Diversifying Top-k Point-of-Interest Queries via Collective Social Reach

Stella Maropaki
NTNU
Trondheim, Norway
stella.maropaki@ntnu.no

Sean Chester
University of Victoria
Victoria, Canada
schester@uvic.ca

Christos Doulkeridis
University of Piraeus
Piraeus, Greece
cdoulk@unipi.gr

Kjetil Nørvåg
NTNU
Trondheim, Norway
noervaag@ntnu.no

ABSTRACT

By "checking into" various *points-of-interest* (POIs), users create a rich source of location-based social network data that can be used in expressive spatio-social queries. This paper studies the use of popularity as a means to diversify results of top-k nearby POI queries. In contrast to previous work, we evaluate social diversity as a group-based, rather than individual POI, metric. Algorithmically, evaluating this set-based notion of diversity is challenging, yet we present several effective algorithms based on (integer) linear programming, a greedy framework, and r-tree distance browsing. Experiments show scalability and interactive response times for up to 100 million unique check-ins across 25000 POIs.

CCS CONCEPTS

• **Information systems** → **Database query processing**; **Location based services**.

KEYWORDS

top-k queries, socio-spatial queries, result diversification, nearest neighbours, best first search, linear programming

ACM Reference Format:
Stella Maropaki, Sean Chester, Christos Doulkeridis, and Kjetil Nørvåg. 2020. Diversifying Top-k Point-of-Interest Queries via Collective Social Reach. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3340531.3412097

1 INTRODUCTION

After the success of apps like Foursquare and Jiepang, many social networks now allow users to share their current geographic location with a "check-in." These location-based social networks (LBSNs) can incorporate both social and spatial factors into queries to improve the relevance of results, e.g., by considering how many users have checked into each nearby point-of-interest (POI) [8, 13].

The canonical approaches are: 1) to set thresholds on popularity and/or proximity; and 2) to rank POIs by a weighted sum of their popularity and proximity. In either case, treating popularity as independent for each POI can produce very homogeneous results. For example, of the 50 Parisian coffee shops with the most Foursquare user check-ins, 41 are Starbucks franchisees.[1] Obviously, diversity is a property of groups, not of individuals. Attempts had been made on diversification of POIs using the aforementioned canonical approaches [7, 9, 11] but with no focus on social aspect.

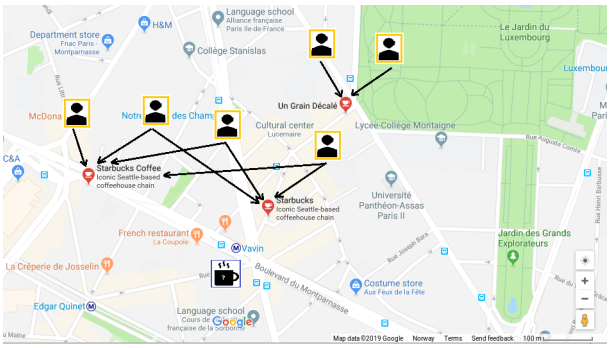


Figure 1: A user at Vavin in Paris searches for "café".

Here, we consider social diversity as the number of unique users checked into a group of POIs (a set union). We call this *collective social reach* and it leverages two insights: a) a larger set of unique users corresponds to a diverse set of underlying products [6]; and b) merchants in tourism should target first time check-ins (variety behavior) [14]. We assume weighted sums, but arbitrary predicates (particularly distance thresholds) could trivially be supported.

Figure 1 presents an example with a toy check-in set over cafés near the Vavin metro stop in Paris. The closest POIs are also the most popular. Techniques without collective reach (e.g., [8, 11, 13]) will return two instances of the same chain. These two Starbucks attract seven overlapping check-ins, but only four unique customers. On the other hand, *Un Grain Décalé* attracts two new customers; a top-2 ranking that includes reaches of 5–6 unique customers.

However, the diversity comes at a computational cost as the score of an object is no longer independent of other objects. To this end, we introduce eight algorithms with unique features. The most efficient of these, REHEAP, is based on a best-first incremental nearest neighbour algorithm [4]; we prove that it produces identical results to a conservative algorithm that pessimistically recomputes the best greedy solution k times incrementally (Theorem 4.3).

