Distance-based Similarity Searching

and its Applications in Multimedia Retrieval with Deep Learning

David Novak

Laboratory of Data Intensive Systems and Applications (DISA)
Masaryk University, Brno, Czech Republic

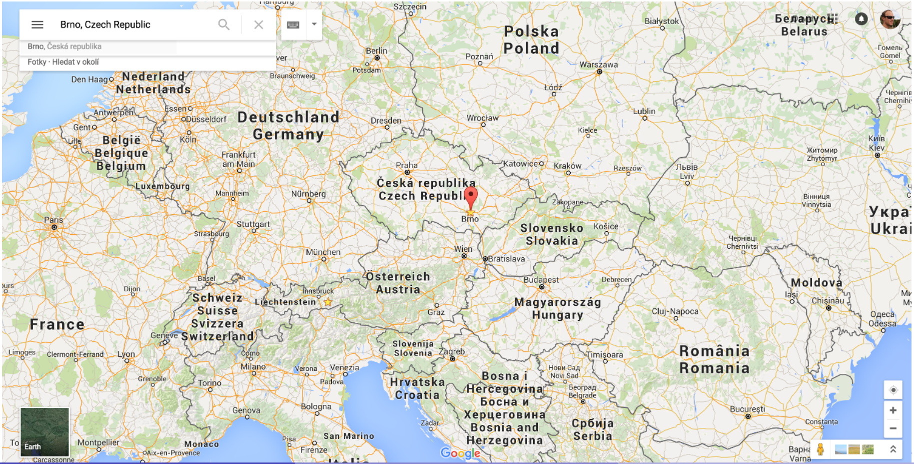
Fulbright scholar at CIIR, UMass, Amherst, MA, USA

CIIR Meeting, 21st September 2015

http://disa.fi.muni.cz/david-novak/

Where Do I Come From

Lab of Data Intensive Systems and Applications (DISA) http://disa.fi.muni.cz/
Faculty of Informatics, Masaryk University http://www.fi.muni.cz/
Brno, Czech Republic



Where Do I Come From

Lab of Data Intensive Systems and Applications (DISA) http://disa.fi.muni.cz/
Faculty of Informatics, Masaryk University http://www.fi.muni.cz/
Brno, Czech Republic

- Head of DISA Lab: prof. Pavel Zezula
- Post-doc researches:
  - Michal Batko
  - Petra Budikova
  - Vlastislav Dohnal
  - David Novak
  - Jan Sedmidubsky
- PhD students:
  - current: 6
  - successful (over 12 years): 10
- undergraduates
- fields of interest: databases, Big Data processing, similarity searching, multimedia retrieval, biometrics

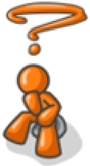
Similarity Searching

- The similarity is key to human cognition, learning, memory…

[cognitive psychology]

Similarity Searching

- The similarity is key to human cognition, learning, memory. . .
  [cognitive psychology]
- Everything we can see, hear, measure, observe is in digital form



Similarity Searching

- The similarity is key to human cognition, learning, memory. . .

  [cognitive psychology]
- Everything we can see, hear, measure, observe is in digital form
- Computers should be able to search data based on similarity

Similarity Searching

- The similarity is key to human cognition, learning, memory. . .

  [cognitive psychology]
- Everything we can see, hear, measure, observe is in digital form
- Computers should be able to search data based on similarity

The similarity search problem has two aspects

- **effectiveness**: how to measure similarity of two "objects"
  - domain specific (data- and application-specific, context dependent, . . .)
  - photos, video, X-rays, voice, music, EEG, MTR, texts, . . .

Similarity Searching

- The similarity is key to human cognition, learning, memory...

  [cognitive psychology]
- Everything we can see, hear, measure, observe is in digital form
- Computers should be able to search data based on similarity

The similarity search problem has two aspects

- **effectiveness**: how to measure similarity of two "objects"
  - domain specific (data- and application-specific, context dependent, ...)
  - photos, video, X-rays, voice, music, EEG, MTR, texts, ...
- **efficiency**: how to realize similarity search fast
  - using a given data + similarity measure
  - on very large data collections

Efficiency: Motivation Example

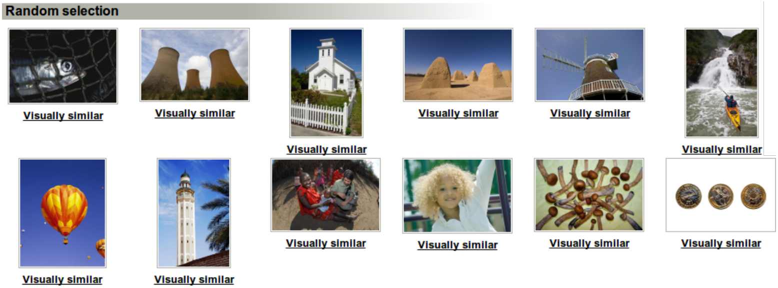
Example of data:

- general images (photos)
- every image processed by a deep convolutional neural network
  - to obtain a visual characterization of the image (feature)
  - compared by Euclidean distance to measure generic visual similarity

Efficiency: Motivation Example

Example of data:

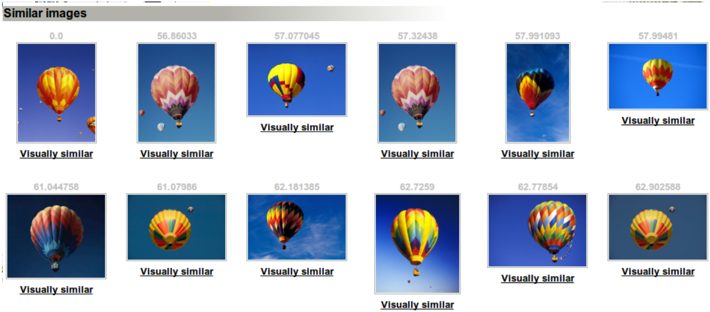
- general images (photos)
- every image processed by a deep convolutional neural network
  - to obtain a visual characterization of the image (feature)
  - compared by Euclidean distance to measure generic visual similarity



Efficiency: Motivation Example

Example of data:

- general images (photos)
- every image processed by a deep convolutional neural network
  - to obtain a visual characterization of the image (feature)
  - compared by Euclidean distance to measure generic visual similarity



Efficiency: Motivation Example

Example of data:

- general images (photos)
- every image processed by a deep convolutional neural network
  - to obtain a visual characterization of the image (feature)
  - compared by Euclidean distance to measure generic visual similarity

Efficiency problem:

- 20 million of images with such descriptors
- each descriptor is a 4096-dimensional float vector (16 kB)
- ⇒ over 320 GB of data to be organized for similarity search
  - answer similarity queries online

Outline of the Talk

1. Motivation & Fundamentals
   - Similarity Search: Effectiveness and Efficiency
2. Indexing & Searching in Metric Spaces
   - Metric-based Model of Similarity
   - Overview and Principles
   - Voronoi Partitioning
3. Specific Similarity Indexes
   - M-Index
   - PPP-Codes
4. Deep Convolutional Neural Networks
   - and their applications in image recognition
5. Visual Search Demo

Metric-based Model of Similarity

- We model the data 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 the following postulates $\forall x, y, x \in \mathcal{D}$:
  - identity: $\delta(x, x) = 0$
  - symmetry: $\delta(x, y) = \delta(y, x)$
  - triangle inequality: $\delta(x, y) \leq \delta(x, z) + \delta(z, y)$

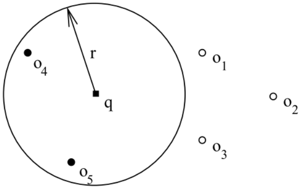
Metric-based Model of Similarity

- We model the data 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 the following postulates $\forall x, y, x \in \mathcal{D}$:
  - identity: $\delta(x, x) = 0$
  - symmetry: $\delta(x, y) = \delta(y, x)$
  - triangle inequality: $\delta(x, y) \leq \delta(x, z) + \delta(z, y)$
- Objective: organize data collection $\mathcal{X} \subseteq \mathcal{D}$, resolve query by example

Metric-based Model of Similarity

- We model the data 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 the following postulates $\forall x, y, x \in \mathcal{D}$:
  - identity: $\delta(x, x) = 0$
  - symmetry: $\delta(x, y) = \delta(y, x)$
  - triangle inequality: $\delta(x, y) \leq \delta(x, z) + \delta(z, y)$
- Objective: organize data collection $\mathcal{X} \subseteq \mathcal{D}$, resolve query by example

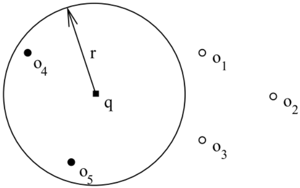
range query $R(q, r)$ returns all objects $x \in \mathcal{X}$ with $\delta(q, x) \leq r$



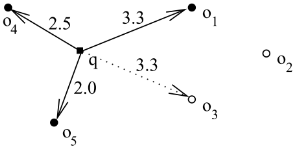
Metric-based Model of Similarity

- We model the data 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 the following postulates $\forall x, y, x \in \mathcal{D}$:
  - identity: $\delta(x, x) = 0$
  - symmetry: $\delta(x, y) = \delta(y, x)$
  - triangle inequality: $\delta(x, y) \leq \delta(x, z) + \delta(z, y)$
- Objective: organize data collection $\mathcal{X} \subseteq \mathcal{D}$, resolve query by example

range query $R(q, r)$ returns all objects $x \in \mathcal{X}$ with $\delta(q, x) \leq r$



k-NN(q) query returns k objects $x \in \mathcal{X}$ with the smallest $\delta(q, x)$



Metric vs. Other Models

Metric model of similarity

- is very generic
  - applicable to many data types + similarity functions
- we can build one index structure and apply many times
- in some cases, we can even omit the triangle inequality (see below)

Metric vs. Other Models

Metric model of similarity

- is very generic
  - applicable to many data types + similarity functions
- we can build one index structure and apply many times
- in some cases, we can even omit the triangle inequality (see below)

On the other hand

- techniques for specific similarity are often more efficient
  - e.g. cosine similarity used with vector space model in IR
  - the inverted file indexes are very convenient for sparse vectors

Metric vs. Other Models

Metric model of similarity

- is very generic
  - applicable to many data types + similarity functions
- we can build one index structure and apply many times
- in some cases, we can even omit the triangle inequality (see below)

On the other hand

- techniques for specific similarity are often more efficient
  - e.g. cosine similarity used with vector space model in IR
  - the inverted file indexes are very convenient for sparse vectors

But the distance-based indexing is often able to capture intrinsic complexity of the similarity space

- ignoring unnecessary external "dimensions" of the data

Examples of Metric Spaces

Vector data:

- L_p metrics (Minkowski distances)
  - L_1 – Manhattan (city block) distance $$L_1(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$
  - L_2 – Euclidean distance $$L_2(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$
  - L_∞ – Chebyshev distance $$L_\infty(x,y) = \max_{i=1}^{n} |x_i - y_i|$$

Examples of Metric Spaces

Vector data:

- L_p metrics (Minkowski distances)
  - L_1 – Manhattan (city block) distance $$L_1(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$
  - L_2 – Euclidean distance $$L_2(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$
  - L_∞ – Chebyshev distance $$L_\infty(x,y) = \max_{i=1}^{n} |x_i - y_i|$$
- quadratic form (Mahalanobis) $$\delta(x,y) = \left((x-y)^T \cdot \mathrm{M} \cdot (x-y)\right)^{\frac{1}{2}}$$
  - where M is a matrix $n \times n$

Examples of Metric Spaces

Vector data:

- L_p metrics (Minkowski distances)
  - L_1 – Manhattan (city block) distance $$L_1(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$
  - L_2 – Euclidean distance $$L_2(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$
  - L_∞ – Chebyshev distance $$L_\infty(x,y) = \max_{i=1}^{n} |x_i - y_i|$$
- quadratic form (Mahalanobis) $$\delta(x,y) = \left((x-y)^T \cdot \mathrm{M} \cdot (x-y)\right)^{\frac{1}{2}}$$
  - where M is a matrix $n \times n$
- cosine distance $$1 - \cos(x,y)$$
  - does NOT fulfill triangle inequality
  - if required, use angular similarity $$1 - \frac{\cos^{-1}\cos(x,y)}{\pi}$$

