Answering Why-Not Questions on Spatial Keyword Top-k Queries

Lei Chen†, Xin Lin*, Haibo Hu†, Christian S. Jensen**, Jianliang Xu†
† Hong Kong Baptist University, *East China Normal University, **Aalborg University

Introduction
Spatial keyword top-k queries find top-k results by considering both spatial and textual attributes of the objects.

Why Oakwood not returned?
- Top-2 hotels around COEX
  - Intercontinental
  - Park Hyatt
- Why Oakwood not returned?
  - Use larger k?
  - \( \text{SDist} \bowtie \text{Tsim} \)?

Understanding why may aid users in retrieving better results.

Preliminaries
Spatial Keyword Top-k Query
- Data: A set of spatial-textual objects \((o, \text{loc}, o, \text{doc})\)
- \(Q(q, \text{loc}, q, \text{doc}, k, \vec{w})\)
- Scoring function:
  \[ f(q, o) = w_1 \ast (1 - \text{SDist}(q, o)) + w_2 \ast \text{TSim}(q, o) \]
- Higher score, higher rank

Why-Not Spatial Keyword Top-k Query
- An initial spatial keyword top-k query
  \[ Q(q, \text{loc}, q, \text{doc}, k_0, \vec{w}_0) \]
- A set of missing objects
  \[ \{o_1, o_2, \ldots, o_m\} \]
- “Why-not” returns a refined query minimizing the penalty
  \[ Q'(q, \text{loc}, q, \text{doc}, k', \vec{w}') \]
- Penalty function
  \[ \text{penalty}(k', \vec{w}') = \lambda \cdot \frac{1}{\text{SDist}(q, o)} - k_0 + (1 - \lambda) \cdot \frac{\vec{w}}{\sqrt{1 + \|\vec{w}_0\|^2}} \]
  \[ \Delta k = \max(0, k' - k_0) \] and \( \Delta \vec{w} = \|\vec{w}' - \vec{w}_0\|_2 \)

Geometry Property
- Fixed-score segment for each object \(o\)
  \[ w_1 \ast (1 - \text{SDist}(q, o)) + w_2 \ast \text{DSim}(q, o) = C \]
  \(C\) is a positive constant
- Property of fixed-score segments
  - Along a direction \(\vec{w}\), the closer the object’s fixed score segment to the origin, the higher the rank.
  - Rank changes rules
    - Degraded points: \(P_1, P_3\)
    - Promoted points: \(P_2, P_4\)
  - The weighting vector of the best refined query must point to one of the promoted points (if not \(\vec{w}_0\)).

Basic Algorithm
- Determine the missing object’s rank under the initial query;
- Use a range query to find each object \(o'\) that satisfies \(\text{Dist}(o', q) < \text{SDist}(o, q) \land \text{DSim}(o', q) < \text{DSim}(o, q)\) or \(\text{Dist}(o', q) > \text{SDist}(o, q) \land \text{DSim}(o', q) > \text{DSim}(o, q)\);
- Compute the ranks under all promoted points;
- Compute the penalties of all promoted points and the initial weighting vector, return the one with the smallest penalty.

Optimized Bound and Prune Method
- BIR-tree (Bounded IR-tree)
  Each node in the BIR-tree stores: \(ctn\)
  Each term \(t\) in pseudo documents stores: \(w_{\text{min}}\) and \(w_{\text{max}}\)

Bound and prune algorithm using BIR-tree
- Estimate the number of degraded points without unfolding a BIR-tree node

Performance Results
- Tested algorithms
  - Baseline, TM (Testing Model), BS (Basic Algorithm), BP (Bound and Prune Algorithm)
- Real-world datasets
  - Euro: 162,033 objects, 35,315 distinct words
  - GN: 1,868,821 objects, 222,407 distinct words
- Varying parameters
  - \(k_0\) # keywords, \(\vec{w}_0\), \(\lambda\), # missing objects