TECHNISCHE UNIVERSITÄT DRESDEN

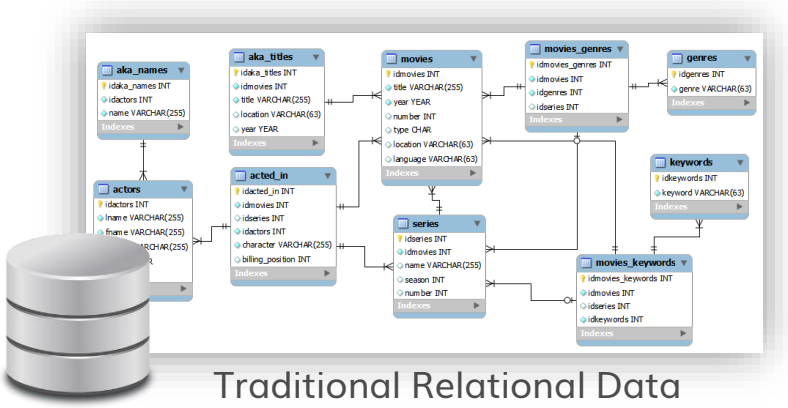
Dresden Database Systems Group

Fast Approximated Nearest Neighbor Joins For Relational Database Systems

Michael Günther, Maik Thiele, and Wolfgang Lehner

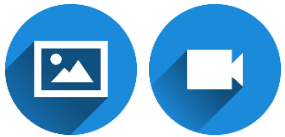
BTW 2019, 07.03.2019

Motivation

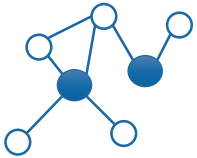


Traditional Relational Data

Unstructured Text

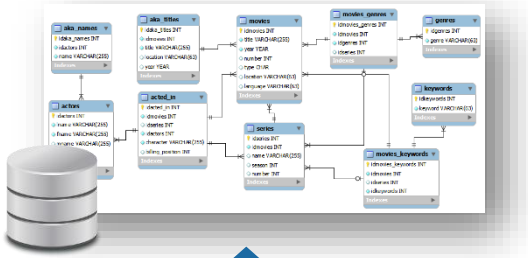


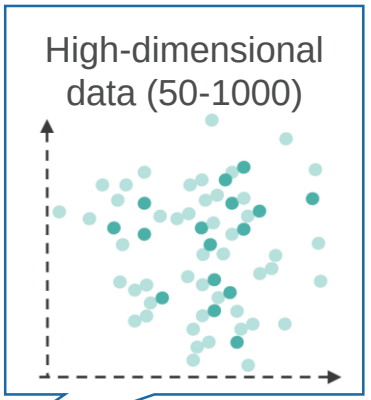
Multimedia Data



Graph Data

Motivation





Vector Representation

Word2Vec / Glove	SIFT / GIST	Deepwalk / Node2Vec
Word Embeddings	Image Descriptors	Graph Embeddings

Unstructured Text

Multimedia Data

Graph Data

Motivation (2)

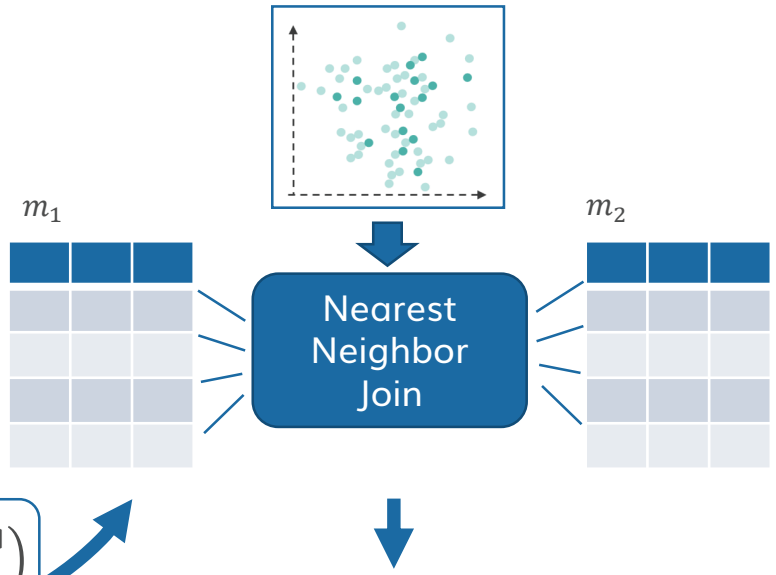
Dresden Database Systems Group

Word Embedding Operations within SQL

Movies Relation

m_id	title	...
1	Inception	...
2	Godfather	...
3	Forrest Gump	...
...		
10^6	Scarfac	

$$kNN\left(\begin{matrix} movies.titles\ AS\ m1 \\ movies.titles\ AS\ m2 \end{matrix} \bowtie\right)$$



m1	m2
Inception	Shutter Island
...	...

Bulk operation

→ **millions of vector distance calculations**

Key Operation: kNN

Dresden Database Systems Group

queries

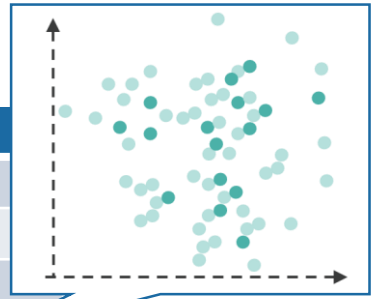
q_1 :
```
SELECT kNN(movies.title,3)
FROM movies
```

database

m_id	title	...
1	Inception	...
2	Godfather	...
3	Forrest Gump	...
...	...	...
10^6	Scarface	...