Examples of Metric Spaces

Vector data:

- L_p metrics (Minkowski distances)
  - L_1 – Manhattan (city block) distance $$L_1(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$
  - L_2 – Euclidean distance $$L_2(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$
  - L_∞ – Chebyshev distance $$L_\infty(x,y) = \max_{i=1}^{n} |x_i - y_i|$$
- quadratic form (Mahalanobis) $$\delta(x,y) = \left((x-y)^T \cdot \mathrm{M} \cdot (x-y)\right)^{\frac{1}{2}}$$
  - where M is a matrix $n \times n$
- cosine distance $$1 - \cos(x,y)$$
  - does NOT fulfill triangle inequality
  - if required, use angular similarity $$1 - \frac{\cos^{-1}\cos(x,y)}{\pi}$$
- Hamming distance (on vectors over a finite field, e.g. binary vectors)

Examples of Metric Spaces (2)

Strings:

- edit distance (Levenshtein distance)
  - minimum number of *insertions*, *deletions* or *substitutions*

Examples of Metric Spaces (2)

Strings:

- edit distance (Levenshtein distance)
  - minimum number of *insertions*, *deletions* or *substitutions*

Sets:

- Jaccard's coefficient $\delta(X, Y) = 1 - \frac{X \cap Y}{X \cup Y}$
- Hausdorff distance
  - for sets with elements related by another distance

Examples of Metric Spaces (2)

Strings:

- edit distance (Levenshtein distance)
  - minimum number of *insertions*, *deletions* or *substitutions*

Sets:

- Jaccard's coefficient $\delta(X,Y) = 1 - \frac{X \cap Y}{X \cup Y}$
- Hausdorff distance
  - for sets with elements related by another distance

Other types of data:

- Earth mover's distance (for histograms)
- Tree Edit distance
- Signature Quadratic Form distance, etc.

Problem Formulation and Overview

Objective: Preprocess and organize collection of objects $\mathcal{X} \subseteq \mathcal{D}$ in such a way that similarity queries are processed efficiently

- collections can be very large
- computation of distance function δ can be expensive

Problem Formulation and Overview

Objective: Preprocess and organize collection of objects $\mathcal{X} \subseteq \mathcal{D}$ in such a way that similarity queries are processed efficiently

- collections can be very large
- computation of distance function δ can be expensive

Two decades of research in this area

- theoretical principles identified
- static and dynamic memory structures for precise similarity search

Problem Formulation and Overview

Objective: Preprocess and organize collection of objects $\mathcal{X} \subseteq \mathcal{D}$ in such a way that similarity queries are processed efficiently

- collections can be very large
- computation of distance function δ can be expensive

Two decades of research in this area

- theoretical principles identified
- static and dynamic memory structures for precise similarity search
- efficient disk-oriented techniques
  - precise and approximate (not all objects from k-NN answer returned)
- distributed processing

Space Partitioning

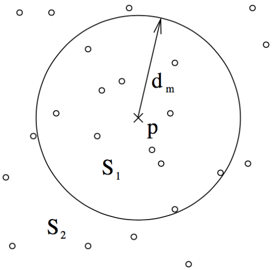
- Key and fundamental task of indexing is to partition the collection

Space Partitioning

- Key and fundamental task of indexing is to partition the collection
- But in metric space
  - the objects do not have any dimensions to partition the space
  - there is no “absolute” ordering of the objects
  - just with respect to some object

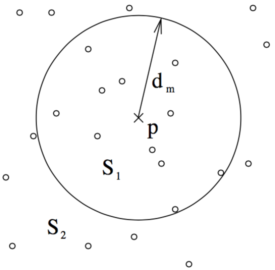
Space Partitioning

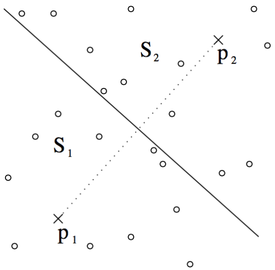
- Key and fundamental task of indexing is to partition the collection
- But in metric space
  - the objects do not have any dimensions to partition the space
  - there is no "absolute" ordering of the objects
  - just with respect to some object



Space Partitioning

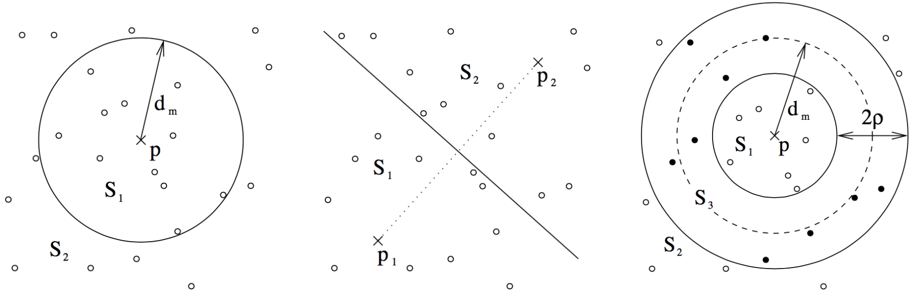
- Key and fundamental task of indexing is to partition the collection
- But in metric space
  - the objects do not have any dimensions to partition the space
  - there is no "absolute" ordering of the objects
  - just with respect to some object





Space Partitioning

- Key and fundamental task of indexing is to partition the collection
- But in metric space
  - the objects do not have any dimensions to partition the space
  - there is no “absolute” ordering of the objects
  - just with respect to some object

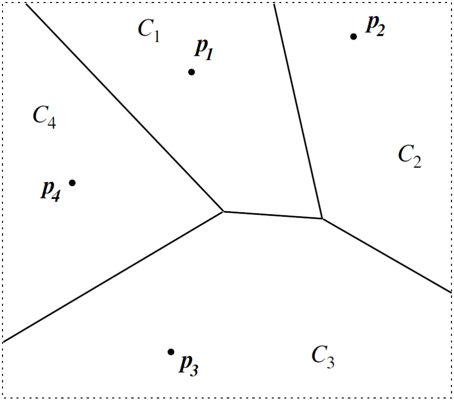


Voronoi Partitioning

- Partitioning using a fixed set of reference objects (pivots)

