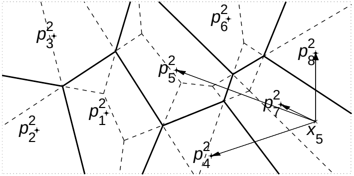
each Voronoi cell corresponds to a pivot permutation prefix (PPP) of length l: $\Pi_x(1..l)$



Multiple Pivot Space Partitioning

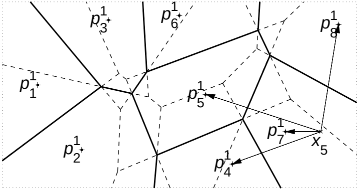
We propose to create λ independent pivot space partitionings

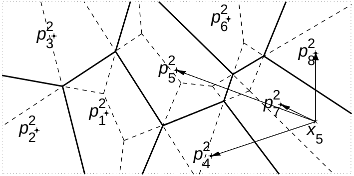




Multiple Pivot Space Partitioning

We propose to create λ independent pivot space partitionings



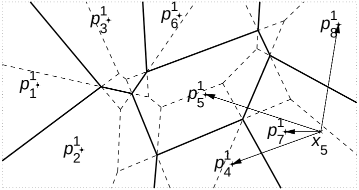


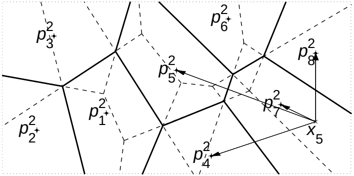
data objects $x \in \mathcal{X}$ are encoded as

$$PPP_l^{1..\lambda}(x) = \langle \Pi_x^1(1..l), \ldots, \Pi_x^\lambda(1..l) \rangle$$

Multiple Pivot Space Partitioning

We propose to create λ independent pivot space partitionings





data objects $x \in \mathcal{X}$ are encoded as

$$PPP_l^{1..\lambda}(x) = \langle \Pi_x^1(1..l), \ldots, \Pi_x^\lambda(1..l) \rangle$$

in the example above $\lambda = 2,\ k = 8,\ l = 4$:

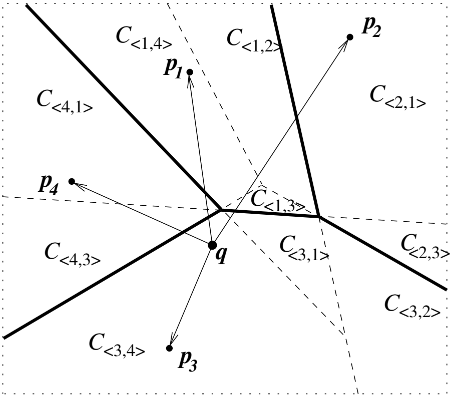
$$PPP_4^{1..2}(x_5) = \langle \langle 7,4,8,5 \rangle, \langle 7,8,4,6 \rangle \rangle$$

Ranking within a Single Pivot Space

Task: Having data $x \in \mathcal{X}$ encoded by PPP $\Pi_x(1..l)$ (single recursive Voronoi partitioning), define ranking of the PPPs with respect to $q \in \mathcal{D}$

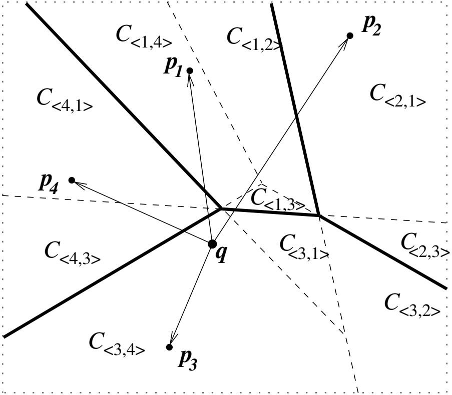
Ranking within a Single Pivot Space

Task: Having data $x \in \mathcal{X}$ encoded by PPP $\Pi_x(1..l)$ (single recursive Voronoi partitioning), define ranking of the PPPs with respect to $q \in \mathcal{D}$