+

term	vector
2010	[0,21; 0,58; ...; -0,77]
Brando	[-0,46; 0,25; ...; 0,44]
Connery	[0,76; 0,48; ...; -0,51]
...	...
Untouchables	[0,86; -0,22; ...; 0,12]



3 Million Vectors

results

title	VARCHAR
Inception	2010
Inception	Shutter Island
...	...
Godfather	Brando

query set R

Key Operation: kNN-Join

Dresden Database Systems Group

queries

q_2:

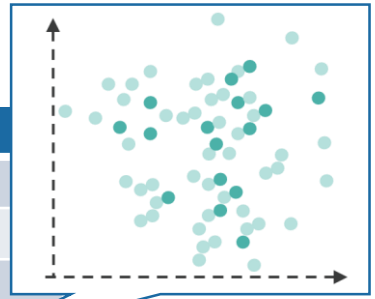
```
SELECT m1.title, m2.title
FROM movies m1 kNN-Join(3) movies m2
ON m1.title ~ m2.title
WHERE m1.year= … AND m2.genre IN (…)
```

database

m_id	title	…
1	Inception	…
2	Godfather	…
3	Forrest Gump	…
…	…	…
10^5	Scarface	…

+

term	vector
Shutter Island	[-0,46; 0,12; …; 0,87]
Forrest Gump	[0,16; 0,62; …; -0,19]
The Dark Knight	[0,76; 0,48; …; -0,51]
…	…
Untouchables	[0,86; -0,22; …; 0,12]



100.000 Vectors

results

title	VARCHAR
Inception	Shutter Island
Inception	The Dark Knight
…	…
Scarface	Untouchables

query set R

target set T

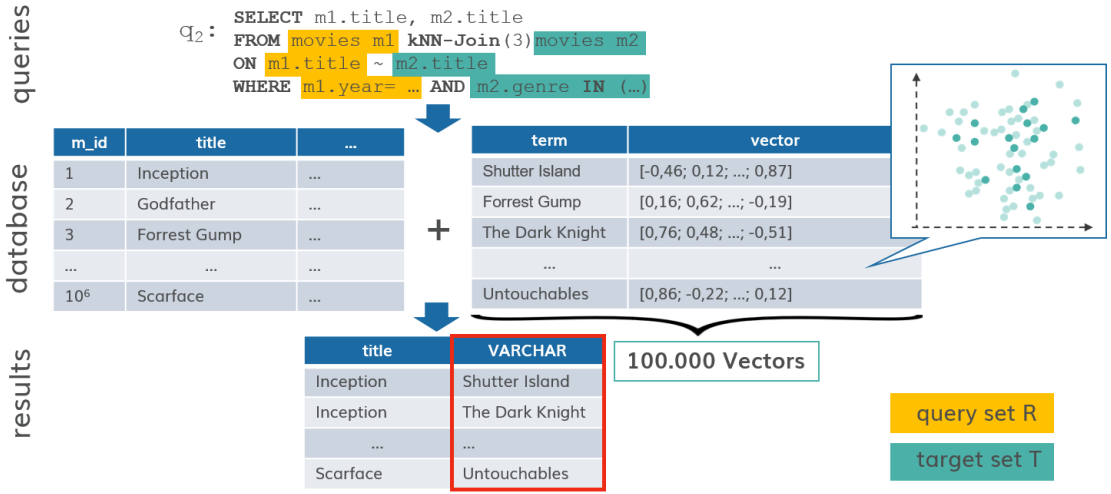
$$kNN(R \ltimes T) = \{(\boldsymbol{r}, \boldsymbol{t}) | \, \boldsymbol{t} \in kNN(\boldsymbol{r}, T), \boldsymbol{r} \in R\}$$

kNN-Joins in RDBMS

Dresden Database Systems Group

Challenges

- **Batch-wise kNN Search** for large query sets
  → reduce interface and retrieval times
- **High Dimensional Data**
  → Previous Work mainly focus on low dimensional data (e.g. spatial data)
- **Adaptive kNN-Join Algorithm**
  → Different cardinalities of join operands
  → Only one index for all vectors
- **Different Demands on Precision and Response Time**



$$kNN(R \ltimes T) = \{(\boldsymbol{r}, \boldsymbol{t}) | \, \boldsymbol{t} \in kNN(\boldsymbol{r}, T), \boldsymbol{r} \in R\}$$

kNN-Join Algorithm

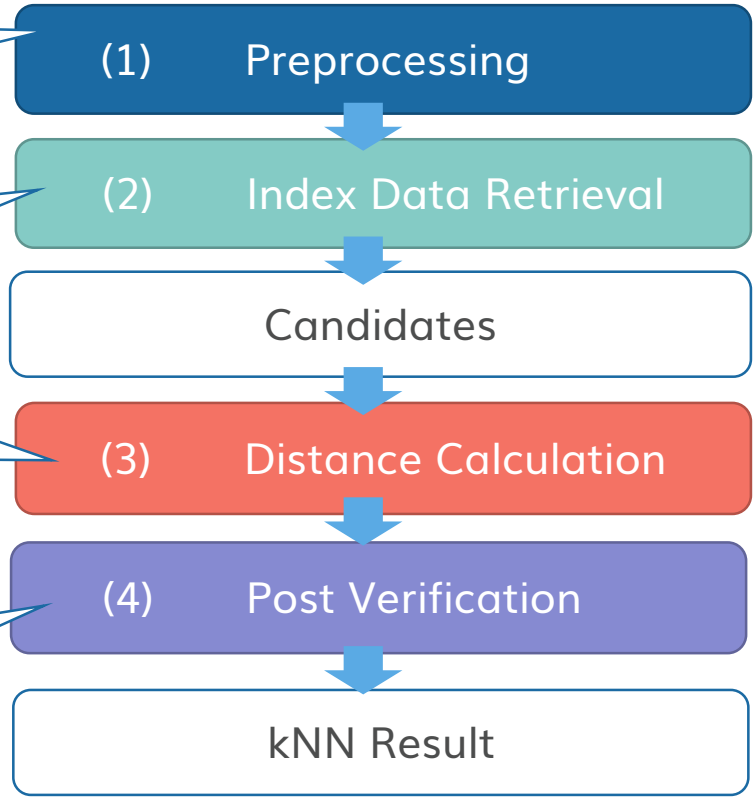
Dresden Database Systems Group

Precomputing data for distance calculation

Read out relevant data from the index (candidates)

Calculate distances between query vectors and candidates to determine most similar targets