Voronoi Partitioning

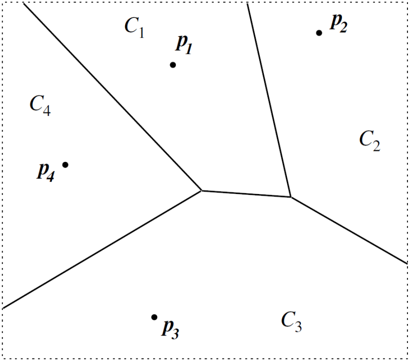
- Partitioning using a fixed set of reference objects (pivots)
- Let us have a set of n pivots $\{p_1, \ldots, p_n\}$



- Voronoi cell C_i = all objects for which pivot p_i is the closest

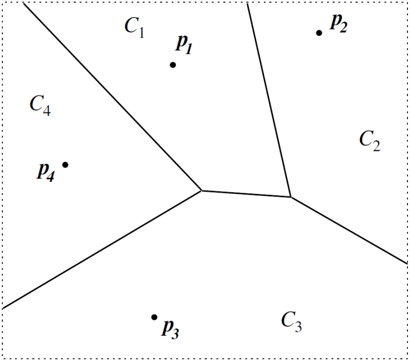
Recursive Voronoi Partitioning

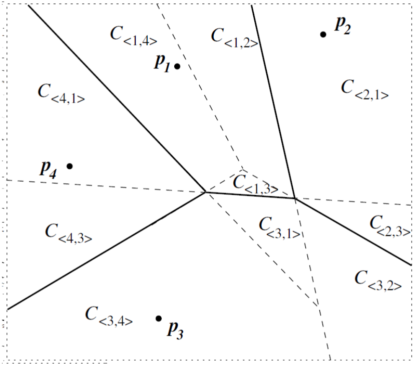
- Let us use the same set of n pivots $p_1, \ldots, p_n$ recursively



Recursive Voronoi Partitioning

- Let us use the same set of n pivots $p_1, \ldots, p_n$ recursively

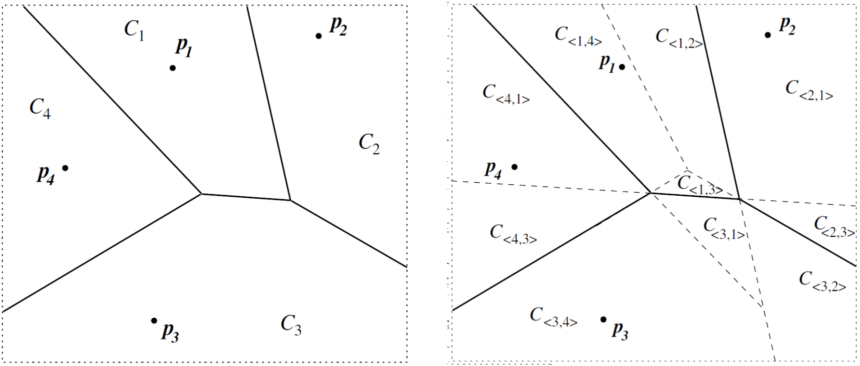




- Partition each C_i using the other pivots $p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n$

Recursive Voronoi Partitioning

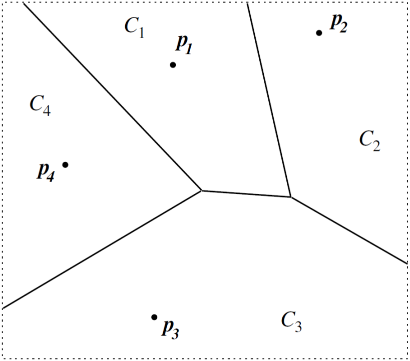
- Let us use the same set of n pivots $p_1, \ldots, p_n$ recursively

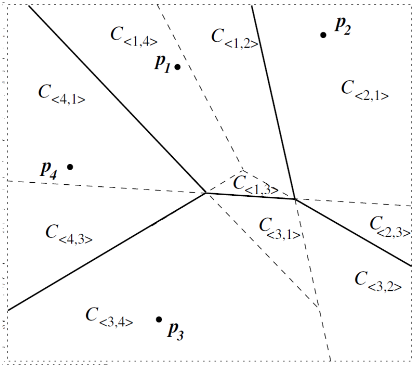


- Partition each C_i using the other pivots $p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n$
- $C_{i,j}$ = objects for which p_i is the closest and p_j the second closest

Recursive Voronoi Partitioning

- Let us use the same set of n pivots $p_1, \ldots, p_n$ recursively



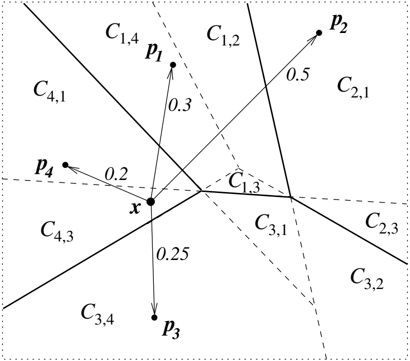


- Partition each C_i using the other pivots $p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n$
- $C_{i,j}$ = objects for which p_i is the closest and p_j the second closest
  - this principle can be used l-times recursively up to level $l = n$

Pivot Permutations

A different point of view: **(prefixes of) pivot permutations**

- Given object $x \in \mathcal{X}$, order the pivots according to distances $\delta(x, p_i)$



Pivot Permutations

A different point of view: **(prefixes of) pivot permutations**

- Given object $x \in \mathcal{X}$, order the pivots according to distances $\delta(x, p_i)$

- Let Π_x be a permutation on the set of pivot indexes $\{1, \ldots, n\}$ such that $\Pi_x(j)$ is index of the j-th closest pivot from x
  - for example, $\Pi_x(1)$ is index of the closest pivot from x
  - $p_{\Pi_x(j)}$ is the j-the closest pivot from x

Pivot Permutations

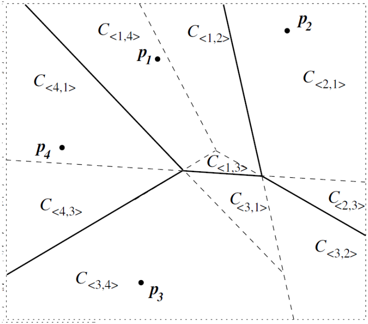
A different point of view: **(prefixes of) pivot permutations**

- Given object $x \in \mathcal{X}$, order the pivots according to distances $\delta(x, p_i)$

- Let Π_x be a permutation on the set of pivot indexes $\{1, \ldots, n\}$ such that $\Pi_x(j)$ is index of the j-th closest pivot from x
  - for example, $\Pi_x(1)$ is index of the closest pivot from x
  - $p_{\Pi_x(j)}$ is the j-the closest pivot from x

- Π_x is denoted as pivot permutation (PP) with respect to x.

