Efficient Reverse k-Nearest Neighbor Search in Arbitrary Metric Spaces

SIGMOD 2006
Chicago, U.S.A.

Elke Achtert, Christian Böhm, Peer Kröger, Peter Kunath, Alexey Pryakhin, Matthias Renz

Database Group
Institute for Computer Science
University of Munich, Germany
Outline of the Talk

1. Introduction
2. kNN Distance Approximations
3. Optimal Approximation Line
4. Experimental Evaluation
5. Conclusions and Future Work
Introduction

RkNN query applications

Game Tactics

Merchandising

Franco’s Pizza

Giacomo’s Pizza

Paolo’s Pizza

Giovanni’s Pizza

Pipo’s Pizza
Outline of the Talk

1. Introduction
2. kNN Distance Approximations
3. Optimal Approximation Line
4. Experimental Evaluation
5. Conclusions and Future Work
Problem Definition

Definition: Reverse $k$-Nearest Neighbor Query

Let $DB$ be a database of $n$ metric objects and $k \leq n$:

The set of reverse $k$ nearest neighbors (R$k$NN) of an object $q$ is defined as:

$$RNN_k(q) := \{p \in DB : q \in NN_k(p)\}$$

Idea: to speed up the R$k$NN computation, pre-compute all $k$NN-distances of an object for $k \leq k_{max}$
Problem: a massive storage overhead due to the per-object $k$NN-distance table
Solution: use approximations of the actual $k$NN-distances

- conservative (upper bound, UB) and progressive (lower bound, LB) $k$NN distance approximations for an object $p$

\[
dist(p, q) \leq LB_{k\text{NN-Dist}}(p)
\rightarrow p \text{ true hit, i.e. } p \in RNN_k(q)
\]
Example: $q = q_1$

\[
dist(p, q) \geq UB_{k\text{NN-Dist}}(p)
\rightarrow p \text{ true drop, i.e. } p \notin RNN_k(q)
\]
Example: $q = q_2$

\[
UB_{k\text{NN-Dist}}(p) \leq dist(p, q) \leq LB_{k\text{NN-Dist}}(p)
\rightarrow p \text{ candidate}
\]
Example: $q = q_2$

Conservative and Progressive Distance Approximations
**kNN-Distance Storage Problem**

**Input:** for each object \( p \) a sequence of kNN-distances, 
\[
\langle 1\text{NN-Dist}(p), 2\text{NN-Dist}(p), \ldots, k_{\text{max}}\text{NN-Dist}(p) \rangle
\]
for a sufficient large \( k_{\text{max}} \)

**Problem:** how to store the UB- and LB-approximations of the kNN-distances in a compact way?

- the theory of self-similarity indicates that the relationship between
  - the number of enclosed objects \( \text{encl}(\varepsilon) \) and
  - the radius \( \varepsilon \) of a hypersphere
follows a power law: \( \text{encl}(\varepsilon) \propto \varepsilon^{d_f} \), where \( d_f \) is the fractal dimension
→ the distances approximately form a line in log-log-space:
\[
\log(k\text{NN-Dist}(p)) \propto \log(k) / d_f
\]
Distances in Log-Log Space

Uniformly distributed Points

Points forming two Gaussian Clusters
The MR$k$NNCoP Approach

The distances are approximated by a line in log-log-space → this is much cheaper than storing all $k$NN-distances in the index

**LB- and UB-approximations**

the UB-approximation is represented by a line in log-log-space, where

$$\forall k \leq k_{\text{max}} : k\text{NN} - \text{Dist}(p) \leq \text{UB}_{k\text{NN-Dist}}(p)$$

the LB-approximation is represented by a line in log-log-space, where

$$\forall k \leq k_{\text{max}} : k\text{NN} - \text{Dist}(p) \geq \text{LB}_{k\text{NN-Dist}}(p)$$
Outline of the Talk

1. Introduction
2. kNN Distance Approximations
3. Optimal Approximation Line
4. Experimental Evaluation
5. Conclusions and Future Work
Optimization Goal

- Given the sequence of $k$NN-distances of an object $p$

**Conventional linear regression**
Finds $m$, $t$ of the linear function $L(o)$: $y = m \cdot x + t$ minimizing mean square error
\[
\sum_{1 \leq k \leq k_{\text{max}}} ((m \cdot x_k + t) - y_k)^2 \rightarrow \min
\]

**Optimal conservative approximation**
Finds $m_{\text{opt}}$, $t_{\text{opt}}$ of the linear function $L_{\text{opt}}(o)$: $y_{\text{opt}} = m_{\text{opt}} \cdot x + t_{\text{opt}}$ in log-log-space with the following constraints:

C1 $L_{\text{opt}}(o)$ is a conservative approximation of $y_k$, i.e.
\[
\forall k \leq k_{\text{max}} : y_k \leq m_{\text{opt}} \cdot x_k + t_{\text{opt}}
\]

C2 $L_{\text{opt}}(o)$ minimizes the mean square error
Constraints on the Optimal Line

- Given the upper convex hull \( UCH = \langle (x_{k_1}, y_{k_1}), (x_{k_2}, y_{k_2}), ..., (x_{k_u}, y_{k_u}) \rangle \)
of the approximated point set (2 ≤ \( u \) ≤ \( k_{\text{max}} \))

I. The optimal line must interpolate either two neighboring points or one single point of the \( UCH \)

II. The optimal line must interpolate an anchor point with mean square error

III. The line is optimal if both the successor and the predecessor of the anchor point are under the line
Optimization Algorithm, Initialization

0. Input: sequence of $k$NN distances, $k_{max}$

Compute the $UCH$ of the $k$NN distances in the log-log-space

→ based on Graham’s scan algorithm for the convex hull

1. Perform a search for the optimum approximation line

The bisection search starts with the complete $UCH$ as input
2. Select the median point of the UCH as anchor point and compute the anchor’s optimal line \((aol)\).

3. Inspect the successor and predecessor of the anchor and determine if the left neighbor (predecessor) or the right neighbor (successor) is above \(aol\) → proceed with the left respectively right half of the UCH.
4. Test if either both neighbor points are below \( aol \) or if the \( aol \) interpolates the last two points considered in the interactive search → the global optimum has been reached

Output: optimal approximation line

The slope of the computed line is used to identify the search space of the subsequent search step.
→ In each step of the algorithm, the problem size is divided by two
The conservative and progressive approximations can also be used for the nodes of the index to prune irrelevant sub-trees.

Aggregate for each data node the $k$NN distances of the objects within that node:

- for the maximum $k$NN distances, use the **conservative** approximations $UB_{kNN-Dist}$
- for the minimum $k$NN distances, use the **progressive** approximations $LB_{kNN-Dist}$
RkNN Query Search Algorithm

RkNN_query(D, q, k) // Assumption: D is organized as MRkNNCoP
queue = LIST OF (dist:Real, obj:Object) ORDERED BY dist ASCENDING;
queue.insert((0.0, D.root));
WHILE NOT queue.isEmpty() DO
  N = queue.removeFirst();
  IF N.isNode() THEN
    IF mindist(q, N.getRegion()) ≤ UB_{kNN-Dist}(N.getRegion()) THEN
      FOR i=0 TO N.size() DO
        queue.insert(((mindist(q, N.getElement(i)), N.getElement(i)));
    ELSE // N is a point
      FOR i=0 TO N.size() DO
        IF dist(q, N.getElement(i)) < LB_{kNN-Dist}(N.getElement(i)) THEN
          add N to result set;
        ELSE IF dist(q, N.getElement(i)) ≤ UB_{kNN-Dist}(N.getElement(i)) THEN
          add N to candidate set;
      END WHILE
  END IF
END WHILE
Outline of the Talk

1. Introduction
2. kNN Distance Approximations
3. Optimal Approximation Line
4. Experimental Evaluation
5. Conclusions and Future Work
Experimental Evaluation

Road Network

Metric data set
6105 objects

Sequoia 2000

Euclidean data set
100000 feature vectors
5 dimensions

Corel

Euclidean data sets
68040 feature vectors
9 and 16 dimensions

→ efficiency and effectiveness evaluation of the proposed MRkNNCoP approach
Runtime results (Oldenburg data set)

Varying database size
\((k = 50, k_{max} = 100)\)

Varying parameter \(k\)
\((2500 \text{ objects}, k_{max} = 150)\)


Pruning capability for varying parameter $k$ ($k_{\text{max}} = 100$)

Oldenburg data set (5000 objects)

Color Textures (30000 objects)

$\rightarrow$ the conservative approximation is a very effective upper bound

the progressive approximation is a very effective lower bound
MRkNNCoP outperforms the TPL competitor by up to 40%
the performance gap of MRkNNCoP vs TPL increases with increasing $k$
Outline of the Talk

1. Introduction
2. kNN Distance Approximations
3. Optimal Approximation Line
4. Experimental Evaluation
5. Conclusions and Future Work
Conclusions and Future Work

Our Approach:

- RKNN method that works for both metric and euclidean data spaces
- Exploits the high pruning power of the $k$NN distances
- The $k$NN distances are approximated conservatively and progressively by a cheap function to avoid storage overhead

Future Work:

- Develop data structures for parallel and distributed RKNN queries
- Extend our approach to approximative RKNN queries