Refine the result with exact distance calculation



Approximated kNN Search

K Nearest Neighbors

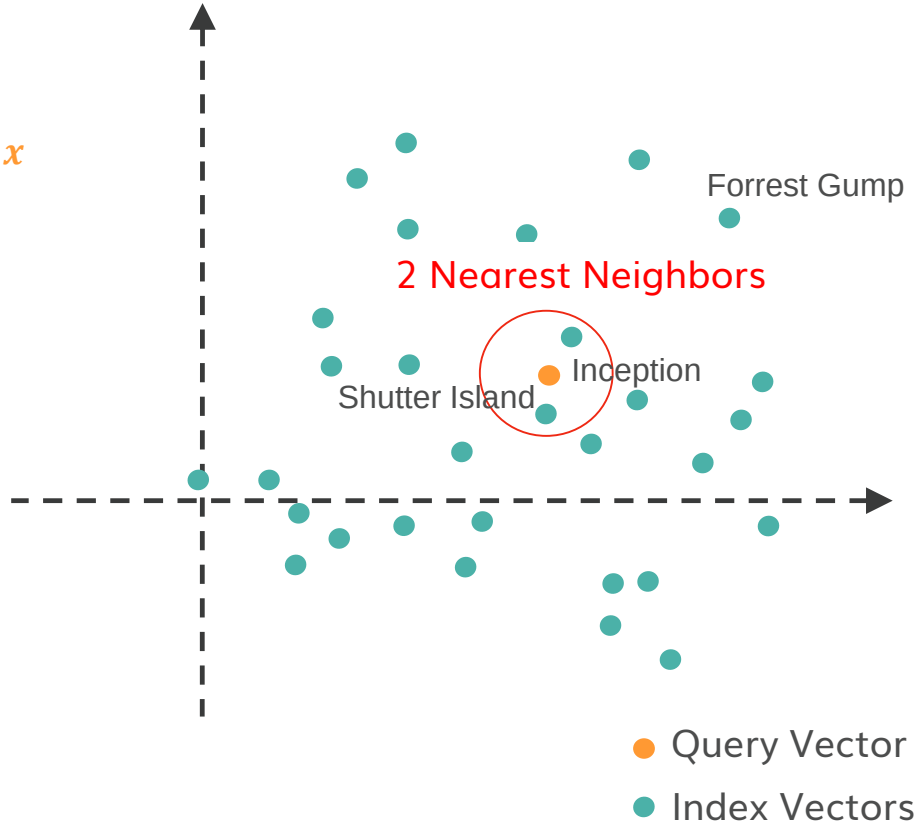
Dresden Database Systems Group

Naïve Algorithm

- Determine all distances between **query vector x** **vectors $p \in P$** in the target set

  → Select vectors with lowest distances (kNN)

- Complexity: $O(|P| \cdot D)$



Vector Quantization

Dresden Database Systems Group

Objective

- Transform vector data into **compact representation**

Quantization

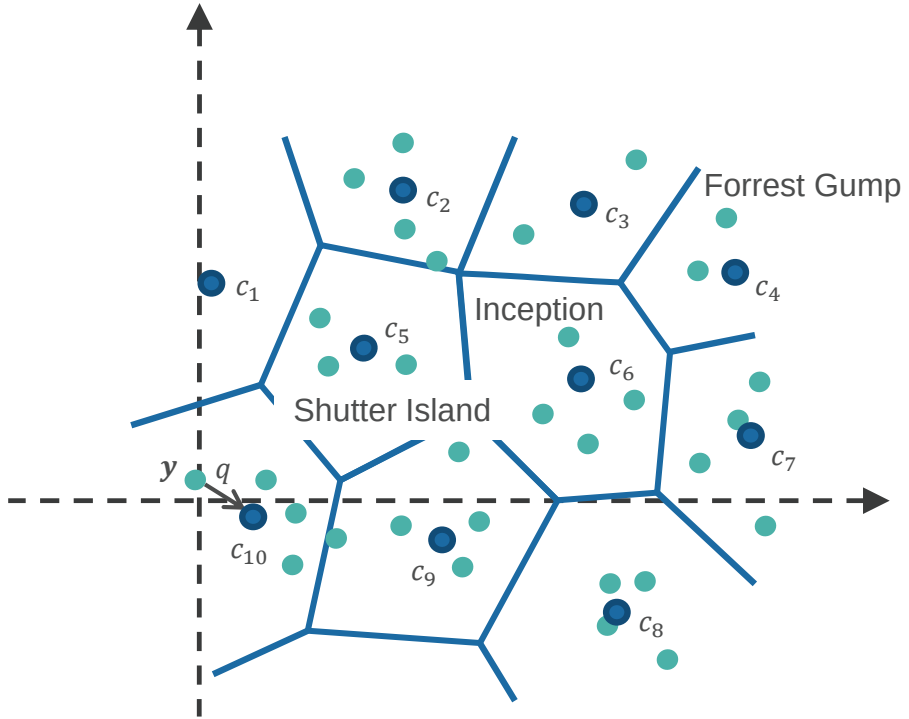
- Cluster Data: kMeans
- **Objective:** Obtain a set of centroids which minimize the distortion:
  → Minimize $\mathrm{d}(\boldsymbol{y}, q(\boldsymbol{y}))$

Quantization Function

- Def.: $q: \boldsymbol{y} \in \mathbb{R}^D \rightarrow C, C \subset \mathbb{R}^D$

  Assign every vector to its nearest neighbor out of a set of centroids

  Represent vector by centroid id



$q(\boldsymbol{y}) = c_{10} \rightarrow \mathrm{id}: 10$

Product Quantization Search

Dresden Database Systems Group

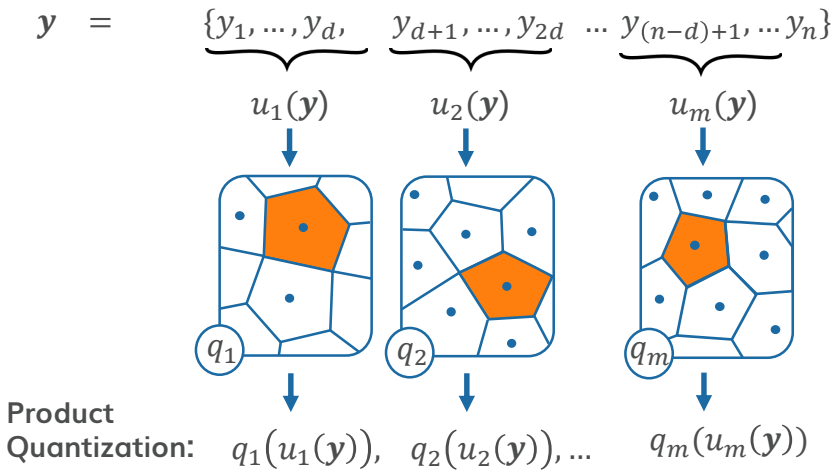
Fast Computation of kNN via Pre-Processing

- Approximated Distances

$$\hat{d}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_j d(u_j(\mathbf{x}), q_j(u_j(\mathbf{y})))^2}$$