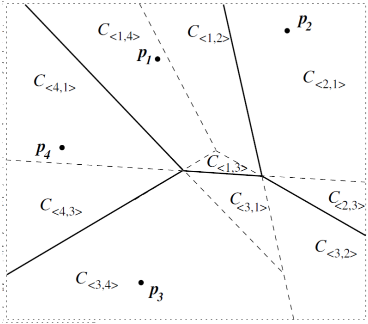
Correspondence between Voronoi partitioning and PPs



- Recursive Voronoi partitioning to level l
- Cell $C_{\langle i_1,\dots,i_l \rangle}$ contains objects x for which

$$\Pi_x(1) = i_1,\ \Pi_x(2) = i_2,\ \dots, \Pi_x(l) = i_l$$

Correspondence between Voronoi partitioning and PPs



- Recursive Voronoi partitioning to level l
- Cell $C_{\langle i_1,\ldots,i_l\rangle}$ contains objects x for which

$$\Pi_x(1) = i_1,\ \Pi_x(2) = i_2,\ \ldots, \Pi_x(l) = i_l$$

- l-tuple $\langle i_1, \ldots, i_l\rangle$ is an l-prefix of pivot permutation Π_x
  - pivot permutation prefix (PPP)
  - there is one-to-one correspondence between "Voronoi cell" and "PPP"

M-Index Indexing Structure

Specific Voronoi-based indexes: M-Index, Distributed M-Index, PPP-Codes

M-Index Indexing Structure

Specific Voronoi-based indexes: M-Index, Distributed M-Index, PPP-Codes

M-Index: basic properties

- uses dynamic recursive Voronoi partitioning (see below)
- it defines a (hash) mapping from the metric space to (float) numbers
- data either in memory or on disk (continuous chunks)
- both precise and approximate similarity search

M-Index Indexing Structure

Specific Voronoi-based indexes: M-Index, Distributed M-Index, PPP-Codes

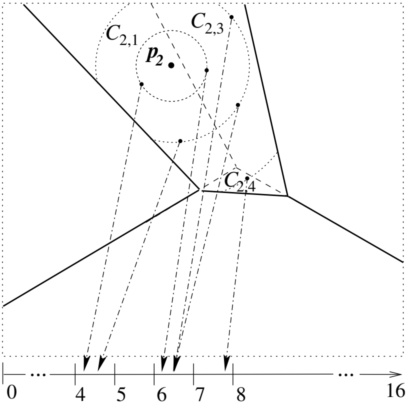
M-Index: basic properties

- uses dynamic recursive Voronoi partitioning (see below)
- it defines a (hash) mapping from the metric space to (float) numbers
- data either in memory or on disk (continuous chunks)
- both precise and approximate similarity search

Novak, D. and Batko, M. (2009). Metric Index: An Efficient and Scalable Solution for Similarity Search. In *Proceedings of SISAP '09*, (pp. 65–73). IEEE Comput. Soc. Press.

Novak, D., Batko, M. and Zezula, P. (2011). Metric Index: An Efficient and Scalable Solution for Precise and Approximate Similarity Search. *Inform. Syst.*, 36(4), 721–733.

M-Index Mapping Function



example with $n = 4$ and $l = 2$

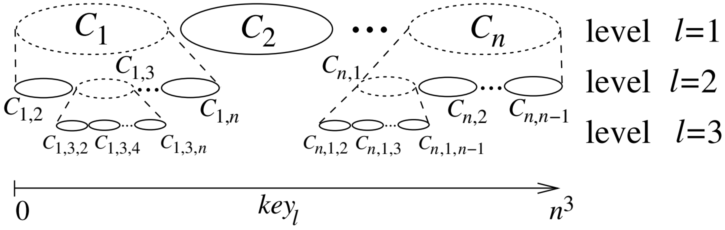
integral part of the key

- identification of the cell

fractional part of the key

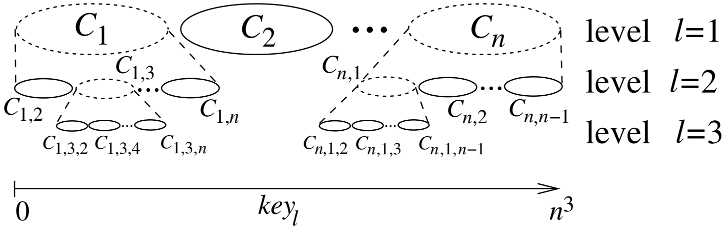
- position within the cell
- distance from the closest pivot

M-Index with Dynamic Level



- partition only those cells that exceed certain capacity
  - pick a maximum level $1 \leq l_{\max} \leq n$

M-Index with Dynamic Level



- partition only those cells that exceed certain capacity
  - pick a maximum level $1 \leq l_{max} \leq n$
- a kind of dynamic hashing, similar to extensible hashing
- it is a locality sensitive hashing for generic distance spaces

M-Index: Precise Query Evaluation Strategy

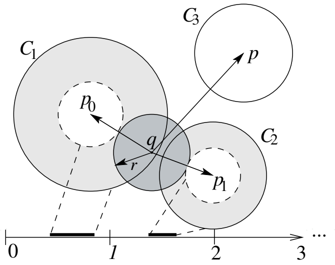
- The precise query evaluation gives guarantees to return all results
  - for both range query $R(q, r)$ and k-NN queries

M-Index: Precise Query Evaluation Strategy

- The precise query evaluation gives guarantees to return all results
  - for both range query $R(q, r)$ and k-NN queries
- search principle: filter & refine
  - filter out parts of index that cannot contain query relevant objects

