The Web Is (Mostly) Mobile

- A quickly evolving mobile Internet infrastructure.
  - Mobile devices, e.g., smartphones, tablets, laptops, navigation devices
  - Communication networks and users with access
- Rapidly increasing device sales (millions)
- Mobile is a mega trend.
  - Google went “mobile first” in 2013.
  - Mobile data traffic 2020 = mobile data traffic 2010 x 1000

Mobile Is Spatial

- Increasingly sophisticated technologies enable the accurate geo-positioning of mobile users.
  - GPS-based technologies
  - Positioning based on Wi-Fi and other communication networks
  - New technologies are underway (e.g., GNSSs and indoor).

Outline

- Background and motivation
- Top-k spatial keyword queries
- Continuous top-k queries
- Accounting for co-location
- Aggregate queries, including collective and group queries
- Summary and challenges

(Acknowledgments and references are given at the end.)

Spatial Web Querying

- Total web queries
  - Google: 2011 daily average: 4.7 billion (uncertain)
- Queries with local intent
  - Google: ~20% of desktop queries
  - Bing: 50% of mobile queries
- Vision: Improve web querying by exploiting accurate user and content geo-location
  - Smartphone users issue keyword-based queries
  - The queries concern websites for places
- Balance spatial proximity and textual relevance
- Support different use cases

Spatial Web Objects

- Objects: \( p = (\lambda, \psi) \) (location, text description)
- Example:

\[
\lambda = (56.158889, 10.191667)
\]

\( \psi \) = Den Gamle By Open-Air Museum

Den Gamle By, "The Old Town" was founded in 1909 as the world’s first open-air museum of urban history and culture.
Spatial Web Objects — Sources

- Web pages with location
- Online business directories
  - Business name, location, categories, reviews, etc.
  - Example: Google Places
- Geocoded micro-blog posts
  - Example: Twitter
  - Messages with up to 140 characters.

Top-k spatial keyword querying

Top-k Spatial Keyword Query

- Objects: \( p = \{l, d\} \) (location, text description)
- Query: \( q = \{l, k\} \) (location, keywords, # of objects)

Ranking function

\[
rank(p) = \alpha \frac{\|q \cdot p\|_2}{\max D} + (1-\alpha)(1 - \frac{tr(p|q)}{\max P})
\]

- Distance: \( \|q \cdot p\|_2 \)
- Text relevancy: \( tr(p|q) \)
  - Probability of generating the keywords in the query from the language models of the documents

- Generalizes the kNN query and text retrieval

Spatial Keyword Query Processing

- How do we process spatial keyword queries efficiently?

Proposal

- Prune both spatially and textually in an integrated fashion
- Apply indexing to accomplish this

The IR-tree [Cong et al. 2009; Li et al. 2011; Wu et al. 2012]

- Combines the R-tree with inverted files
- R-tree: good for spatial
- Inverted files: good for text
### Why Not Top-k Spatial Keyword Query I

$$f(q,p) = \alpha (1 - SDist(p,q)) + (1 - \alpha) TSim(q,p)$$

- Top-2 “clean/comfortable” hotels near COEX
  - Rank 1: Intercontinental
  - Rank 2: Oakwood
- Rank 3: Park Hyatt (not returned)

Refined query:
- Use larger k?
- Set k to 3 or larger
- Modify both $$\alpha$$ & k?

<table>
<thead>
<tr>
<th></th>
<th>1-SDist()</th>
<th>TSim()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercontinental</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>Oakwood</td>
<td>0.7</td>
<td>0.6</td>
</tr>
<tr>
<td>Park Hyatt</td>
<td>0.3</td>
<td>0.9</td>
</tr>
</tbody>
</table>

### Why Not Top-k Spatial Keyword Query II

- Top-2 “Clean, Comfortable” hotels near Conference Venue:
  - Rank 1: Holiday Inn
  - Rank 2: Omena Hotel
  - Rank 3: Raddison Blu (not returned)

Refined query:
- Use a larger k?
- User other query keywords?
- Query with “Clean, Comfortable, Luxury”
- Modify both $$k$$ & $$\alpha$$?

### Continuous Spatial Keyword Queries

- Objects: $$p = \langle z, y \rangle$$ (location and text description)
- Query: $$q = \langle z, y, k \rangle$$ (location, keywords, # of objects)
- A continuous query where argument $$z$$ changes continuously

- Ranking function

$$rank_q(p) = \frac{||q \cdot z, p \cdot z||}{Tr_q(p,y)}$$

- Euclidean distance (changes continuously)
- Text relevancy (query dependent)
Continuous Spatial Keyword Queries

- How can we process such queries efficiently?
  - Server-side computation cost
  - Client-server communication cost

- While the argument changes continuously, the result changes only discretely.
  - Do computation only when the result may have changed

- Use safe zones
  - When the user remains within the zone, the result does not change.
  - The user requests a new result when about to exit the safe zone.