- Precompute

$$d\left(\underbrace{u_j(\boldsymbol{x})}_{\text{Query Sub Vector}}, \underbrace{q_j\left(u_j(\boldsymbol{y})\right)}_{\text{Centroid of Quantizer}}\right)^2$$



→ *Computation of square distances resolves to a sum of m precomputed distances*

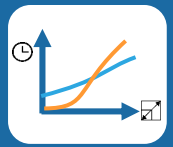
Optimized kNN-Join

Product Quantization Search

Dresden Database Systems Group

(1) Preprocessing can be time consuming

- Should be adaptive to cardinalities of query and target set



(2) Index Data Retrieval

Candidates

(3) Distance Calculation

(4) Post Verification

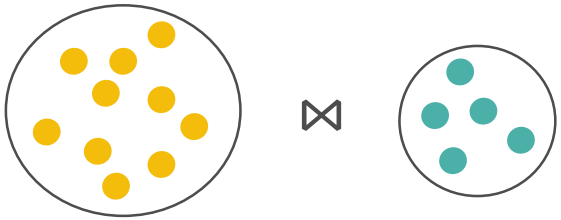
kNN Result

Adaptive Preprocessing

Dresden Database Systems Group

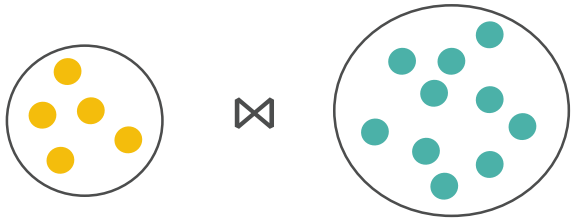
QUERY:

```
SELECT m1.title, m2.title
FROM movies m1 kNN-Join(3)movies m2
ON m1.title ~ m2.title
WHERE m1.year= … AND m2.genre IN (…)
```



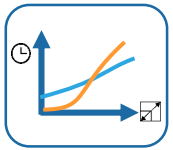
Large Query Set ⋈ Small Target Set

→ **Objective:** Safe preprocessing time



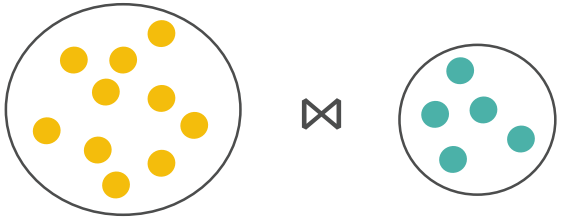
Small Query Set ⋈ Large Target Set

→ **Objective:** Safe time during search



Adaptive Preprocessing

Dresden Database Systems Group



Large Query Set ⋈ Small Target Set

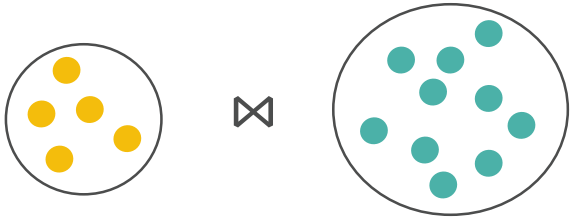
→ **Objective:** Safe preprocessing time

few centroids **but** many quantizer functions

LONG Codes

Number of centroids: 8×10
PQ-Sequences e.g.: $1, 7, 0, 5, 6, 2, 6, 1, 1, 4$

Fast preprocessing
(80 distance calculations per query)
Slower search
(10 additions per query-target-pair)



Small Query Set ⋈ Large Target Set

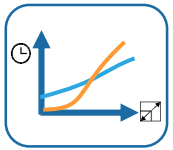
→ **Objective:** Safe time during search

many centroids **but** only few quantizer functions

SHORT Codes

Number of centroids: 64×5
PQ-Sequences e.g.: $15, 5, 50, 49, 12$

Slow preprocessing
(320 distance calculations per query)
Faster search
(5 additions per query-target-pair)



Adaptive Preprocessing

Dresden Database Systems Group

Large Query S… …get Set

→ **Objective:** … …search

It is only **a single index** necessary!

few centroids **but** … …antizer functions

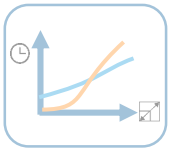
LONG Codes

Number of centroids: …
PQ-Sequences e.g.: 1, 7, 0, 5, 6, 2, 6, 1, 1, 4

Fast preprocessing
(80 distance calculations per query)
Slower search
(10 additions per query-target-pair)

PQ-Sequences e.g.: 15, 5, 50, 49, 12

Slow preprocessing
(320 distance calculations per query)
Faster search
(5 additions per query-target-pair)



Adaptive Preprocessing (2)

Dresden Database Systems Group

Evaluation Setup

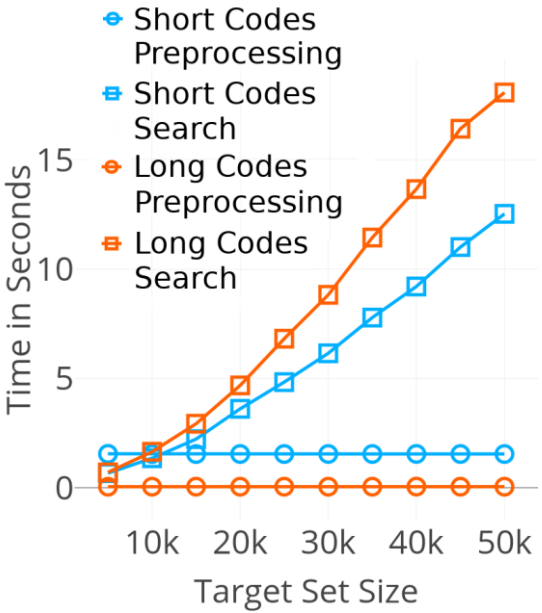
- kNN-Joins with different target set sizes
- Vectors are sampled randomly
- Measurements are done 10 times for each point with different vectors