Contributions and Outline In this work, we:

- Introduce *collective social reach* to diversify top-k spatio-social POI queries (Section 2);
- Describe 8 algorithms to solve the problem (Section 4);
- Empirically compare all 8 algorithms (Section 5).

[1] https://www.4sqstat.com, "Coffee Shop" category, as of 06 May 2020

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3412097

2 PRELIMINARIES

Let $G = (U, P, E)$ be an unlabelled bipartite graph, where an edge $e_{ij} \in E = (u_i, p_j)$ indicates that user $u_i \in U$ has "checked in" to point $p_j \in P$. Moreover, let $\delta(p_i, p_i)$ denote the distance between points $p_i, p_j \in P$ and $D = \max_{p_i,p_j \in P} \delta(p_i, p_j)$. We define the *spatial proximity* of a query point q to point p_i as $\pi(q, p_i) = 1 - \delta(q, p_i)/D$ and to a set of points, P' as $\pi(q, P') = \sum_{p \in P'} \pi(p, q)$.

Moreover, let $U(P') = \bigcup_{p_i \in P'} \{u_j : (p_i, u_j) \in E\}$ denote the set of users who have checked into a point in P'. We measure the *diverse appeal*, $s(P')$, of a set of points, P', as the percentage of unique users that are linked to any point $p \in P'$: $s(P') = |U(P')|/|U|$.

Given G and a user-specified query point q, our objective is to determine the k points P' that maximise a normalised, weighted trade-off of spatial proximity and diverse appeal:

$$f(G, q, k, \alpha, P') = \alpha \cdot \pi(q, P')/k + (1 - \alpha) \cdot s(P') \tag{1}$$

where the weight α is used to express this trade-off.

When $\alpha = 1$, the problem reduces to standard spatial k-nearest neighbours; when $\alpha = 0$, to the NP-hard maximum coverage problem. A set of very popular points does not necessarily maximise diverse appeal if it attracts overlapping users. Arbitrary selection predicates can be applied to both U and P; e.g., U could be restricted to the immediate or 2-hop neighbourhood of the query user, and P could be restricted to a certain category of points-of-interest, if this additional data, e.g., a social network, is available. For example, this query could be issued for k diverse, nearby coffee shops, as frequented by friends and friends of friends.

3 RELATED WORK

This paper relates to works in the three broad areas described below.

Location selection The work in [5] quantifies a location's popularity based on how many users have the location in their k nearest neighbours results. Similarly, location selection based minimisation of the average distance between a customer and his nearest facility is studied in [2, 10]. Our work differs from these papers, as (a) we exploit user check-ins rather than user locations, and (b) we minimise the sum of distances. Saleem et al. [12] identify sets of influential locations that have a large geographical impact with a temporal constraint; they follow a count-based approach, whereas we adopt a set-based approach. Location recommendations based on user check-in data using different approaches, including social relationships, is studied in [15]. Similarly, the *maximum covering location problem* [3] identifies locations for facilities in order to service the maximum amount of the population as possible.

Diversified retrieval Shameem et al. [9] combine coverage and diversity for item recommendation. Qian et al. [11] retrieve POIs using the union of associated topics as a measure of diversity to which a threshold is applied. From a modelling perspective, this resembles our problem (with topics instead of users); however our work incorporates diversity *into the ranking function*. Similarly, in [7], coverage and diversity are maximised for spatio-temporal posts by a weighted sum of diversity and coverage using thresholds.

User mobility In GeoLife [16], historical user trajectories are analysed to define user similarity and POI visits are used as a measure of popularity, but ignoring social connections. Socio-spatial search is studied in [8] to retrieve users (instead of locations) with a weighted sum of spatial and social similarity, while a variety of ranking functions is used in [1]. Sohail et al. [13] compute a weighted sum of POI distance and 1-hop check-in neighbourhood size using both top-k and skyline variants. None of these works considers POI diversity.

$$\textbf{maximise}: \frac{(1-\alpha)}{|U|} \sum_{u_i} y_i + \frac{\alpha}{k} \sum_{p_j} \pi(q, p_j) x_j$$

$$subject\ to: k \geq \sum_{p_j} x_j,$$

$$y_i \leq \sum_{u_i \in checkins(p_j)} x_j,$$

$$\{y_i, x_j\} \in \{0, 1\}.$$

Figure 2: Integer Linear Programming Formulation

4 PROPOSED ALGORITHMS

1-Pass Baselines (Dist**,** User**,** 1P-Score**)** We define three single-pass greedy baseline algorithms, mostly to study the characteristics of the problem. All three algorithms iterate every $p \in P$ and compute a heuristic score. They maintain a heap of the k points that maximise their respective heuristics. The heap contains the proposed solution once all $p \in P$ have been processed.