Ranking within a Single Pivot Space

Solution: We define distance between Voronoi cell $C_{\langle i_1,\dots,i_l\rangle}$ and query q as a weighted arithmetic mean of distances $\delta(q, p_{i_1}), \dots, \delta(q, p_{i_l})$



Ranking using Multiple Pivot Spaces

Task: Having λ rankings of PPPs from λ pivot spaces, aggregate these rankings effectively into a final ranking

Ranking using Multiple Pivot Spaces

Task: Having λ rankings of PPPs from λ pivot spaces, aggregate these rankings effectively into a final ranking

$q \in \mathcal{D}$

$$
\begin{aligned}
\psi_q^{1}&: \overbrace{\{\boldsymbol{x}\; y_1\, y_2\}}^{\text{objects with the rank '1'}}\; \overbrace{\{y_3\, y_4\, y_5\}}^{\text{rank '2'}}\; \overbrace{\{y_6\}}^{\text{rank '3'}} \ldots\\
\psi_q^{2}&: \{y_3\, y_2\}\; \{y_1\, y_4\, y_6\, y_7\}\; \{\boldsymbol{x}\; y_8\} \ldots\\
\psi_q^{3}&: \{\boldsymbol{x}\}\; \{y_3\, y_4\, y_5\}\; \{y_2\, y_6\} \ldots\\
\psi_q^{4}&: \{y_1\, y_2\}\; \{y_3\, y_4\, y_5\}\; \{y_8\}\; \{y_6\} \ldots\\
\psi_q^{5}&: \{y_1\, y_2\}\; \{y_4\, y_5\}\; \{y_3\}\; \{\boldsymbol{x}\; y_7\} \ldots
\end{aligned}
$$

Ranking using Multiple Pivot Spaces

Solution: Ranking of object x is p-percentile (e.g. median) of its λ ranks

$$\Psi_{\mathbf{p}}(q,x) = percentile_{\mathbf{p}}(\psi_q^1(x), \psi_q^2(x), \ldots, \psi_q^\lambda(x))$$

$q \in \mathcal{D}$

$$\psi_q^1: \overbrace{\{\boldsymbol{x}\ y_1\ y_2\}}^{\text{objects with the rank '1'}}\ \overbrace{\{y_3\ y_4\ y_5\}}^{\text{rank '2'}}\ \overbrace{\{y_6\}}^{\text{rank '3'}}\ \ldots$$

$$\psi_q^2: \{y_3\ y_2\}\ \{y_1\ y_4\ y_6\ y_7\}\ \{\boldsymbol{x}\ y_8\}\ \ldots$$

$$\psi_q^3: \{\boldsymbol{x}\}\ \{y_3\ y_4\ y_5\}\ \{y_2\ y_6\}\ \ldots$$

$$\psi_q^4: \{y_1\ y_2\}\ \{y_3\ y_4\ y_5\}\ \{y_8\}\ \{y_6\}\ \ldots$$

$$\psi_q^5: \{y_1\ y_2\}\ \{y_4\ y_5\}\ \{y_3\}\ \{\boldsymbol{x}\ y_7\}\ \ldots$$

$$\Psi_{0.5}(q,x) = percentile_{0.5}\{1, 1, \mathbf{3}, 4, ?\} = 3$$

Idea Behind the Rank Aggregation

- the Voronoi cells span large areas of the space

Idea Behind the Rank Aggregation

- the Voronoi cells span large areas of the space
- given a query, the "close" cells contain also distant data objects
  - there is many more distant ones

Idea Behind the Rank Aggregation

- the Voronoi cells span large areas of the space
- given a query, the “close” cells contain also distant data objects
  - there is many more distant ones
- having several “orthogonal” partitionings
  - the query-relevant objects should be often at top positions
  - the distant objects vary