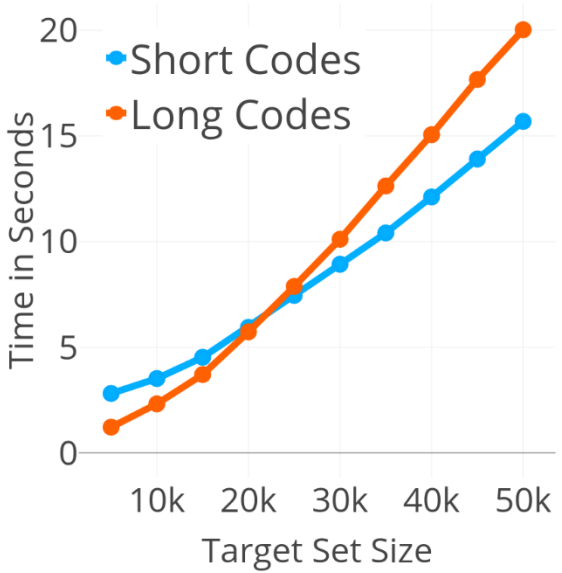
Dataset

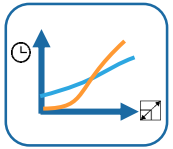
- 3M 300-dimensional word vectors

Parameters

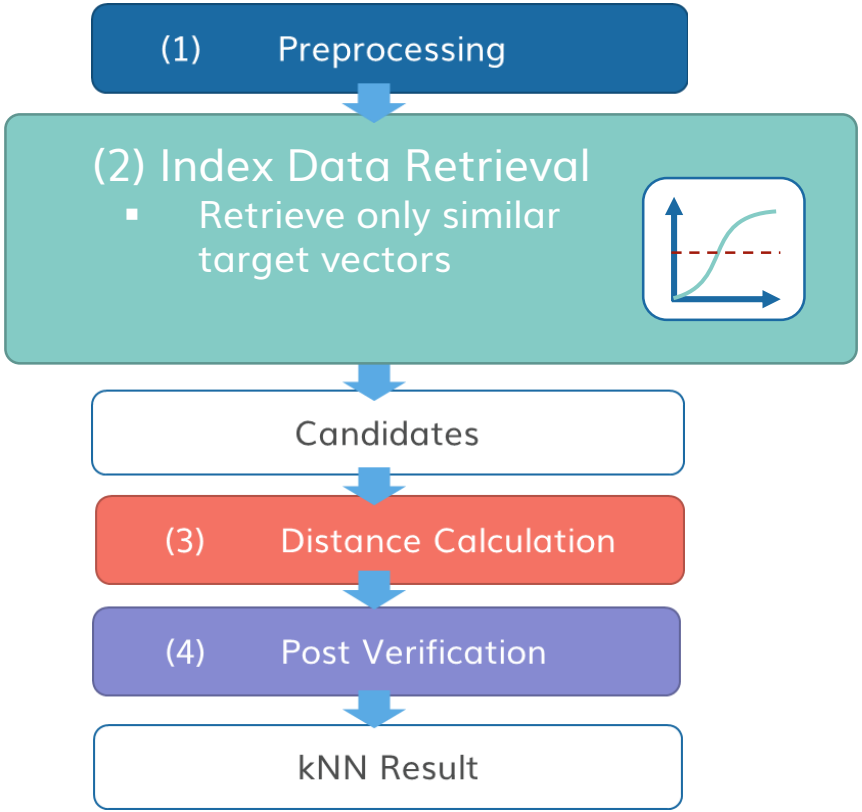
- Query Set Size: 5,000
- K: 5







kNN-Join Algorithmus



Inverted Indexing

Dresden Database Systems Group

Idea

Accelerate kNN-Search by **non-exhaustive** search schema

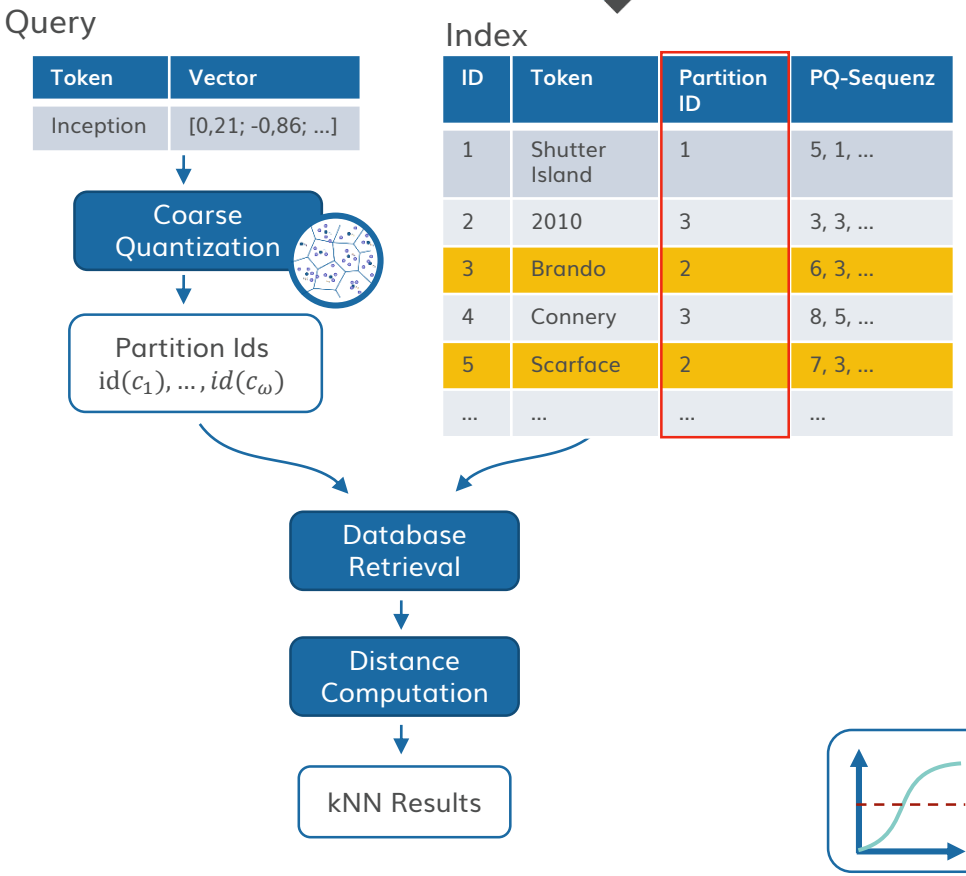
Preprocessing

Additional *coarse quantizer* q_c is applied to the complete vectors

→ Assigns vectors to partitions

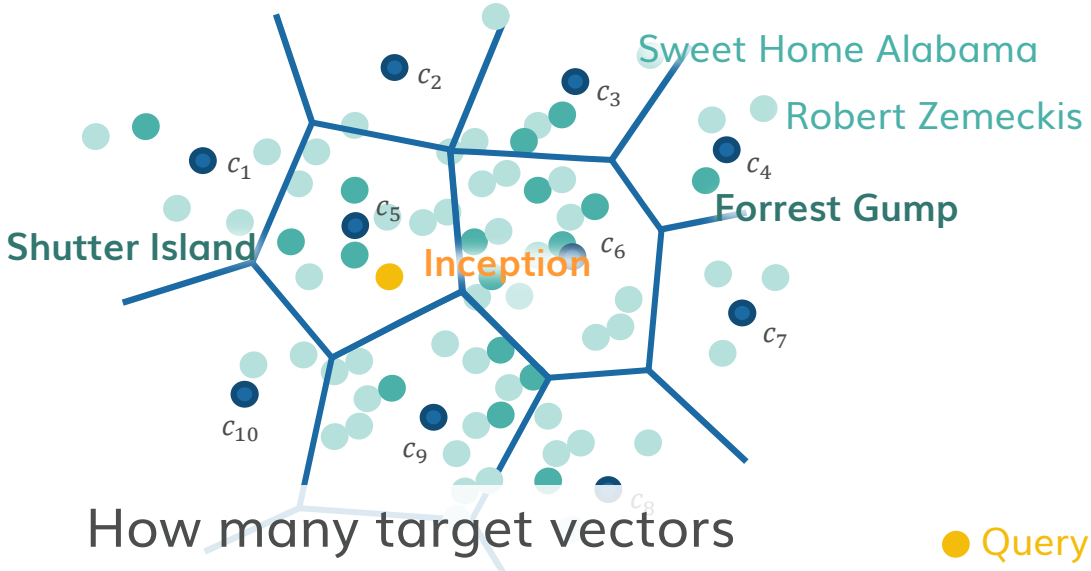
kNN Calculation

- Coarse quantization of query vector is calculated → $\mathrm{id}(c_1), \ldots, id(c_\omega)$
  ….ω determine number of candidates
- Partitions with PID ∈ $\{\mathrm{id}(c_1), \ldots, id(c_\omega)\}$ are retrieved
- Distances Calculation → kNN



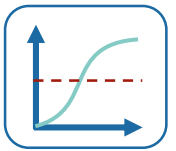
Query Construction

Dresden Database Systems Group



How many target vectors are in the ω nearest partitions?

- Query Vector
- Target Set
- Not in Target Set



Query Construction

Dresden Database Systems Group

Problem

- Number of partitions to retrieve depends on target set size

Estimate Probability of 'Getting at least n targets'