They are differentiated by their heuristics: Dist calculates just spatial proximity; User calculates just diverse appeal; and 1P-Score calculates the full, α-weighted objective function.

k-Pass Baseline (kP-Score**)** The 1-pass baselines are clearly limited as they may select points p_i with duplicate overlapping users. Our k-pass baseline takes the opposite extreme: it behaves like 1P-Score, but only selects the highest scoring point and inserts it in the intermediate solution P'. Thereafter, it recomputes the *contribution*, $\Delta(p, P')$ for every point $p \notin P'$, explicitly calculating the $|U(p) \cup U(P')|$, as per this greedy equation:

$$\begin{aligned} \Delta(p, P') &= f(G, q, k, \alpha, \{p\} \cup P') - f(G, q, k, \alpha, P') \\ &= \alpha \cdot \pi(q, p)/k + (1 - \alpha) \cdot s(U(p) \setminus U(P')). \end{aligned} \tag{2}$$

This process is repeated k times to progressively produce a solution with k points. We can optimise the score calculation with Eq. 2 by incrementally maintaining an interim solution for each P', $|P'| < k$, the value $\pi(q, P')$ and the set $U(P')$.

Linear Programming (ILP**,** LP-round**)** Figure 2 gives an integer linear programming (ILP) formulation, based on the analogous ILP formulation for the maximum coverage problem. The unknowns y_i, x_j are binary variables indicating whether the solution covers user u_i and POI p_j; i.e., iff $x_j = 1$, then POI p_j is in the solution and iff $y_i = 1$ then user u_i has checked into one of those k POIs.

The constraints indicate that $\leq k$ binary POI variables x_j can be set and that a binary user variable y_i can only be set if at least one x_j, corresponding to a POI that user u_i has checked into, has also been set; i.e., it produces an induced subgraph with all neighbours of k POIs. ILP is optimal, but the problem is intractable.

LP-round first relaxes the problem to a solvable linear program by allowing all real-valued solutions $0 \leq \{y_i, x_j\} \leq 1$. Naturally, this produces a higher score than the ILP, but does not usually correspond to a real solution as the solution is non-integral: only if all values are $\in \{0, 1\}$ can a set of POIs be extracted. To build a valid solution, we select the k largest x_j variables. We use ILP to evaluate the accuracy of other algorithms.

Algorithm 1 RTree and Reheap algorithms

```
1:  function Query(User-R-tree R, query parameters q, α, k)
2:      P' ← ∅
3:      initialise a priority queue Q;
4:      add all the entries {e_i} of R's root to Q, score = Δ(e_i, ∅)
5:      while Q is not empty and |P'| < k do
6:          e ← Q.pop()
7:          if e is internal node then
8:              add all children {e_i} of e to Q, score = Δ(e_i, P')
9:          else                                          ▷ e is a POI p
10:             if Δ(p, P') ≥ Q.top().score or Not Reheap then
11:                 add POI p to result P'
12:             else                                      ▷ score for p is obsolete
13:                 add p to Q, recomputed score = Δ(p, P')
        return P'
```

Indexed Algorithms (RTree**,** Reheap**)** We also present two methods that leverage a spatial index. P is indexed in an R-Tree and R-Tree nodes contain a pointer to a list of all users who have checked into a POI in that subtree. We call this a User-R-Tree.

We use the same incremental maintenance ideas of kP-Score, but generalise them for a User-R-tree internal node, e. Let $\pi(q, e)$ denote the spatial proximity of query point q to the nearest point on/in the MBB at User-R-tree entry e, and let $U(e)$ denote the union of all users checked into a point in the subtree rooted at e. We generalise Eq. 2 to a User-R-Tree entry, e, as follows:

$$\Delta(e, P') = \alpha \cdot \pi(q, e)/k + (1 - \alpha) \cdot s(U(e) \setminus U(P')). \tag{3}$$

User-R-Tree is used in a *best-first* incremental nearest neighbour algorithm, a.k.a., distance browsing [4], (RTree in Algorithm 1). Query q begins at the root node and every child entry e_i is pushed onto a priority queue Q based on score $\Delta(e_i, \emptyset)$. Thereafter, we pop entries off Q and push their children entries on; if the child entry corresponds to a point, then, as in [4], we add it to the solution.