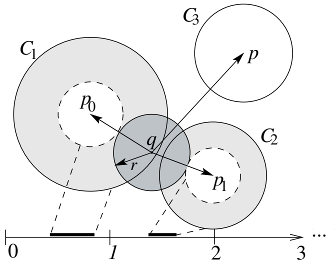
M-Index: Precise Query Evaluation Strategy

- The precise query evaluation gives guarantees to return all results
  - for both range query $R(q, r)$ and k-NN queries
- search principle: filter & refine
  - filter out parts of index that cannot contain query relevant objects



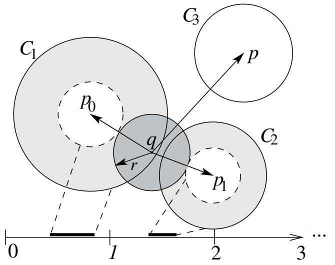
M-Index: Precise Query Evaluation Strategy

- The precise query evaluation gives guarantees to return all results
  - for both range query $R(q, r)$ and k-NN queries
- search principle: filter & refine
  - filter out parts of index that cannot contain query relevant objects
  - for objects $x \in \mathcal{X}$ that cannot be filtered out, calculate $\delta(q, x)$



M-Index: Precise Query Evaluation Strategy

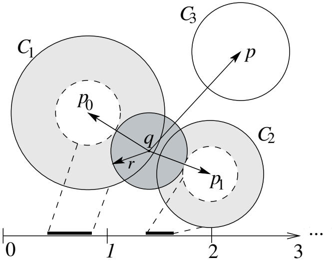
- The precise query evaluation gives guarantees to return all results
  - for both range query $R(q, r)$ and k-NN queries
- search principle: filter & refine
  - filter out parts of index that cannot contain query relevant objects
  - for objects $x \in \mathcal{X}$ that cannot be filtered out, calculate $\delta(q, x)$



- M-Index employs practically all known metric principles of space pruning and filtering
  - triangle inequality required

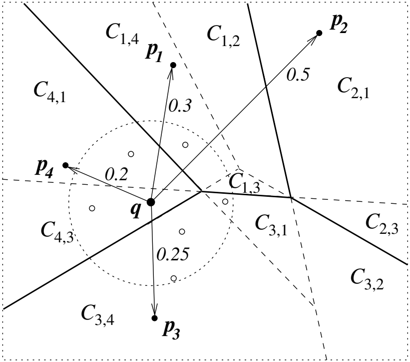
M-Index: Precise Query Evaluation Strategy

- The precise query evaluation gives guarantees to return all results
  - for both range query $R(q, r)$ and k-NN queries
- search principle: filter & refine
  - filter out parts of index that cannot contain query relevant objects
  - for objects $x \in \mathcal{X}$ that cannot be filtered out, calculate $\delta(q, x)$
- M-Index employs practically all known metric principles of space pruning and filtering
  - triangle inequality required



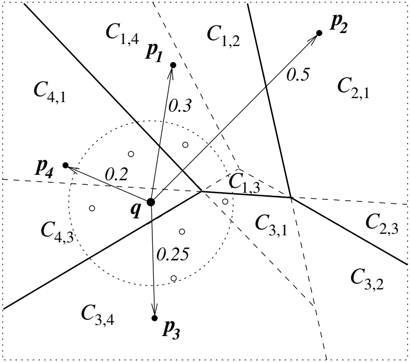
- the search must still access and refine about 30–50% of the data
  - for collections with high intrinsic dimensionality (dimensionality curse)
  - for $R(q, r)$ and k-NN queries with reasonable r and k

Approximate Strategy for M-Index



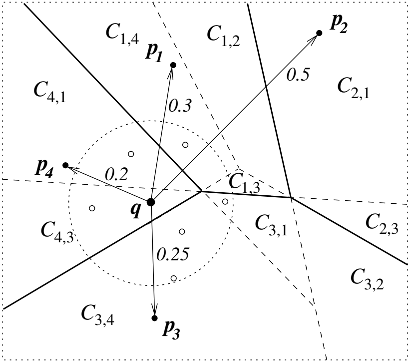
- determine cells with a high chance to contain relevant objects
- use query-pivot distances $\delta(q, p_1), \delta(q, p_2), \ldots, \delta(q, p_n)$

Approximate Strategy for M-Index



- determine cells with a high chance to contain relevant objects
- use query-pivot distances $\delta(q, p_1), \delta(q, p_2), \ldots, \delta(q, p_n)$
- estimate "distances" between the query and Voronoi cells (PPPs)
- $\Longrightarrow$ rank the Voronoi cells (data) with respect to the query

Approximate Strategy for M-Index

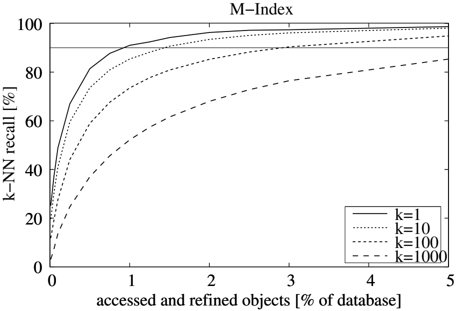


- determine cells with a high chance to contain relevant objects
- use query-pivot distances $\delta(q, p_1), \delta(q, p_2), \ldots, \delta(q, p_n)$
- estimate "distances" between the query and Voronoi cells (PPPs)
- $\Longrightarrow$ rank the Voronoi cells (data) with respect to the query

- measure quality of k-NN approximate search by recall = precision

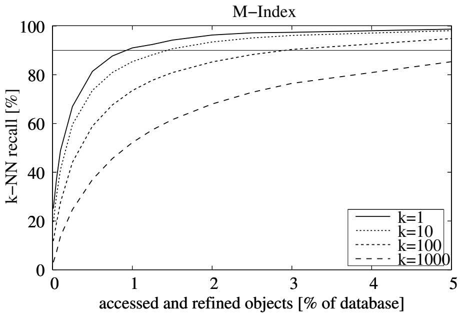
$$recall(A) = \frac{|A \cap A^P|}{k} \cdot 100\%$$