→ Hypergeometric Distr. (draw without replacement)

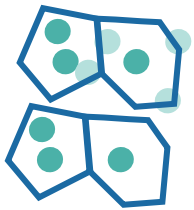
$$P(m \geq n) = \sum_{n \leq m \leq d} \frac{\binom{K}{m}\binom{N-K}{d-m}}{\binom{N}{d}}$$

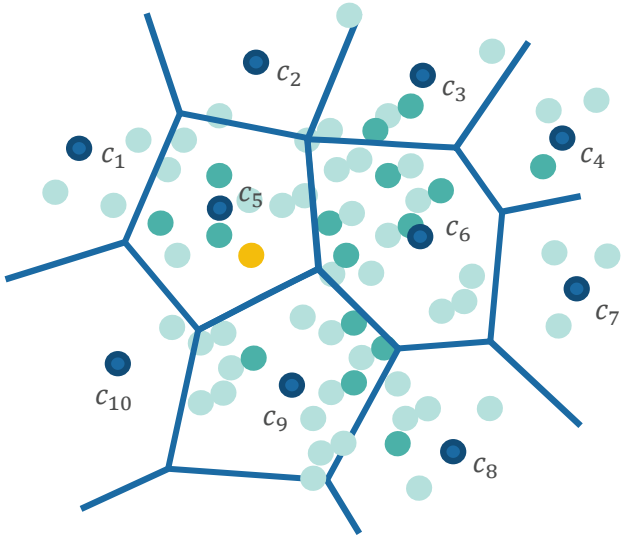
Parameter

- Population Size $\boldsymbol{N}$:
  Number of all index vectors

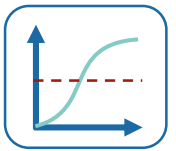
- Number of success states in the population $\boldsymbol{K}$:
  Number of target vectors

- Number of draws $\boldsymbol{d}$:
  Number of vectors in selected partitions
- Number of Successes $\boldsymbol{m}$:
  Number of targets in selected partitions





- Query Vector
- Target Set
- Not in Target Set



Query Construction

Dresden Database Systems Group

Problem

- Number of partitions to retrieve depends on target set size

Estimate Probability of 'Getting at least n targets'

→ Hypergeometric Distr. (draw without replacement)

Parameter

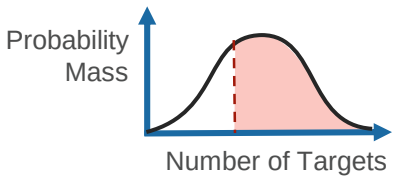
- Population
  Number of
- Number of
  Number of
- Number of
  Number of vectors in selected partitions
- Number of Successes m :
  Number of targets in selected partitions

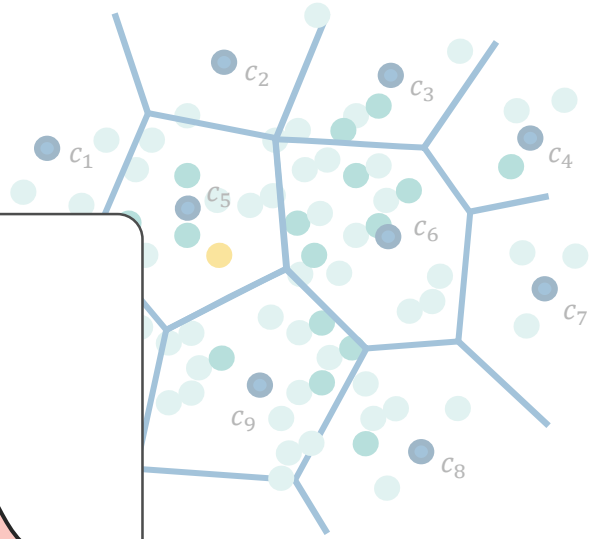
Calculation is very expensive!