Idea Behind the Rank Aggregation

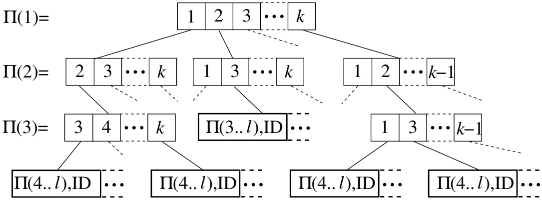
- the Voronoi cells span large areas of the space
- given a query, the “close” cells contain also distant data objects
  - there is many more distant ones
- having several “orthogonal” partitionings
  - the query-relevant objects should be often at top positions
  - the distant objects vary

- the percentile-based aggregation increases probability that query-relevant objects are ranked higher than the distant ones

Indexing the PPP-Codes

We build trie-like structure for each pivot space

- leafs: only suffixes of PPPs (spare memory)
- dynamic splits to optimize the memory usage
- possible grouping and delta-encoding of IDs in leaves



PPPRank: The Search Algorithm

$$
\begin{array}{ll}
\psi_q^{1}: & \overbrace{\{\boldsymbol{x}\ y_1\, y_2\}}^{\text{objects with the rank '1'}}\ \overbrace{\{y_3\, y_4\, y_5\}}^{\text{rank '2'}}\ \overbrace{\{y_6\}}^{\text{rank '3'}}\ \ldots \\
\psi_q^{2}: & \{y_3\, y_2\}\ \{y_1\, y_4\, y_6\, y_7\}\ \{\boldsymbol{x}\ y_8\}\ \ldots \\
\psi_q^{3}: & \{\boldsymbol{x}\}\ \{y_3\, y_4\, y_5\}\ \{y_2\, y_6\}\ \ldots \\
\psi_q^{4}: & \{y_1\, y_2\}\ \{y_3\, y_4\, y_5\}\ \{y_8\}\ \{y_6\}\ \ldots \\
\psi_q^{5}: & \{y_1\, y_2\}\ \{y_4\, y_5\}\ \{y_3\}\ \{\boldsymbol{x}\ y_7\}\ \ldots
\end{array}
$$

$$\Psi_{0.5}(q, x) = percentile_{0.5}\{1, 1, \mathbf{3}, 4, ?\} = 3$$

Given query $q \in \mathcal{D}$, our search algorithm:

PPPRank: The Search Algorithm

$$
\begin{array}{ll}
\psi_q^1: & \overbrace{\{\boldsymbol{x}\ y_1 y_2\}}^{\text{objects with the rank '1'}}\ \overbrace{\{y_3 y_4 y_5\}}^{\text{rank '2'}}\ \overbrace{\{y_6\}}^{\text{rank '3'}} \ \ldots \\
\psi_q^2: & \{y_3 y_2\}\ \{y_1 y_4 y_6 y_7\}\ \{\boldsymbol{x}\ y_8\} \ \ldots \\
\psi_q^3: & \{\boldsymbol{x}\}\ \{y_3 y_4 y_5\}\ \{y_2 y_6\} \ \ldots \\
\psi_q^4: & \{y_1 y_2\}\ \{y_3 y_4 y_5\}\ \{y_8\}\ \{y_6\} \ \ldots \\
\psi_q^5: & \{y_1 y_2\}\ \{y_4 y_5\}\ \{y_3\}\ \{\boldsymbol{x}\ y_7\} \ \ldots
\end{array}
$$

$$\Psi_{0.5}(q, x) = percentile_{0.5}\{1, 1, \mathbf{3}, 4, ?\} = 3$$

Given query $q \in \mathcal{D}$, our search algorithm:

1. generates one-by-one individual rankings ψ_q^j (GetNextIDs algorithm, it uses the trie structures)