M-Index Approximate Strategy: Brief Evaluation



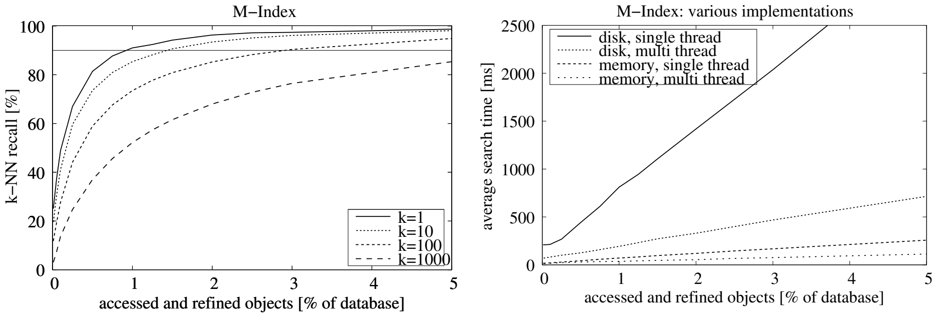
- collection of 1M 4096-dimensional float vectors with L_2 distance

M-Index Approximate Strategy: Brief Evaluation



- collection of 1M 4096-dimensional float vectors with L_2 distance
- access and refine 3–5 % of the data
  - to achieve recall over 90 %
  - the larger the data collection, the lower the percentage

M-Index Approximate Strategy: Brief Evaluation

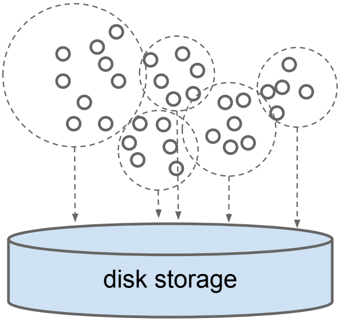


- collection of 1M 4096-dimensional float vectors with L_2 distance
- access and refine 3–5 % of the data
  - to achieve recall over 90 %
  - the larger the data collection, the lower the percentage

Standard Similarity Search Approach

standard approach to large-scale approximate search (e.g. M-Index):

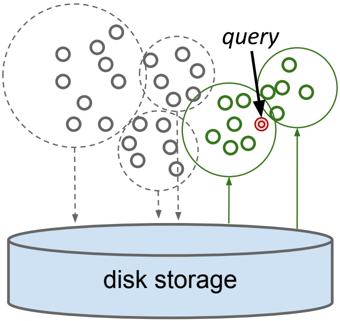
- dataset $\mathcal{X}$ is partitioned and stored



Standard Similarity Search Approach

standard approach to large-scale approximate search (e.g. M-Index):

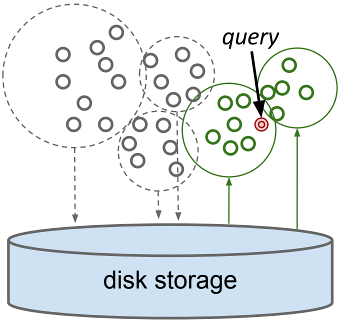
- dataset $\mathcal{X}$ is partitioned and stored
- given query q, the "most-promising" partitions form the candidate set



Standard Similarity Search Approach

standard approach to large-scale approximate search (e.g. M-Index):

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



Standard Similarity Search Approach

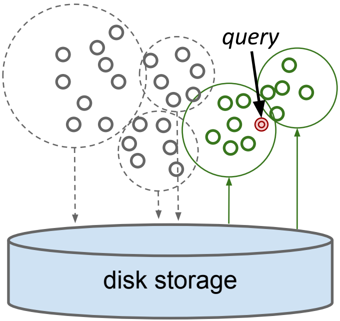
standard approach to large-scale approximate search (e.g. M-Index):

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

majority of the search costs:

- reading and refinement of S_C

$\Longrightarrow$ accuracy of the candidate set is key



PPP-Code: The Core Idea

- the Voronoi cells span large areas of the space

PPP-Code: The Core Idea

- the Voronoi cells span large areas of the space
- given a query, the "close" cells contain also distant data objects
    - actually, far more distant objects than the close ones

PPP-Code: The Core Idea

- the Voronoi cells span large areas of the space
- given a query, the "close" cells contain also distant data objects
  - actually, far more distant objects than the close ones
- having several independent "orthogonal" partitionings
  - the relevant objects should be in close cells of "all" partitionings
  - the distant objects in close cells vary

PPP-Code: The Core Idea

- the Voronoi cells span large areas of the space
- given a query, the “close” cells contain also distant data objects
  - actually, far more distant objects than the close ones
- having several independent “orthogonal” partitionings
  - the relevant objects should be in close cells of “all” partitionings
  - the distant objects in close cells vary
- aggregate the candidate sets from individual partitionings so that objects that appear “often” on top positions are ranked high

PPP-Code: The Core Idea

- the Voronoi cells span large areas of the space
- given a query, the “close” cells contain also distant data objects
  - actually, far more distant objects than the close ones
- having several independent “orthogonal” partitionings
  - the relevant objects should be in close cells of “all” partitionings
  - the distant objects in close cells vary
- aggregate the candidate sets from individual partitionings so that objects that appear “often” on top positions are ranked high

Novak, D. and Zezula, P. Rank Aggregation of Candidate Sets for Efficient Similarity Search. In Proceedings of 25th Inter. Conf. on Database and Expert Systems Applications (DEXA 2014)

Best paper of DEXA 2014 Award.