However, unlike spatial kNN, the quality of RTree deteriorates as solution points are discovered, because the diverse appeal component of the scoring function depends on P': once an MBB is heaped, its score is never revisited, even though new solution points are discovered. Thus, the score reflects an obsolete state. Reheap addresses this by reheaping points. Whenever it deheaps a leaf/point p, rather than immediately add it to the solution, it first recomputes $\Delta(p, P')$. If this would still top Q, then it is added to the solution; otherwise, it is reheaped with its recalculated score. The algorithm terminates once discovering k points that do not need to be reheaped.

Lemmata 4.1–4.2 inspire Lines 10–11 of Algorithm 1: no point enqueued in Q will merge with P' to produce a higher score than will p; and every point not yet enqueued will produce a score not better than its parent entry that is currently enqueued. I.e., $\forall p' \notin P', \Delta(p, P') \geq \Delta(p', P')$. As a result, we get Theorem 4.3.

Lemma 4.1. *Given $p, p' \notin P''$ and $P' \subseteq P''$, $\Delta(p, P'') \geq \Delta(p', P')$ $\implies$ $f(G, q, k, \alpha, P'' \cup \{p\}) \geq f(G, q, k, \alpha, P'' \cup \{p'\})$.*

Proof. It suffices to show that $\Delta(p', P'') \leq \Delta(p', P')$. Given the independence of $\pi()$ on P', this reduces to: $s(U(p) \setminus U(P'')) \leq s(U(p) \setminus U(P'))$, which follows trivially from $P' \subseteq P''$. □

Lemma 4.2. *Given an R-tree entry e and a point $p \notin P'$ in the subtree rooted at e, $\Delta(p, P') \leq \Delta(e, P')$.*

Proof. Observe that e produces a not-lower score than p with respect to both spatial proximity and diverse appeal. Spatially, the MBB at e must contain p; therefore, the closest point to q on the MBB is at least as close as p. For $s()$, observe that $U(p) \subseteq U(e)$. The proof then follows as in Lemma 4.1. □

Thus we have Theorem 4.3, which follows from Lemmata 4.1-4.2:

Theorem 4.3. *k*P-Score *and* Reheap *produce identical results.*

5 EXPERIMENTS

Setup Experiments are run on a server with dual 12-core 2.30GHz Intel Xeon Gold 5118 CPUs, with 128 GB RAM, using Ubuntu 18.04. All algorithms are memory-resident and are implemented in C++ using g++ version 7.4.0. (I)LP algorithms use the GNU LP Kit.[2]

Datasets The Yelp dataset[3] has user reviews in 11 cities; we construct a graph of 27K points in Las Vegas (YelpLV) and 665K users who have rated those POIs, generating 2.2M edges. Including the other 10 distant cities isn't particularly meaningful when evaluating by spatial proximity and would be filtered by a user predicate, anyway. We also use synthetic data to analyse scalability and performance relative to graph density ($|E|/(|P| + |U|)$). These datasets are created by taking $|P|$ points from YelpLV, creating $|U|$ users, and generating $|E|$ reviews (i.e., edges) between them uniformly at random. The synthetic dataset S1 ($|P| = 50$, $|U| = 200$, $|E| = 2K$) is created at the limit for which exact solutions can be found (using ILP). For the scalability experiments, we use a synthetic dataset S2 ($|P| = 25K$, $|U| = 1M$, $|E| = 100M$) and vary $|E|$. To study the impact of $|P|$, we create datasets with constant $|U| = 1M$ and a constant ratio between $|E|/|P| = 4000$; the largest such dataset mirrors the experiment that vary $|E|$.