PPPRank: The Search Algorithm

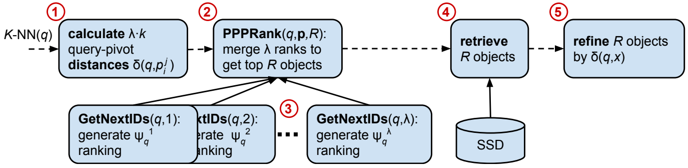
$$
\begin{array}{ll}
\psi_q^{1}: & \overbrace{\{\boldsymbol{x}\ y_1\, y_2\}}^{\text{objects with the rank '1'}}\ \overbrace{\{y_3\, y_4\, y_5\}}^{\text{rank '2'}}\ \overbrace{\{y_6\}}^{\text{rank '3'}}\ \ldots \\
\psi_q^{2}: & \{y_3\, y_2\}\ \{y_1\, y_4\, y_6\, y_7\}\ \{\boldsymbol{x}\ y_8\}\ \ldots \\
\psi_q^{3}: & \{\boldsymbol{x}\}\ \{y_3\, y_4\, y_5\}\ \{y_2\, y_6\}\ \ldots \\
\psi_q^{4}: & \{y_1\, y_2\}\ \{y_3\, y_4\, y_5\}\ \{y_8\}\ \{y_6\}\ \ldots \\
\psi_q^{5}: & \{y_1\, y_2\}\ \{y_4\, y_5\}\ \{y_3\}\ \{\boldsymbol{x}\ y_7\}\ \ldots
\end{array}
$$

$$\Psi_{0.5}(q, x) = percentile_{0.5}\{1, 1, \mathbf{3}, 4, ?\} = 3$$

Given query $q \in \mathcal{D}$, our search algorithm:

1. generates one-by-one individual rankings ψ_q^j (GetNextIDs algorithm, it uses the trie structures)
2. outputs objects with the best aggregated ranks (PPPRank algorithm based on MedRank by Fagin et al.)

PPPRank: The Search Algorithm



Given query $q \in \mathcal{D}$, our search algorithm:

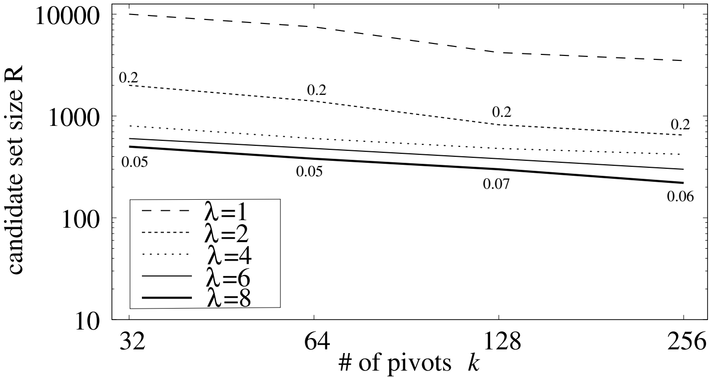
1. generates one-by-one individual rankings ψ_q^j (GetNextIDs algorithm, it uses the trie structures)
2. outputs objects with the best aggregated ranks (PPPRank algorithm based on MedRank by Fagin et al.)

Evaluation: Accuracy of the Candidate Set

Given K-NN, we consider $\mathit{recall}(A) = \frac{|A \cap A^P|}{K} \cdot 100\%$ vs. candidate set size

Evaluation: Accuracy of the Candidate Set

Given K-NN, we consider *recall*$(A) = \frac{|A \cap A^P|}{K} \cdot 100\%$ vs. candidate set size



Candidate set size R necessary to achieve 80% of 1-NN recall

Settings: 1M CoPhIR dataset, $l = 8$ and $\mathbf{p} = 0.75$

Experimental Evaluation Criteria

three datasets:

- 100M CoPhIR (280-dim, complex metric, obj.: 600 B, δ time 0.01 ms)
- 1M SQFD (quadratic form distance, obj.: 2 kB, δ time 0.5 ms)
- 10M ADJ ($[0,1]^{32}$ uniform, L_2, obj.: 0.5–4.0 kB, δ time 0.001–1.0 ms)

Experimental Evaluation Criteria

three datasets:

- 100M CoPhIR (280-dim, complex metric, obj.: 600 B, δ time 0.01 ms)
- 1M SQFD (quadratic form distance, obj.: 2 kB, δ time 0.5 ms)
- 10M ADJ ($[0,1]^{32}$ uniform, L_2, obj.: 0.5–4.0 kB, δ time 0.001–1.0 ms)

technical evaluation of our approach:

- mutual influence of various parameters to recall
  - k, l, λ, $\mathbf{p}$, size of the PPP-Code representation

Experimental Evaluation Criteria

three datasets:

- 100M CoPhIR (280-dim, complex metric, obj.: 600 B, δ time 0.01 ms)
- 1M SQFD (quadratic form distance, obj.: 2 kB, δ time 0.5 ms)
- 10M ADJ ($[0, 1]^{32}$ uniform, L_2, obj.: 0.5–4.0 kB, δ time 0.001–1.0 ms)

technical evaluation of our approach:

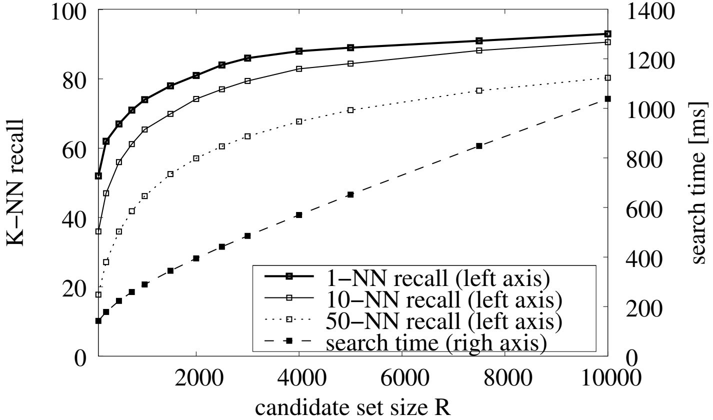
- mutual influence of various parameters to recall
  - k, l, λ, $\mathbf{p}$, size of the PPP-Code representation
- $k \in \{64, 128, 256, 512\}$, $l = 8$, $\lambda = 5$, $\mathbf{p} = 0.5$ (3rd rank out of 5)

Evaluation: Candidate Set vs. Recall

candidate set size R vs. recall

Evaluation: Candidate Set vs. Recall

candidate set size R vs. recall



Recall and search time on while increasing candidate set size R.

Settings: 100M CoPhIR dataset, $k = 512$

Evaluation: Tradeoff

complexity of the PPPRank algorithm vs. candidate set reduction

Evaluation: Tradeoff

complexity of the PPPRank algorithm vs. candidate set reduction

	PPP-Code / M-Index [ms]	size of an object [bytes] 512	1024	2048	4096
δ time	0.001 ms	370 / **240**	**370** / 410	**370** / 1270	**370** / 1700
	0.01 ms	**380** / 660	**380** / 750	**380** / 1350	**380** / 1850
	0.1 ms	**400** / 5400	**400** / 5400	**420** / 5500	**420** / 5700
	1 ms	**1100** / 52500	**1100** / 52500	**1100** / 52500	**1100** / 52500

Search times [ms] of PPP-Codes / M-Index smaller search times are in boldface.

Settings: 10M ADJUSTABLE dataset, 10-NN recall = 85 %, $k = 128$;
PPP-Codes: $R = 1000$; M-Index: $R = 400000$

Conclusions

The PPP-Codes technique

- use multiple pivot spaces to encode data objects
- rank data with respect to query within individual pivot spaces
- final candidate set is aggregation of these rankings
- efficient indexing and searching mechanisms are defined

Conclusions

The PPP-Codes technique

- use multiple pivot spaces to encode data objects
- rank data with respect to query within individual pivot spaces
- final candidate set is aggregation of these rankings
- efficient indexing and searching mechanisms are defined

The results show that

- even two pivot spaces help, more than five do not help much
- the candidate set is reduced by one–two orders of magnitude
- the rank & merge algorithm is complex but usually worth