→ Approximation via **normal distribution**

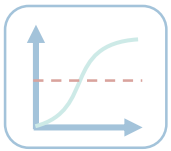
→ Integral of the normal distribution describes the confidence of 'getting at least n targets'

→ **Statistics** of partition sizes for accurate estimation





Query Vector
Target Set
Not in Target Set



Query Construction (2)

Dresden Database Systems Group

Setup

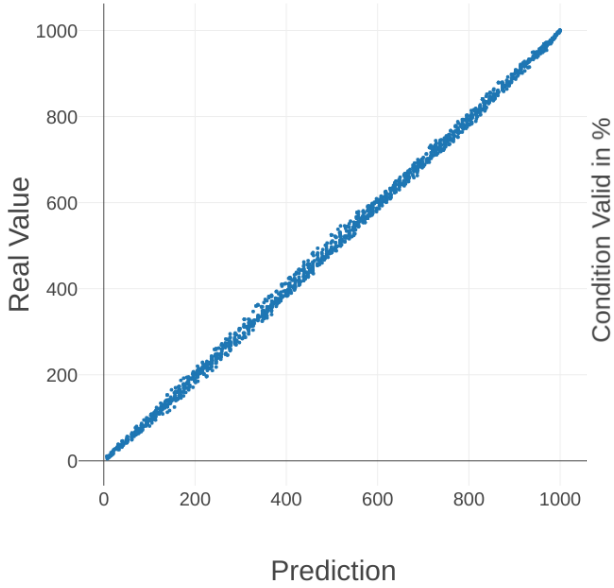
- kNN-Joins with a **single** randomly sampled **query vector** and 1000 target vectors are executed

Estimation of Target Numbers

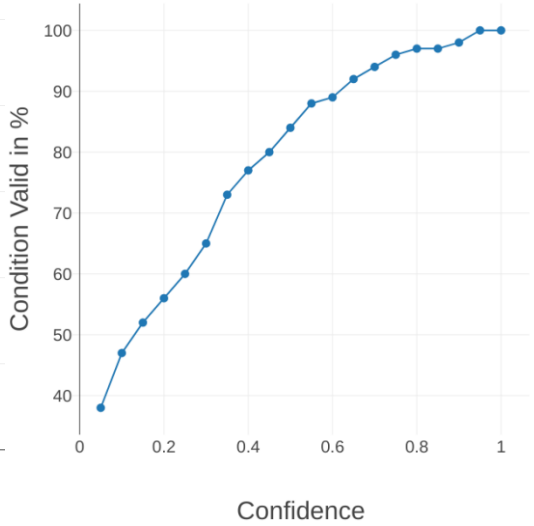
- Different selectivity values
- Prediction of the normal distribution displayed

Estimation of Confidence

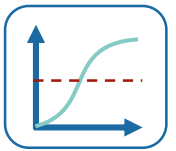
- Number of partitions is adjusted to a probability of getting at least n targets > *confidence*
- Plot shows validity of condition (each rate based on 1000 queries)



Estimated Number of Targets (Mean of the Distribution)

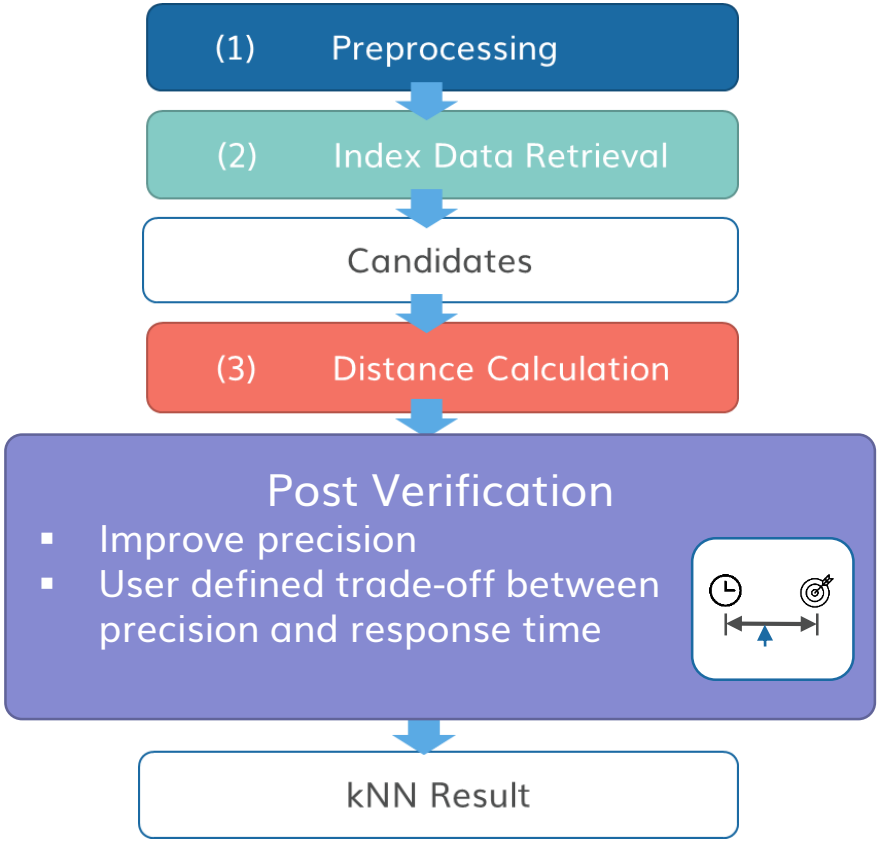


Probability Estimation



kNN-Join Algorithm

Dresden Database Systems Group



Distance Calculation with Post Verification

Dresden Database Systems Group

Combined Distance Calculation – Post Verification

Exact Calculation

very slow

vs

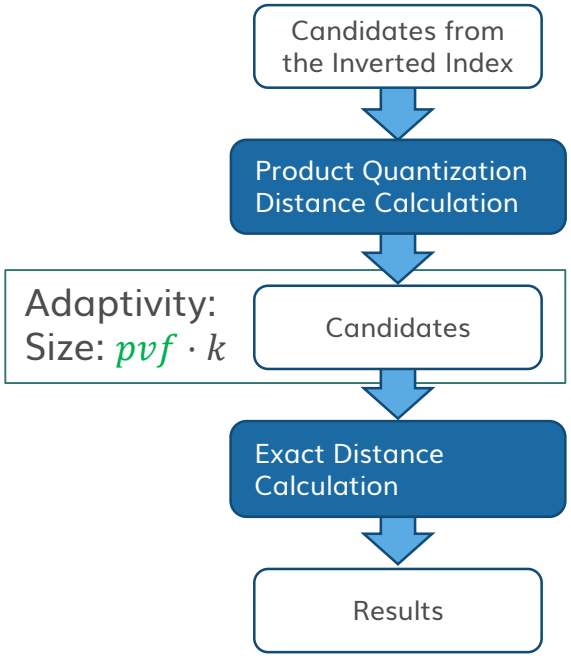
Product Quantization Distance Calculation