Processing Continuous Queries

- Compute results
  - As before...

- Compute corresponding safe zones
  - Integrate with result computation
  - Prune objects that do not contribute to the safe zone without inspecting them
    - Use the IR-tree
    - Access objects in border-distance order
    - Prune sub-trees
    - Terminate safely when a stopping criterion is met

---

Representation of a Multiplicatively Weighted Voronoi Cell

Influence Objects

$I^+ \cup I^- \cup I^-$
Pruning Objects $p_3$ with Higher Weights

$$\exists p' \in I \left( C_{p_3, p'} \not\supseteq C_{p, p'} \right)$$

Pruning Objects with Equal Weights

$$\exists p' \in I \left( \bot_{p, p'} \supseteq C_{p, p'} \right)$$

$$\exists p' \in I \left( \bot_{p, p'} \supseteq \bot_{p', p'} \right)$$

Pruning Objects with Lower Weights

$$\exists p' \in I \left( C_{p, p'} \cap C_{p', p'} = \emptyset \right)$$

$$\exists p' \in I \left( C_{p, p'} \subseteq C_{p', p'} \right)$$

$$\exists p' \in I \left( C_{p, p'} \cap \bot_{p', p'} = \emptyset \right)$$

**Accounting for Co-Location**

- So far, we have considered data objects as independent, but they are not.

- It is common that similar places co-locate.
  - Markets with many similar stands
  - Shopping centers, districts
  - Shopping malls, malls, mall, mall...
  - Restaurant and bar districts
  - Car dealerships

- How can we capture and take into account the apparent benefits of co-location?

**Prestige-based ranking**
Top-k Spatial Keyword Query

- Objects: \( p = \{ \langle \lambda, \psi \rangle \} \) (location, text description)
- Query: \( q = \{ \langle \lambda, \psi, k \rangle \} \) (location, keywords, # of objects)

Ranking function

\[
pr_{rank}(p) = \alpha \frac{\| q, \lambda, p, \lambda \|}{\max D} + (1-\alpha)(1 - pr_{\psi}(p, \psi))
\]

- Distance: \( \| q, \lambda, p, \lambda \| \)
- Text relevancy: \( pr_{\psi}(p, \psi) \)
  - PR score: prestige-based text relevancy (normalized)

Standard Retrieval Approach

Prestige-Based Retrieval

- Prestige propagation using a graph \( G = (V, E, W) \)
  - Vertices \( V \): spatial web objects
  - Edges \( E \): connect objects that meet constraints
  - Distance threshold: \( \| p, \lambda, p, \lambda \| \leq \delta \)
  - Similarity threshold: \( \| \text{sim}(p, \psi, p, \psi) \| \leq \varepsilon \) (vector space model)
  - Edge weights \( W \): \( \| p, \lambda, p, \lambda \| \)
- Use Personalized PageRank for ranking \cite{JehWidom2003}

Prestige-Based Ranking

- Local experts are asked to provide query keywords for locations and then to evaluate the results of the resulting queries.
- The studies suggest that the approach is able to produce better results than is the baseline without score propagation.

Experimental Study
Digression: Methodology

- The same underlying methodology underlies the studies covered.
- Define precisely a problem of perceived real-world interest.
- Develop solutions
  - Concepts, data structures, algorithms
- Carry out mathematical analyses
  - Correctness, complexity, storage size
- Prototype the solutions and perform empirical studies
  - Often, real data is needed
  - Offers detailed insight in the design properties of the solutions
- Iterate!

Aggregate Spatial Keyword Querying

- So far, the granularity of a result has been a single object.
- We may want to return sets of objects that collectively satisfy a query.

  - Aggregate queries
    - Find a set of objects that collectively satisfy the query
    - Aggregate the result documents into a single document
    - Apply spatial proximity conditions to the result objects internally and with respect to the query

  - Top-k groups queries
    - Find groups of objects that satisfy the query
    - Each object in a group is relevant to the keywords
    - Apply spatial proximity conditions to the result objects internally and with respect to the query

Collective Spatial Keyword Querying

- The spatial aspect offers natural ways of aggregating data objects and providing aggregate query results.