PPP-Codes in a Nutshell

1. data space is partitioned multiple-times by recursive Voronoi

PPP-Codes in a Nutshell

1. data space is partitioned multiple-times by recursive Voronoi
2. based on these partitionings, objects are mapped onto memory codes
    - a dynamic memory index is created using these codes

PPP-Codes in a Nutshell

1. data space is partitioned multiple-times by recursive Voronoi
2. based on these partitionings, objects are mapped onto memory codes
   - a dynamic memory index is created using these codes
3. given query q, multiple ranked candidate sets are generated

PPP-Codes in a Nutshell

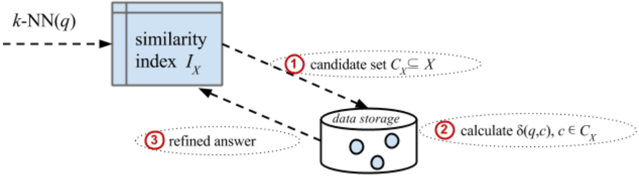
1. data space is partitioned multiple-times by recursive Voronoi
2. based on these partitionings, objects are mapped onto memory codes
    - a dynamic memory index is created using these codes
3. given query q, multiple ranked candidate sets are generated
4. these candidate rankings are merged using median of individual ranks
    - the merged candidate set is smaller and more accurate

PPP-Codes in a Nutshell

1. data space is partitioned multiple-times by recursive Voronoi
2. based on these partitionings, objects are mapped onto memory codes
   - a dynamic memory index is created using these codes
3. given query q, multiple ranked candidate sets are generated
4. these candidate rankings are merged using median of individual ranks
   - the merged candidate set is smaller and more accurate
5. the final candidate set is retrieved and refined
   - object-by-object, objects stored in a ID-object store

PPP-Codes in a Nutshell

1. data space is partitioned multiple-times by recursive Voronoi
2. based on these partitionings, objects are mapped onto memory codes
   - a dynamic memory index is created using these codes
3. given query q, multiple ranked candidate sets are generated
4. these candidate rankings are merged using median of individual ranks
   - the merged candidate set is smaller and more accurate
5. the final candidate set is retrieved and refined
   - object-by-object, objects stored in a ID-object store



Candidate Aggregation Principle

Given query q, Voronoi cells from each of the λ partitioning are ranked

- λ candidate rankings ψ_q^j of object IDs to be aggregated

$q \in \mathcal{D}$

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

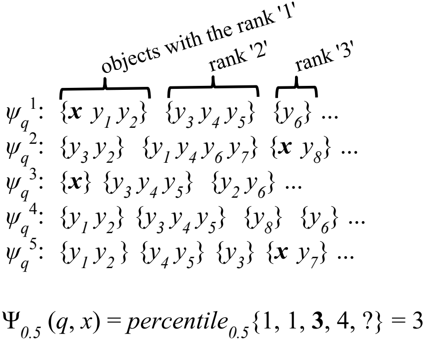
Candidate Aggregation Principle

Given query q, Voronoi cells from each of the λ partitioning are ranked

- λ candidate rankings ψ_q^j of object IDs to be aggregated
- final rank of object x is **p**-percentile (e.g. median) of its λ ranks

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

$q \in \mathcal{D}$

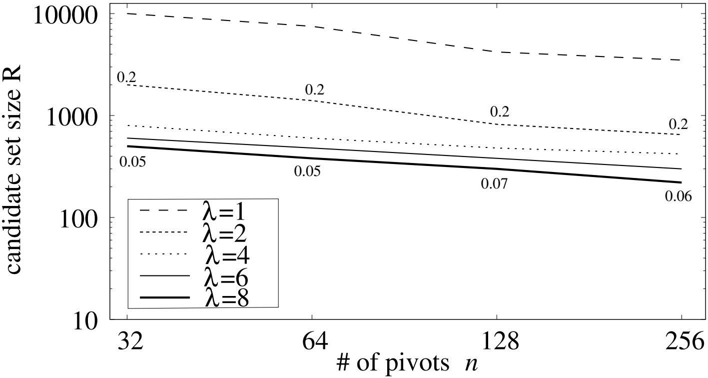


Evaluation 1: Accuracy of the Candidate Set

How many candidate objects are needed to achieve certain recall level

Evaluation 1: Accuracy of the Candidate Set

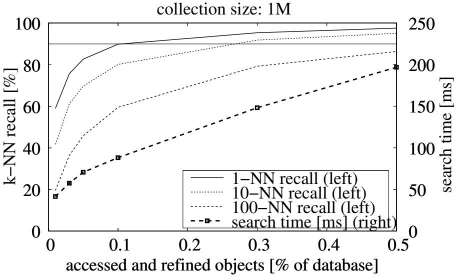
How many candidate objects are needed to achieve certain recall level



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

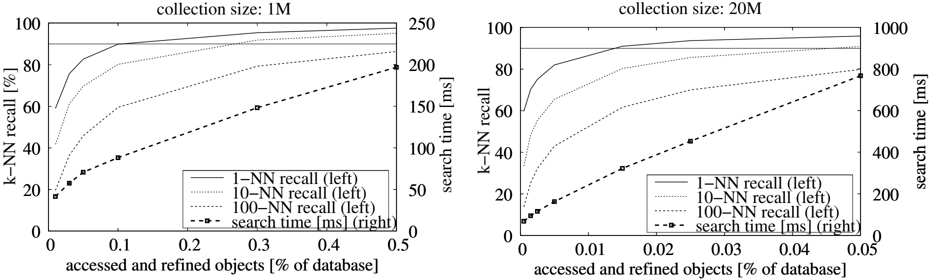
Settings: 1M CoPhIR dataset, $l = 8$ and $\mathbf{p} = 0.75$

Evaluation 2: Overall Efficiency



- 4096-dimensional float vectors with L_2 distance

Evaluation 2: Overall Efficiency



- 4096-dimensional float vectors with L_2 distance
- shrinking the candidate set size by one or two orders of magnitude
  - while preserving the answer quality
  - the larger the data collection, the lower the percentage

Deep Convolutional Neural Networks

▸ a separate Google docs presentation

Online Demonstration

Demonstration of large-scale image visual search

- 20 million images from a photo stock company
- features from deep convolutional neural networks
  - 4096-dimensional vectors with L_2 distance
- collection was recently released for research purposes
  - `http://disa.fi.muni.cz/profiset/`

Online Demonstration

Demonstration of large-scale image visual search

- 20 million images from a photo stock company
- features from deep convolutional neural networks
  - 4096-dimensional vectors with L_2 distance
- collection was recently released for research purposes
  - http://disa.fi.muni.cz/profiset/
- PPP-Codes index (1 GB in memory, 124 GB on the SSD disk)

http://disa.fi.muni.cz/demos/profiset-decaf/

front-end temporarily running at

http://cybela12.fi.muni.cz:8888/demos/profiset-decaf/

Novak, D., Batko. M. and Zezula, P. (2015). Large-scale Image Retrieval using Neural Net Descriptors. Presented at SIGIR 2015.