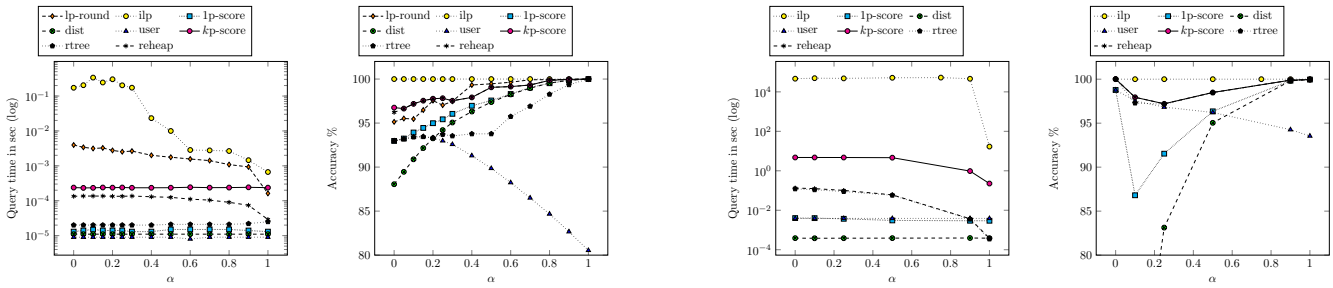
We report the median of 10 random query points: points taken from the dataset and distorted by a distance of ≤ 0.01.

Effect of α Figure 3 illustrates how α affects the query time and accuracy of each algorithm on real (a) and small synthetic (b) data ($k = 10$). ILP can run on small instances such as these and produces an exact solution. Accuracy is reported as the score of each algorithm's solution, relative to the score produced by ILP.

The naive single-pass baselines (Dist, User, and 1P-Score) have query time independent of α. Dist and User don't use α at all. 1P-Score only uses it as a constant in its heuristic. However, their respective accuracy illustrates their problem well: 1P-Score corresponds both to the standard $(1 - \frac{1}{e})$-approximation algorithm for maximum coverage ($\alpha = 0$) and the standard best-first algorithm for kNN ($\alpha = 1$). User and Dist at their respective strengths score equally with 1P-Score, then gradually degrade toward the right/left. On YelpLV, Dist achieves $\approx 0\%$ accuracy at $\alpha = 0$ (cut from figure for readability), illustrating the problem's dual nature.

RTree is competitive with the single-pass baselines in terms of query time, but has consistently poor accuracy. As points are progressively added to the solution, the priority queue becomes less accurate, as it does not take into account the overlap between the

[2] https://www.gnu.org/software/glpk/
[3] https://www.kaggle.com/yelp-dataset/yelp-dataset



(a) Synthetic dataset S1 **(b) YelpLV dataset**

Figure 3: Running times and accuracy of algorithms as a function of α

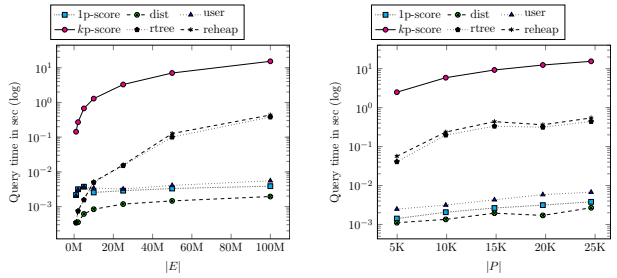


Figure 4: Scalability of algorithms on synthetic datasets relative to the number of check-ins or POIs ($\alpha = 0.5$)

enqueued points and those newly added to the solution. Its solution quality continually degrades as k grows (not shown).

kP-Score and Reheap produce equivalent solutions, but by different means. The former recomputes the score estimate for every point after every progressive addition to the solution; the latter only recomputes for those points that top the priority queue. Thus, Reheap achieves much better performance, as it does substantially fewer recomputations. In fact, its query time is very close to that of RTree, as few points actually need to be reheaped (not shown).

Scalability Figure 4 shows scalability as data grows in terms of the edge set (left, increasing density) or point set (right, constant density). We set $\alpha = 0.5$, a consistent inflection point for RTree in Figure 3. ILP and LP-round are omitted (too slow to generate constraint matrices of this size); so, accuracy is not reported.