  - We may want to return sets of objects that collectively satisfy a query.
The Collective Spatial Keyword Query

- Query location: 🌟
- Query keywords: theater, gym

Collective Query Variants

- Type 1: cost function:
  \[ \text{Cost}(Q, \mathcal{Y}) = \sum_{o \in \mathcal{Y}} \text{Dist}(o, Q) \]
  
  - Application scenario
    - The user wishes to visit the places one by one while returning to the query location in-between.
    - Go to the hotel between the museum visit and the jazz concert
    - NP-hard: proof by reduction from the Weighted Set Cover problem

- Type 2: Cost function:
  \[ \text{Cost}(Q, \mathcal{Y}) = \max_{o \in \mathcal{Y}} \text{Dist}(o, Q) + \max_{q \in \mathcal{Y}} \text{Dist}(o, o) \]
  
  - Application scenario
    - Visit places without returning to the query location in-between
    - E.g., go to a movie and then dinner
    - NP-hard: proof from reduction from the 3-SAT problem

Approximation Algorithm - T1A1

- Exploit existing well known greedy algorithm
  - Partial query \( q \): the unmatched part of the query keywords \( Q \).

Exact Algorithm Without an Index - T1E1

- Aim: Develop an exact algorithm with a running time that is exponential in the number of query keywords, not the number of objects.
  - The number of query keywords is small.

The Collective Spatial Keyword Query

- Objects: \( o = \{ o_1, o_2, o_3, o_4 \} \) (location and text description)
- Query: \( Q = \{ t_1, t_2, t_3 \} \) (location and keywords)

- The result is a group of objects \( \mathcal{Y} \) satisfying two conditions.
  - \( Q \mathcal{Y} \subseteq \bigcup_{o \in \mathcal{Y}} \text{Dist}(o, Q) \)
  - \( \text{Cost}(Q, \mathcal{Y}) \) is minimized.

- \( \text{Cost}(Q, \mathcal{Y}) = \alpha C_1(\mathcal{Y}) + (1 -\alpha) C_2(\mathcal{Y}) \)
  - \( C_1(\mathcal{Y}) \) depends on the distances of the objects in \( \mathcal{Y} \) to \( Q \)
  - \( C_2(\mathcal{Y}) \) characterizes the inter-object distances among objects in \( \mathcal{Y} \)
  - \( \alpha \) balances the weights of the two components.

**Table:**

<table>
<thead>
<tr>
<th>Object</th>
<th>Words</th>
<th>Dist(o, Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>o1</td>
<td>t1, t2</td>
<td>1.4</td>
</tr>
<tr>
<td>o2</td>
<td>t1, t3</td>
<td>2.8</td>
</tr>
<tr>
<td>o3</td>
<td>t1, t3</td>
<td>3</td>
</tr>
<tr>
<td>o4</td>
<td>t1, t3</td>
<td>3.2</td>
</tr>
<tr>
<td>o5</td>
<td>t2, t3</td>
<td>4.5</td>
</tr>
<tr>
<td>o6</td>
<td>t2, t3</td>
<td>8</td>
</tr>
<tr>
<td>o7</td>
<td>t2, t3</td>
<td>9</td>
</tr>
<tr>
<td>o8</td>
<td>t1, t3</td>
<td>8</td>
</tr>
</tbody>
</table>

**Figure:**

- Objects: \( o_1, o_2, o_3, o_4 \)
- Query: \( t_1, t_2, t_3 \)

**Table:**

<table>
<thead>
<tr>
<th>Partition</th>
<th>Objects</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1, t2, t3</td>
<td>αnull</td>
<td>6.2</td>
</tr>
</tbody>
</table>

**Figure:**

- Partition \( t_1, t_2, t_3 \) for the query \( t_1, t_2, t_3 \)
- Costs: \( \alpha o_2 + o_3 \)
Approximation Algorithm 1 – T2A1

- For each query keyword, find the nearest object covering it using an IR-tree. The group of these object serve as the result set.

\[ Q^Q = \{t_1, t_3, t_5\} \]

\[ \text{Cost}(q, \tilde{q}) = \text{Dist}(o_q, q) + \text{Dist}(o_q, o_k) \]

Approximation Algorithm 2 – T2A2

- Utilize the first approximation algorithm:

For each object \( o_k \) containing \( t_q \), issue a new query \( q_{new} = \{o_k, Q^Q\} \) and call T2A1.

Finally we select the group with the smallest cost.

Repeat until we reach an object further than the current cost value.