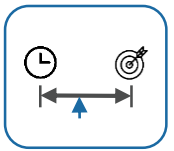
low precision

Post Verification Method

- Product Quantization Methods can be used to obtain **candidate set –** size determined by pvf
- Refine candidate set with exact calculation
- pvf is set based on precision and response time demands

→Post verification converges fast to results of exact calculation





Evaluation

Dresden Database Systems Group

Setup

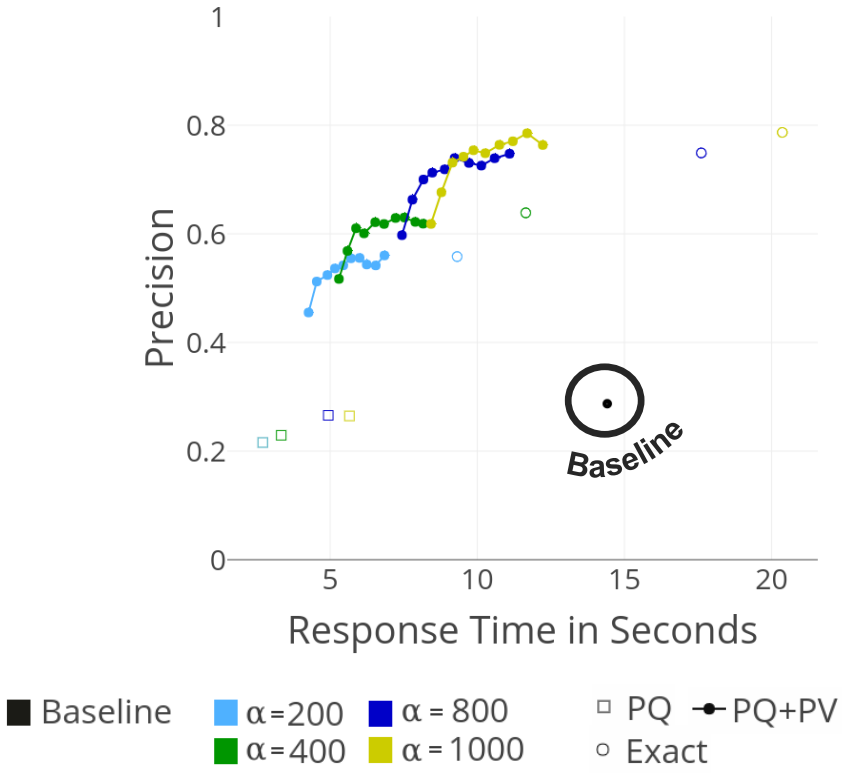
- Time and precisions measurements for randomly sampled word vectors

Dataset

- 3M 300-dimensional word vectors

Parameters

- Number of query vectors: 5,000
- Number of target vectors: 50,000
- K: 5
- Selectivity of the filter α (high values are less selective)
- Different sizes of candidate sets for post verification



Evaluation

Dresden Database Systems Group

Setup

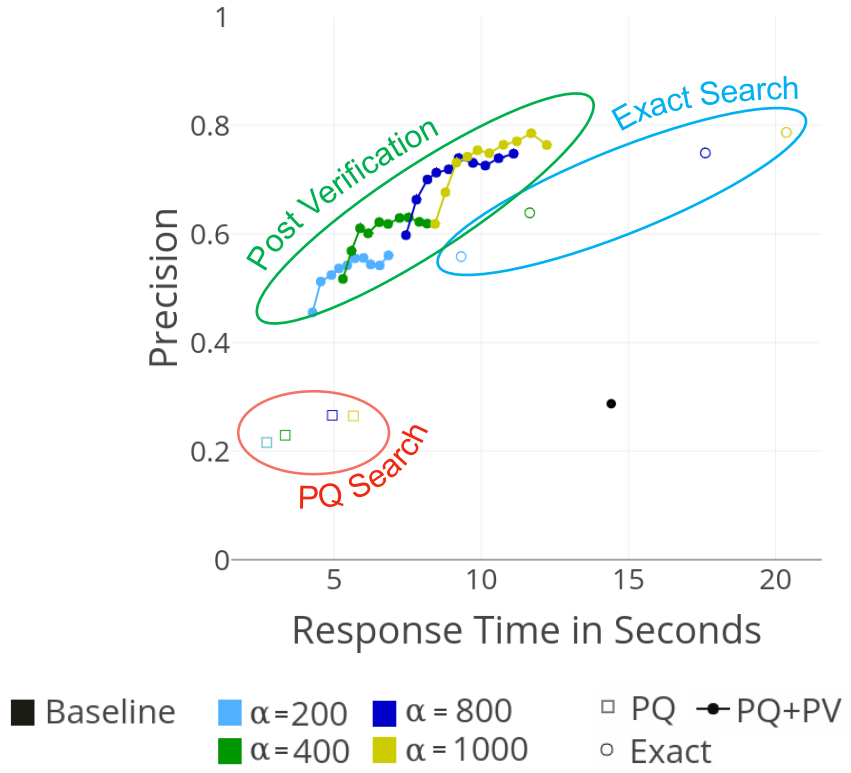
- Time and precisions measurements for randomly sampled word vectors

Dataset

- 3M 300-dimensional word vectors

Parameters

- Number of query vectors: 5,000
- Number of target vectors: 50,000
- K: 5
- Selectivity of the filter α (high values are less selective)
- Different sizes of candidate sets for post verification



Conclusion

Dresden Database Systems Group

Contributions

- Adaptive product quantization search
- **kNN-Join** based **inverted** product quantization search and post verification for **flexible target sets** → faster than the product quantization baseline
- Implementation into a relational database system

Outlook

- Automatic parameter configuration for maximal response time or minimal precision requirements