The one-pass baselines are largely unaffected by $|E|$ (left) but degrade with an increase in $|P|$, as they calculate scores for each $p \in P$. In contrast, the indexed (RTree, Reheap) and kP-Score algorithms *are* affected by the density of the bipartite network; their performance degrades with increasing $|E|$. We conjecture that this arises from more expensive calculations of $s(P')$, which involves set operations on increasingly longer lists of users. Increasing $|P|$ with a fixed check-in ratio degrades the performance of all three algorithms, though the indexed solutions absorb the increase better because they only need to process a larger tree, not every point.

Memory usage and indexing cost (not shown) All approximate algorithms reach a memory high water mark that is nearly linear in $|E|$. At $|E| = 100$M, the non-index-based algorithms require ≈ 3 GB of memory, while the index-based algorithms require ≈ 5.3 GB. Indexing time (for RTree and Reheap) is also close to linear in $|E|$ and requires ≈ 11.5 seconds at $|E| = 100$M.

6 CONCLUSION

We introduced a novel notion of diversity—the uniqueness in the set of users—and a POI kNN problem that linearly combines spatial promixity and our diversity measure. This yields an interesting theoretical problem of solving a linear combination of a near-linear and an NP-hard problem. We propose 8 intuitive algorithms to solve this, one of which (Reheap) achieves sub-decisecond performance on a real dataset with > 2 million user reviews, while also achieving accuracy very close to the exact ILP solution that does not scale.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement **CoupledDB** No 753810 (S. Chester); the Hellenic Foundation for Research and Innovation and the General Secretariat for Research and Technology, under grant agreement No 1667 (C. Doulkeridis); and the Norwegian Research Council grant **ExiBiDa** (S. Maropaki and K. Nørvåg). We thank Orestis Telelis for his help with the GNU LP Kit.

REFERENCES

[1] Nikos Armenatzoglou, Ritesh Ahuja, and Dimitris Papadias. 2015. Geo-Social Ranking: functions and query processing. *VLDB J* 24, 6 (2015), 783–799.
[2] Yu-Chi Chung, I-Fang Su, and Chiang Lee. 2018. k-most suitable locations selection. *GeoInformatica* (2018), 1–32.
[3] Richard Church and Charles ReVelle. 1974. The Maximal Covering Location Problem. *Papers of the Regional Science Association* 32 (1974), 101–118.
[4] Gísli R Hjaltason and Hanan Samet. 1998. Incremental distance join algorithms for spatial databases. In *ACM SIGMOD*. 237–248.
[5] Jin Huang et al. 2011. Top-k Most Influential Locations Selection. In *CIKM*.
[6] Rong-Hua Li and Jeffery Xu Yu. 2013. Scalable Diversified Ranking on Large Graphs. *TKDE* 25, 9 (2013), 2133–2146.
[7] Paras Mehta et al. 2016. Coverage and Diversity Aware Top-k Query for Spatio-Temporal Posts. In *SIGSPATIAL*. 37:1–37:10.
[8] Kyriakos Mouratidis et al. 2015. Joint Search by Social and Spatial Proximity. *TKDE* 27, 3 (2015), 781–793.
[9] Shameem A. Puthiya Parambath et al. 2016. A Coverage-Based Approach to Recommendation Diversity On Similarity Graph. In *RecSys*. 15–22.
[10] J Qi et al. 2012. The min-dist location selection query. In *ICDE*. 366–377.
[11] Zhihu Qian et al. 2018. Diversified Spatial Keyword Query on Topic Coverage. In *Workshop on Mobile Web Data Analytics*. 24–34.
[12] Muhammad Aamir Saleem et al. 2017. Location influence in location-based social networks. In *WSDM*. 621–630.
[13] Ammar Sohail et al. 2018. Social-Aware Spatial *Top-k* and *Skyline* Queries. *Comput. J.* 61, 11 (2018), 1620–1638.
[14] Iis P Tussyadiah. 2012. A Concept of Location-Based Social Network Marketing. *Journal of Travel & Tourism Marketing* 29, 3 (2012), 205–220.
[15] Hao Wang et al. 2013. Location Recommendation in Location-based Social Networks using User Check-in Data. In *SIGSPATIAL/GIS*. 364–373.
[16] Yu Zheng et al. 2010. GeoLife: A Collaborative Social Networking Service among User, Location and Trajectory. *IEEE Data Eng Bull* 33, 2 (2010), 32–39.