Top-k Groups Query

- Objects: \( p = \{q, \tilde{q}\} \) (location, text description)
- Query: \( q = \{q, \tilde{q}; k\} \) (location, keywords, # of objects)
- Ranking function

\[ \text{rank}_k(G) = \alpha \left( 1 - \beta \right) \frac{\text{dist}(q, \tilde{q}, G) + (1 - \beta) \text{diam}(G)}{\max D} + (1 - \alpha) \text{TR}_c(q; \tilde{q}, G) \]

- \( 0 \leq \alpha, \beta \leq 1 \)
- Distance: \( \text{dist}(q, \tilde{q}, G) = \min_{o_k \in G} \| q, o_k, \tilde{q} \| \)
- Diameter: \( \text{diam}(G) = \max_{o_k, o_l \in G} \| o_k, o_l, \tilde{q} \| \)
- The text relevance function favors large groups and groups where the query keywords are distributed evenly among group objects.
- Groups are disjoint
**Problem Definition**

- **Distance to the group**
  - Distance to the nearest object

- **Group diameter**
  - Maximum distance between two objects

---

**Road Networks**

**Problem Formalization**

- Road network graph $G$
  - A node represents a road junction point or a location, associated with a set of keywords
  - An edge represents a road segment

- Region $R$
  - A connected subgraph of $G$

- Nodes are weighted
  - Relevance to the query
  - Query-independent weights (e.g., popularity or rating) are also possible

---

**Road Networks**

**Hot Region** Query

- $q = (\lambda, q_r, \lambda)$
  - $\lambda$: a rectangular query range
  - $q_r$: keywords
  - $\lambda$: a road segment length constraint

- Retrieves the region with largest weight given the length constraint and the query range

- Example: $\lambda$ = the whole graph, $\lambda$ = 6

- Result: $<v2, v4, v5, v6>$

---

**Place ranking using GPS records, directions queries**
**Finding Spatial Web Objects**

- Massive volumes of location samples from moving objects are becoming available.
  - GPS location records \((oid, x, y, t)\)
  - Location records based on Wi-Fi and cellular positioning
- How can we utilize this content for identifying spatial web objects?
  - Can be used as a supplement to business directories
  - Potential benefit: more up to date

**From GPS Records to Places**

- Step 1: Extract stay points from raw trajectories
- Step 2: Cluster stay points with existing algorithms
- Step 3: Sample stay points from clusters, reverse geocode them, and obtain their semantics from yellow pages
- Step 4: Split and merge clusters to obtain semantic locations

**Step 1: Extract Stay Points**

- Stay: two consecutive records with a time gap larger than some threshold \(t_{th}\) (e.g., 10 minutes)
- Stay point: the first point in a stay (the end point)
- Data set: 76,139 stay points

**Step 2: Cluster Stay Points**

- Use existing spatial clustering algorithms
  - K-means: 7056 clusters
  - OPTICS: 7088 clusters

**Step 3: Sampling, Reverse Geocoding, Semantics**

- Randomly sample points from each cluster
- Use the Google Maps API for reverse geocoding
- Use a local yellow pages to get semantics

**Step 4: Splitting and Merging**

- Splitting
  - Cluster points in a cluster to obtain sub-clusters
  - Split a cluster if it has sub-clusters with different semantics
- Merge two clusters with similarity larger than a threshold
  - Similarity: consider user lists, semantics lists, average entry times, average stay durations

- Cannot merge with others; becomes a new cluster
- These merge to form a new cluster
### Experimental Study

- **Data**
  - Collection: 119 users in the period 01/01/2007 ~ 31/03/2008
  - Sampling: 1Hz
  - Records: 105,329,114

- **Step 1**
  - Stay point extraction: 76,139

- **Steps 2-4**
  - Clustering and cluster refinement: ~6,500

- **Clustering metrics**:
  - Purity
  - Entropy
  - NMI

### GPS-Based Place Ranking

- **Step 5**
  - Ranking metrics: Precision@n, MAP, nDCG, Runtime

- **Exploit different aspects of the location records**
  - The more visits, the more significant
  - The longer the durations of visits, the more significant
  - The more distinct visitors, the more significant
  - The longer the distances traveled to visit, the more significant
  - The more "near-by" significant places are, the more significant a place is.
  - The more a place is visited by objects that visit significant places, the more significant it is.

### Two-Layered Graph

- **$G_{UL}$**: a link represents a trip between two locations
- **$G_{LL}$**: a link represents a visit of a user to a location

### Results

<table>
<thead>
<tr>
<th></th>
<th>Rank-by-visits</th>
<th>Rank-by-durations</th>
<th>HITS-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>0.202</td>
<td>0.2126</td>
<td>0.062</td>
</tr>
<tr>
<td>P@20</td>
<td>0.45</td>
<td>0.45</td>
<td>0.1</td>
</tr>
<tr>
<td>P@50</td>
<td>0.36</td>
<td>0.38</td>
<td>0.12</td>
</tr>
<tr>
<td>nDCG@20</td>
<td>0.8261</td>
<td>0.8324</td>
<td>0.065</td>
</tr>
<tr>
<td>nDCG@50</td>
<td>0.7747</td>
<td>0.107</td>
<td></td>
</tr>
<tr>
<td>Runtime (s)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **ST-Unified performs the best**

- **Consider: the stay durations and distances between locations; performs the best.**

- **Exploit both types of links: performs better than $U-L$ and $L-L$.**

- **Considers the stay durations and distances between locations; performs the best.**

- **Treat all users equally, does not capture the experience of users.**

- **Ignores the dependency among locations, does not capture the relationships between locations.**

- **No normalization, locations visited many times by very few users gain too high rankings.**
Directions Query Based Place Ranking

- How can we use directions queries for assigning significance to places?
  - The queries will proliferate as navigation goes online.

- Idea: a query \((x \rightarrow y)\) is a vote that \(y\) is an important place.

- Exploit different aspects of the queries
  - Count-based: The more queries to \(y \in \mathcal{T}\), the more significant \(y\) is \((\mathcal{T})\).
  - Distance-based: The longer the distances \(x \rightarrow y\), the more the more significant \(y\) is.
  - Locality-based: The more queries \(x \rightarrow y\), the more significant \(y\) is for users close to \(x\).

Experimental Study

- Using query logs from Google

- The most obvious competitor is reviews and ratings.

- Similar quality as reviews
- Better coverage than reviews
- Better temporal granularity than reviews
  - Examples of finer temporal granularity: after-work bar, weekday lunch restaurant

Spatio-Textual Similarity Join

- Text Similarity Threshold \(T_{\text{text}}\)
- Spatial Distance Threshold \(T_{\text{distance}}\)
- Objective: Retrieve all pairs of geo-textual objects \((o_i, o_j)\) s.t.
  1. \(\text{TextSim}(o_i, o_j) \geq T_{\text{text}}\)
  2. \(\text{SpatialSim}(o_i, o_j) \leq T_{\text{distance}}\)

Other Functionality

Spatio-Textual Similarity Query

- A query region (rectangle)
- A set of keywords
- Thresholds of text similarity and spatial similarity

More Types of Queries

- Approximate String Search in Spatial Cai et al. ICDE’10
- Top-k Spatial Keyword Queries on Road Networks. Rocha-Junior and Narväg. EDBT’12
- Diversified Spatial Keyword Search On Road Networks. Zhang et al. FDBT’14
- Desks: Direction-Aware Spatial Keyword Search. Liu et al. ICDE’14
- Distributed Spatial Keyword Querying on Road Networks. Luo et al. EDBT’14
- Authentication of Moving Top-k Spatial Keyword Queries. Wu et al. TKDE’16
- Reverse Keyword Search for Spatio-Textual Top-k Queries in Location-Based Services. Lin et al. TKDE’16
- Keyword-Aware Continuous kNN Query on Road Networks. Zheng et al. ICDM’16
- ...

Fan et al.: SEAL: Spatio-Textual Similarity Search, PVLDB 12
Acknowledgments and Readings


Summary

- The web is going mobile and has a spatial dimension.
- Many queries have local intent
- Spatial keyword queries
  - k nearest neighbor queries
  - Continuous k nearest neighbor queries
  - Using nearby relevant content for place ranking
  - Retrieve a set of objects that collectively best satisfy a query
  - Retrieve k sets of objects that best satisfy a query

Next Steps

- Which functionality to serve when?
  - Ex: mineral water, dumplings
  - How can context be used for determining user intent?
- More sophisticated ranking!
  - Which signals to use?
  - How to combine them into a function (e.g., as a sum)?
  - Which weight parameters to use (e.g., a weight for each term)?
  - What is the relevant context for this?
    - Dependence on location
    - Dependence on keywords
    - Dependence on search history
    - Dependence on social network
    - Dependence on time
- Evaluation?
  - Which functionality is best where and when and for who?

Further Steps

- Structured queries and Amazon-style and social queries
  - Ample opportunities for much more customization of results
- Build in feedback mechanisms
  - “Figuring out how to build databases that get better the more people use them is actually the secret source of every Web 2.0 company”

- Avoid parameter overload
  - Problem vs. solution parameters
  - Hard-to-set, impossible-to-set parameters — relevance decreases exponentially with the number of such parameters

Acknowledgments and Readings

Thank you for your attention.

😊